



Département des Technologie de l'information et de la
communication (TIC)
Filière Télécommunications
Orientation Sécurité de l'information

Travail de Bachelor

Gestionnaires de mots de passe : quelle sécurité ?

Étudiante

Enseignant responsable

Année académique

Noémie Plancherel

Prof. Sylvain Pasini

2022-2023

Yverdon-les-Bains, le 8 décembre 2022

Département des Technologie de l'information et de la communication (TIC)
Filière Télécommunications
Orientation Sécurité de l'information
Étudiante : Noémie Plancherel
Enseignant responsable : Prof. Sylvain Pasini

Travail de Bachelor 2022-2023
Gestionnaires de mots de passe : quelle sécurité ?

Résumé publiable

Dans ce travail... Ceci est le résumé publiable...

Étudiant :	Date et lieu :	Signature :
Noémie Plancherel
Enseignant responsable :	Date et lieu :	Signature :
Prof. Sylvain Pasini

Préambule

Ce travail de Bachelor (ci-après TB) est réalisé en fin de cursus d'études, en vue de l'obtention du titre de Bachelor of Science HES-SO en Ingénierie.

En tant que travail académique, son contenu, sans préjuger de sa valeur, n'engage ni la responsabilité de l'auteur, ni celles du jury du travail de Bachelor et de l'Ecole.

Toute utilisation, même partielle, de ce TB doit être faite dans le respect du droit d'auteur.

HEIG-VD

Vincent Peiris
Chef de département TIC

Yverdon-les-Bains, le 8 décembre 2022

PRÉAMBULE _____

vi _____

Authentification

La soussignée, Noémie Plancherel, atteste par la présente avoir réalisé ce travail et n'avoir utilisé aucune autre source que celles expressément mentionnées.

Yverdon-les-bains, le 8 décembre 2022

Noémie Plancherel

AUTHENTICATION _____

Remerciements

Je souhaiterais tout d'abord remercier Sylvain Pasini pour m'avoir accompagné durant tout mon travail.

Puis, je souhiaterais également remercier mes proches pour m'avoir soutenu et avoir relu mon rapport afin de rendre la lecture plus agréable.

REMERCIEMENTS _____

x _____

Cahier des charges

Résumé du problème

De nos jours, les gestionnaires de mots de passe sont des outils très fréquemment utilisés. En effet, une bonne pratique est d'utiliser un mot de passe par service. De cette manière, si un service est compromis et le mot de passe divulgué, cela n'impacte pas les autres services. Il est également très important de choisir un mot de passe fort qui ne contient pas d'éléments facilement prévisibles et qui pourrait être brute-forcé rapidement.

Les gestionnaires de mots de passe permettent principalement de faciliter le stockage des mots de passe qui demandent d'être de plus en plus longs et complexes, de manière à ne pas les réutiliser. Ils permettent également d'ajouter une couche sécuritaire aux mots de passe en les stockant de manière sécurisée et en offrant la possibilité de générer des mots de passe forts.

Ces applications offrent plusieurs fonctionnalités sous la forme de différents types ; elles permettent, entre autres, l'utilisation du cloud afin de stocker les mots de passe sur les serveurs du fournisseur pour faciliter la synchronisation des données entre plusieurs devices (mobile, montre, navigateur, etc.). Certains gestionnaires de mots de passe sont également fréquemment utilisés au sein d'entreprises pour permettre le partage de données. Les entreprises vont généralement utiliser une solution self-hosted où elles auront leur propre infrastructure et stockage. Il existe des extensions de navigateur qui proposent le remplissage automatique de mots de passe dans les formulaires de connexion. Enfin, il y a également des applications en local qui vont limiter leur utilisation à un seul appareil.

Étant donné que les utilisateurs se reposent grandement sur les gestionnaires de mots de passe, il est important de s'assurer que ces logiciels satisfassent un certain nombre de principes de sécurité ainsi qu'une implémentation robuste afin d'éviter tout vol ou perte de données.

Objectifs

Ce travail de Bachelor vise à comprendre les menaces d'un gestionnaire de mots de passe, premièrement de manière générique, puis sur des produits spécifiques, sélectionnés à la suite d'une étude complète, en analysant la sécurité sous différents angles (stockage, mémoire, réseau, cryptographie, etc.).

Le travail est réalisé en deux parties distinctes ; une première partie qui est une étude approfondie et complète sur les gestionnaires de mots de passe. Elle permet d'analyser les menaces des différents type de gestionnaires de mots de passe et de présenter les exigences sécuritaires qu'il serait nécessaire de garantir. Elle va également se concentrer sur une étude de marché avec une comparaison de plusieurs gestionnaires de mots de passe existants sous différents aspects.

La deuxième partie du travail se concentrera tout d'abord sur la sélection de quelques candidats (environ 4) en fonction de critères établis au préalable. Ensuite, le but est d'évaluer la sécurité de manière complète de chaque gestionnaire de mot de passe sélectionné ; chaque élément choisi est analysé et évalué en fonction de différents critères comme les choix cryptographiques utilisés, le stockage, ou encore l'architecture de l'application.

Livrables

Les livrables seront les suivants :

1. Une documentation contenant :
 - Présentation des différents types de gestionnaires de mots de passe
 - Étude de marché
 - Une analyse de menaces de différents types de gestionnaires de mots de passe
 - Spécification des exigences sécuritaires à garantir
 - (a) Analyse sécuritaire des quelques candidats représentatifs (environ 4) :
 - Sélection de candidats pour la suite du travailchaque analyse se décomposera ainsi :
 - Sélection de critères d'analyse
 - Analyse complète de chaque aspect
 - Rapport des faiblesses trouvées au fabricant
 - (b) Synthèse des résultats
 - Comparaison entre chaque candidat

Déroulement

En se référant aux dates validées par le responsable de filière ISC, M.Donini, le travail de Bachelor débute le 20 septembre 2022 et se termine au plus tard le 10 février 2023. Il y a 3 dates clés incluant des rendus :

- **14 octobre 2022** - rendu du rapport intermédiaire
- **14 décembre 2022** - rendu du rapport final
- **23 janvier au 10 février 2023** - soutenance du travail de bachelor

Etant donné, que la soutenance du travail implique l'intervention d'un expert, la date doit être définie entre tous les intervenants.

Le volume du travail de bachelor est de 15 crédit ECTS, soit 450 heures. Le rapport intermédiaire représente 150 heures de travail.

Au niveau de la répartition de la charge de travail, cela représente environ 45h/semaine jusqu'au rendu, soit le 14 décembre, car le travail se fait à 100%.

Planning

Le travail de bachelor sera séparé en plusieurs tâches et sous-tâches différentes qui permettront de répartir plus facilement le travail sur des périodes de plusieurs semaines. Ci-dessous, le planning détaillé avec toutes les tâches :

1. Préparation
 - Rédaction du cahier des charges
 - Planification
 - Recherches initiales et introduction
2. Étude de marché
 - Recherche et explication des différents types de gestionnaires de mots de passe
 - Comparaison des fonctionnalités, du prix et des plateformes disponibles
 - Analyse du marché actuel et de la demande
 - Récapitulatif
3. Étude sécuritaire
 - Présentation de la sécurité implémentée dans les gestionnaires de mots de passe
 - Identification et analyse des menaces potentielles
 - Rédaction des exigences sécuritaires
4. Sélection
 - Mise en place des critères de sélection des candidats
 - Sélection des candidats
5. Analyse sécuritaire (pour chaque candidat)
 - Identification et rédaction des critères d'analyse
 - Analyse sécuritaire de chaque aspect
6. Synthèse (pour chaque candidat)
 - Synthèse des résultats
 - Rapport des faiblesses au fabricant

7. Synthèse générale

- Comparaison de tous les résultats
- Conclusion du travail

8. Documentation

- Rédaction du rapport
- Lecture / visualisation de documents
- Tenue d'un journal de travail

Un diagramme de Gantt a également été effectué afin de pouvoir visualiser le planning et ajouter des périodes de temps :

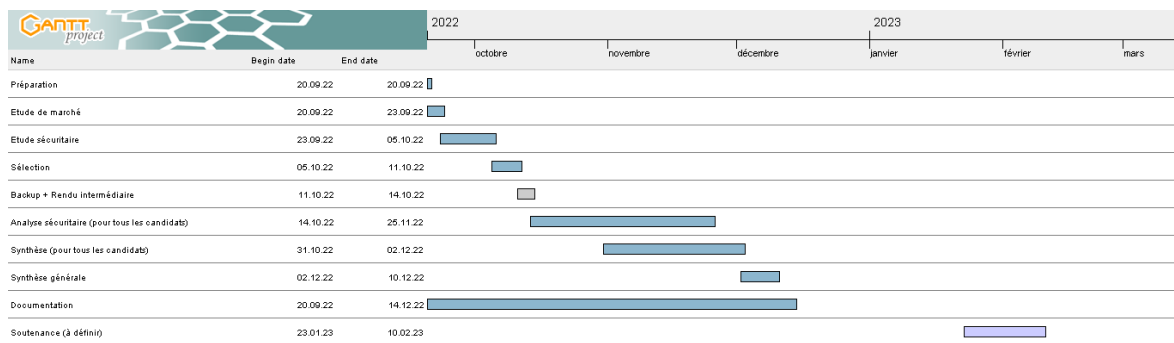


FIGURE 1 – Planning du travail de Bachelor

Table des matières

Préambule	v
Authentification	vii
Remerciements	ix
Cahier des charges	xi
Planning	xv
1 Introduction	1
1.1 Fonctionnement général	2
1.2 Types et fonctionnalités variées	3
1.2.1 Stockage des données	3
1.2.2 Applications	4
1.3 Fonctionnalités	4
1.4 Sécurité	5
2 Étude de marché	7
2.1 Fonctionnalités	7
2.2 Plateformes	9
2.3 Prix	10
2.3.1 Particuliers	10
2.3.2 Entreprises	10

2.4	Récapitulatif de l'étude	11
3	Informations préliminaires	13
3.1	Attaques sur mot de passe	13
3.2	Complexités	14
3.2.1	Entropie	15
3.3	2FA	15
4	Étude sécuritaire	17
4.1	Implémentation de la sécurité dans les gestionnaires de mots de passe	17
4.1.1	Les gestionnaires browser-based	18
4.1.2	Les gestionnaires local-based	21
4.1.3	Les gestionnaires cloud/local-based	24
4.2	Partage d'informations	26
4.3	4 états du gestionnaire de mot de passe	29
4.3.1	État <i>Not Configured</i>	29
4.3.2	État <i>Not Running</i>	29
4.3.3	État <i>Unlocked State</i>	29
4.3.4	État <i>Locked State</i>	30
4.4	Algorithmes cryptographiques	30
5	Analyse de menaces	33
5.1	Modélisation de menaces	33
5.1.1	Définition des critères	35
5.1.2	Établissement du contexte	36
5.1.3	M1 Application haut-niveau	39
5.1.3.1	Identification des risques	39
5.1.3.2	Analyse des risques	42
5.1.4	M2 Backup	43
5.1.4.1	Identification des risques	43

5.1.4.2	Analyse des risques	45
5.1.5	M3 Synchronisation	47
5.1.5.1	Identification des risques	47
5.1.5.2	Analyse des risques	49
5.1.6	M4 Partage	51
5.1.6.1	Identification des risques	51
5.1.6.2	Analyse des risques	53
5.1.7	M5 Application bas-niveau	54
5.1.7.1	Identification des risques	54
5.1.7.2	Analyse des risques	57
5.1.8	Évaluation des risques	57
5.1.9	Traitement des risques	57
5.2	Exigences sécuritaires à respecter	58
6	Sélection des candidats	61
6.1	Critères de sélection	61
6.2	Choix	62
7	LastPass	65
7.1	Environnement	65
7.2	Critères d'analyse	65
7.3	Master password	66
7.3.1	Entropie	67
7.3.2	Brute-force authentification	67
7.4	Choix cryptographiques	68
7.4.1	Chiffrement	68
7.4.2	Dérivation des clés	68
7.4.3	Authentification	69
7.5	Fonctionnalités proposées	69
7.5.1	Génération des mots de passe	69

7.5.2	Auto-complétion	71
7.5.3	Presse-papier	74
7.6	Stockage	75
7.7	Mémoire	76
7.8	Récapitulatif de l'analyse	79
8	KeePass	83
8.1	Environnement	83
8.2	Critères d'analyse	84
8.3	Master password	84
8.3.1	Brute-force authentification	85
8.4	Choix cryptographiques	86
8.4.1	Chiffrement	86
8.4.2	Dérivation des clés	86
8.4.3	Authentification	87
8.5	Fonctionnalités proposées	87
8.5.1	Génération de mots de passe	87
8.5.2	Presse-papier	91
8.6	Stockage	92
8.7	Mémoire	95
8.8	Récapitulatif de l'analyse	95
9	Firefox	97
9.1	Critères d'analyse	97
10	Conclusion	99
	Bibliographie	101
	Liste des figures	105
	Liste des tableaux	107

Liste des listings	111
A Annexes LastPass	113
A.1 Génération de mots de passe	113
A.2 Accès mémoire	115
B Annexes KeePass	119
B.1 Génération de mots de passe	119
C Journal de travail	121

PLANNING _____

Chapitre 1

Introduction

Pour un utilisateur lambda, il peut être difficile de se souvenir de tous ses mots de passe tout en s'assurant d'en utiliser un différent pour chaque service afin d'éviter tout vol de données dû à la compromission d'un service, et donc la compromission d'un seul mot de passe au lieu de plusieurs. Typiquement dans ces situations, l'utilisateur va naturellement utiliser des mots de passe simples, qui sont facilement mémorisables. Comme, par exemple, utiliser son prénom et sa date de naissance, "123456" ou encore "qwerty". De plus, il est plus simple d'utiliser le même mot de passe pour chacun de ses comptes, afin d'en mémoriser uniquement un seul.

Cependant, même si l'unique mot de passe qu'on utilise est fort et aléatoire, il n'est pas garanti à 100% qu'on soit la cible d'aucun attaquant et si une attaque est réalisée, toutes nos données personnelles sont exposées. Par exemple, une personne malveillante pourrait attaquer un service, comme Google, et récupérer quelques données utilisateurs, dont le mot de passe, ce qui lui permettrait par la suite d'utiliser cet unique mot de passe pour usurper l'identité de l'utilisateur sur d'autres services.

Ainsi, dans ce genre de cas, les gestionnaires de mots de passe interviennent et peuvent faciliter le quotidien de la plupart des utilisateurs.

Néanmoins, avant d'aller plus en détails sur le fonctionnement des gestionnaires de mots de passe, nous pouvons mettre en avant la réticence des utilisateurs quant à l'utilisation régulière de ces derniers. D'après un sondage lancé par PasswordManager[37] aux États-Unis avec des personnes âgées de 18-55+, seulement 22.5% utilisent des gestionnaires de mots de passe. Une autre étude de Security.org[44] de novembre 2021, également lancée aux États-Unis, ressort les mêmes statistiques ; 20% des utilisateurs utilisent ces derniers. Les autres solutions communes pour stocker ses identifiants sont, comme nous l'avons cité plus haut, la mémorisation, le papier, la réutilisation, etc. Nous pouvons sans aucun doute déclarer que ces méthodes ne sont pas très sécurisées.

Toutefois, nous pouvons expliquer cette réticence à l'aide des études citées ci-dessus qui déclarent qu'au niveau des utilisateurs qui n'utilisent pas de gestionnaires de mots de passe, 70% ne font pas confiance à la sécurité qu'elles fournissent, ils pensent que leur application pourrait être hackée. Certains, ne font également pas confiance aux constructeurs de ces dernières en pensant qu'ils volent leurs données.

En contradiction à ces avis populaires, en se basant sur un sondage de 2022 de bitwarden[6], globalement, 35% des utilisateurs sont plus inquiets des cyberattaques par rapport à l'année 2020. Nous pouvons justifier ces inquiétudes avec le fait que le nombre de cyberattaques effectuées en 2021 a augmenté (en partie dû au COVID-19 et au *home office*). Le DBIR de 2022 (Data Breach Investigations Report)[45] indique qu'il y a eu une augmentation de 13% des vols de données (dont 85% font partie de vulnérabilités humaines).

Avec toutes ces statistiques, nous pouvons constater que malgré une utilisation encore trop basse des gestionnaires de mots de passe, les particuliers s'y intéressent progressivement dû aux attaques et vols de données en progression constante. Cependant, il y a un manque de confiance général sur ces derniers, surtout envers les constructeurs. C'est pourquoi, la sécurité parfaite au sein des gestionnaires est un sujet très important si l'on souhaite augmenter la protection des données et éviter des vulnérabilités humaines (notamment l'utilisation de mots de passe trop de simple, comme "123456"). La sécurité "presque" parfaite des applications pourraient également baisser les vols de données par des personnes malveillantes.

Ainsi, le travail présent s'appuie sur les statistiques ci-dessus et va essayer d'évaluer au mieux la sécurité des gestionnaires de mots de passe actuellement sur le marché afin de faire gagner la confiance des utilisateurs vis-à-vis des ces applications et de les encourager à en utiliser plus fréquemment et quotidiennement.

1.1 Fonctionnement général

Les gestionnaires de mots de passe sont des applications, parfois multi-plateformes, qui vont permettre de stocker des informations sensibles telles que des mots de passe, numéros de carte de crédit ou encore des fichiers confidentiels. On peut les comparer à des coffres forts ou un trousseau de clés avec plusieurs "clés" qui nous donnent accès à différents services.

Ces derniers proposent un *master password* et / ou une *master key* qui va permettre d'accéder à l'ensemble des données secrètes. En conséquence, la sécurité repose sur un seul mot de passe principal, ce qui est très bénéfique pour les utilisateurs car ils n'ont qu'un mot de passe à retenir. Une fois l'accès à l'application, l'utilisateur a la possibilité de stocker des données, générer des mots de passe forts ainsi que se connecter à des services en ligne (remplissage de formulaire d'identification automatique).

1.2 Types et fonctionnalités variées

Différentes solutions de gestionnaires de mots de passe sont disponibles. Chacune offre des fonctionnalités variées et en plusieurs types différents en fonction du besoin de l'utilisateur et des fonctionnalités proposées.

1.2.1 Stockage des données

Pour le stockage de toutes les données sensibles ainsi que leur synchronisation, il existe 3 solutions différentes.

Local only

Il stocke toutes les données chiffrées en local sur le device de l'utilisateur. Tout est stocké sur un seul device de l'utilisateur. Ainsi, la synchronisation entre plusieurs appareils n'est pas proposée. Il est possible d'utiliser la même base de données sur d'autres devices, mais il est nécessaire d'effectuer la manipulation manuellement.

Ces gestionnaires fonctionnent donc en mode offline (hors-ligne). Ainsi, la sécurité est plutôt bonne comparé à la solution cloud car c'est du hors-ligne, cependant si on récupère / vole le device, la sécurité pourrait être compromise car il y aurait la possibilité d'avoir accès aux informations sensibles du gestionnaire de mots de passe, via notamment une gestion de mémoire mal gérée.

Local / sync cloud

Il propose de stocker les informations en local et également de pouvoir activer la synchronisation dans le cloud afin de stocker les données sur les serveurs du constructeur. Les données sont alors stockées en local et sur le cloud.

Sur ces gestionnaires, il est ainsi possible d'être en offline et dès que l'application est en ligne, toutes les données modifiées et / ou ajoutées sont synchronisées avec les serveurs. L'utilisateur a ainsi la possibilité d'avoir accès à ses données sur n'importe quel device (ordinateur, mobile, montre) et à tout moment.

À propos de la sécurité, elle repose sur le provider de l'application car toutes les informations sont stockées sur leurs propres serveurs et également sur les appareils de l'utilisateur.

Cloud only

Il propose de stocker uniquement les données dans le cloud. Ainsi, il est nécessaire d'avoir une connexion en temps tout et le hors ligne n'est pas possible. La sécurité repose entièrement sur les serveurs du constructeur.

1.2.2 Applications

Au niveau des applications, il existe plusieurs type d'applications différentes. Nous pouvons retrouver les suivantes.

Applications tierces

Elles sont créées et proposées par des constructeurs indépendants et autres que les constructeurs de systèmes d'exploitations ou navigateurs. Ces dernières peuvent être présentes sur plusieurs plateformes différentes : desktop, extension de navigateur ou encore application mobile.

En général, ces applications ont un stockage local et cloud, ce qui leur permet de proposer la synchronisation entre tous les appareils.

Navigateur (browser-based)

Les navigateurs les plus populaires, tels que Firefox, Safari ou Chrome offrent ce gestionnaire de mots de passe qui est directement inclus dans ces derniers. Ils vont faciliter la gestion et la sauvegarde de mots de passe de comptes de sites web.

Il y a également la possibilité de synchroniser toutes les données stockées entre tous les devices qui supportent le navigateur en question (Chrome, Firefox, Safari, etc.).

Pour certains navigateurs, les informations sont stockées et chiffrées en local sur le device de l'utilisateur. Si la synchronisation est activée, les données seront également stockées dans le cloud sur les serveurs du constructeur. Un problème de sécurité importante, et la disponibilité des mots de passe sur le navigateur, si aucun master password n'est configuré et qu'on a accès à la machine, les mots de passe sont accessibles en clair sur le navigateur.

Système d'exploitation

Certains systèmes d'exploitation proposent des gestionnaires de mots de passe intégrés. MacOS propose "iCloud Keychain" et Windows propose "Credential Manager" qui permettent les deux d'ajouter des identifiants et de les stocker. MacOS utilise le iCloud pour la synchronisation de toutes les données et stocke également les données en local sur l'appareil.

Ces derniers sont des solutions assez limitées car ils ne sont pas disponibles pour tous les appareils. Le gestionnaire de mots de passe Windows n'est seulement accessible depuis une seule machine et celui de MacOS est uniquement disponible sur les appareils Apple.

1.3 Fonctionnalités

Les gestionnaires de mots de passe proposent diverses fonctionnalités qui permettent de faciliter l'utilisation quotidienne des utilisateurs. Une des fonctionnalités les plus intéressantes, est la génération de mots de passe forts. Sur certains gestionnaires, il est possible d'ajouter

ses propres critères de génération (par exemple, 16 caractères, avec des chiffres et des lettres).

Une autre fonctionnalité proposée sur les gestionnaires en extension de navigateur, sont l'auto-complétion des champs de connexion. Lors d'une première connexion à un service, l'application va en général demander à l'utilisateur pour enregistrer les identifiants dans le gestionnaire et à chaque prochaine connexion, proposera automatiquement de remplir les champs.

Ce qui est plutôt utile avec un gestionnaire de mots de passe, c'est que certains proposent de stocker d'autres éléments que des identifiants. On a la possibilité d'également stocker des informations sensibles comme notre passeport, nos cartes de crédits ou des contrats importants.

Une fonctionnalité existante, utile pour les familles et les entreprises, permet de partager des données entre utilisateurs. Il est possible de soit créer un identifiant à partager ou de créer un dossier partagé.

Finalement, nous l'avons déjà cité précédemment, certaines applications permet la synchronisation entre plusieurs appareils d'un seul utilisateur. Les données sont stockées et transitent via le cloud. Cela peut grandement faciliter l'utilisation des gestionnaires car il est possible d'avoir accès à nos données autant bien sur notre mobile que sur notre PC.

1.4 Sécurité

Étant donné la réticence actuelle des utilisateurs face aux gestionnaires, il est légitime de se poser la question si ces derniers ont une bonne et forte implémentation de la sécurité, ainsi c'est la question que nous tenteront de répondre durant ce rapport. Toutefois, nous pouvons présenter quelques solutions de sécurité entreprises par les constructeurs.

Au niveau du chiffrement, même si le gestionnaire utilise le cloud pour la synchronisation des données, tout est chiffré en local sur le device de l'utilisateur. Ainsi, toutes les données transmises dans le cloud sont chiffrées et protégées. Cela permet de garder des données chiffrées sur les serveurs des constructeurs, afin que ces derniers n'aient **aucune connaissance** des données claires.

Étant donné que le master password n'est jamais envoyé sur les serveurs des constructeurs afin qu'ils n'aient aucune connaissance des données, certains gestionnaires permettent de récupérer leur coffre-fort si l'utilisateur perd son master password. Ils créent une clé de récupération (key recovery) qui leur est transmise dès le moment où ils activent l'option de récupération.

Des fonctionnalités concernant la sécurité des mots de passe sont également proposées ; avec un abonnement payant, il est parfois possible d'avoir une surveillance active de la fuite ou de la compromission de nos propres données sur des services. Certaines applications activent des alertes qui préviennent l'utilisateur en cas de compromission.

Chapitre 2

Étude de marché

Ce chapitre vise à étudier les différentes fonctionnalités offertes par les gestionnaires de mots de passe en les comparant entre plusieurs produits sélectionnés et en établissant un tableau afin d'avoir une meilleure vue d'ensemble.

Nous allons également analyser les différents prix des applications ainsi que présenter où en est le marché actuel afin d'étudier la popularité de ces dernières.

Pour l'étude de marché, les gestionnaires de mots de passe sélectionnés sont : *LastPass*¹, *Dashlane*², *1Password*³, *KeePass*⁴, *Bitwarden*⁵, *NordPass*⁶, *Padloc*⁷, *Keeper*⁸, *Firefox*⁹.

Ils ont été sélectionnés en se basant sur leur popularité sur le marché ainsi qu'à la suite de lecture d'articles concernant les meilleurs gestionnaires de mots de passe [9][14][28][24]. Les applications open-source ont été avantagées lors de leurs sélections.

2.1 Fonctionnalités

Ci-après, nous avons établi un tableau récapitulatif qui indique quels gestionnaires de mots de passe offre quelles fonctionnalités. Nous nous sommes basés sur les fonctionnalités citées dans le chapitre d'introduction 1.3.

-
1. <https://www.lastpass.com/>
 2. <https://www.dashlane.com/>
 3. <https://1password.com/>
 4. <https://keepass.info/>
 5. <https://bitwarden.com/>
 6. <https://nordpass.com/>
 7. <https://padloc.app/>
 8. <https://www.keepersecurity.com/>
 9. <https://www.mozilla.org/fr/firefox/features/password-manager/>

Application	Stockage d'informations personnelles (cartes de crédit, passeport, contrats, etc.)	Remplissage automatique des formulaires en ligne (auto-complétion)	Partage de données entre plusieurs utilisateurs (par exemple, partage d'informations d'identifications entre une famille)	Générateur de mots de passe forts	Surveillance de la fuite de données ou données compromises	Alerte en cas de données compromises	Synchronisation de données (cloud)	Authentification de données entre devices	Self-hosting	Support prioritaire	Connexion à l'aide de facteurs biométriques (fingerprint ou facial recognition) ou d'un pin	Backup & Restore, possibilité de récupérer une ancienne sauvegarde	Open-source
LastPass	●	●	⋯	○	○	○	●	⋯	○	○			
Dashlane	○	●	⋯	●	○	○	○	⋯				●	●
1Password	○	○	○	○		○	○	○		○		●	●
KeePass	●	●		●	●	○		●	●		●	●	●
Bitwarden	○	●	⋯	●	○	○	●	⋯	○	○	⋯	●	●
NordPass	●	●	○	●	○		●	●	○	○	●		
Padloc	●	●	○	●	○		●	○	●	○	●		●
Keeper	○	○	○	○	○	○	○	○	○	○	○	○	
Firefox		●		●	●	●	●	●					●

● L'application propose cette fonctionnalité
 ○ Fonctionnalité proposée mais avec une version premium (payante)
 ⋯ Limitations avec une version gratuite

TABLE 2.1 – Fonctionnalités des candidats sélectionnés

Sur tous les candidats sélectionnés, nous remarquons que la plupart offrent la majorité des fonctionnalités énumérées plus haut. Nous constatons que l'offre des constructeurs de gestionnaires de mots de passe est assez variée et répond à la demande des particuliers et des entreprises.

Pour KeePass, en général, il est nécessaire d'installer des plugins supplémentaires afin de profiter de toutes les fonctionnalités.

Par rapport aux restaurations de sauvegarde (*Backup & Recovery*), les gestionnaires sync-cloud vont automatiquement créer des backups sur les serveurs du constructeur toutes les nuits, donc la restauration se fait directement dans le gestionnaire. Pour Bitwarden, lorsque le gestionnaire est hébergé on-premise¹⁰, il est nécessaire de créer ses propres procédures de sauvegardes. Etant donné que KeePass est uniquement en local, les sauvegardes doivent

10. Gestionnaire de mots de passe hébergé sur les serveurs de l'utilisateur, souvent utilisé dans des infrastructures d'entreprise

être faites manuellement et peuvent être importées sur l'application. Toutes ces solutions nécessitent le master password. Si ce dernier est oublié, il existe plusieurs solutions différentes en fonction des constructeurs (fonctionnalité pas disponible sur KeePass).

La "sauvegarde" qui ne nécessite pas d'avoir son master password est l'exportation des données en fichier CSV, mais à moins de chiffrer le fichier, les données sont en claires ce qui n'est évidemment pas sécurisé et pas très recommandé.

2.2 Plateformes

Cette partie va permettre de visualiser sur quelles plateformes les gestionnaires de mots de passe sélectionnés sont supportés.

Application	Windows	MacOS	Linux	Android	iOS	Navigateur
LastPass	•	•	•	•	•	•
Dashlane						•
1Password	•	•	•	•	•	
KeePass	•	○	○	○	○	○
Bitwarden	•	•	•	•	•	•
NordPass	•	•	•	•	•	
Padloc	•	•	•	•	•	•
Keeper	•	•	•	•	•	•
Firefox						•

TABLE 2.2 – Plateformes supportées par les différentes applications

- L'application est supportée sur ces plateformes
- Des applications (ou des paquets) compatibles avec KeePass Password Safe non-officielles mais contribuées existent

Même si un gestionnaire supporte toutes les plateformes indiquées, il est nécessaire d'aller vérifier les conditions d'utilisation du système, c'est-à-dire les versions des plateformes afin de s'assurer que l'application fonctionnera quand même.

Cependant, nous constatons que la majorité des applications sont disponibles sur les plateformes les plus courantes, et même si elles ne le sont pas, il y a souvent une solution non-officielle (notamment pour KeePass) ou via le navigateur qui existe.

2.3 Prix

Nous allons passer brièvement en revue les prix proposés par les gestionnaires de mots de passe. Chaque application propose leurs propres gammes de prix avec également des abonnements possibles pour les particuliers, familles ou entreprises.

2.3.1 Particuliers

Pour la plupart des applications, nous pouvons retrouver 3 gammes de prix ; Gratuit, Premium, Famille. L'offre familiale va être plus chère car les gestionnaires de mots de passe sont conçus pour pouvoir avoir plusieurs gestionnaires chiffrés individuels.

Les tarifs ci-dessous sont exprimés en mensualités et en USD.

Application	Gratuit	Premium	Famille
LastPass	\$0	\$3	\$4
Dashlane	\$0	\$3.99	\$5.99
1Password	non	\$2.99	\$4.99
KeePass	\$0	non	non
Bitwarden	\$0	< \$1	\$3.33
NordPass	\$0	\$1.84	\$4.99
Padloc	\$0	\$3.49	\$5.95
Keeper	non	\$2.92	\$6.25
Firefox	non	non	non

TABLE 2.3 – Tarifs pour particuliers

2.3.2 Entreprises

Les entreprises ont quant à elle des prix différents dû à leurs besoins spécifiques où ils pourraient avoir besoin d'un devis personnel afin de choisir l'abonnement qui convient au mieux à leur infrastructure. Il existe plusieurs catégories qui sont en fonction du nombre d'employés et également par rapport aux fonctionnalités souhaitées.

Chaque prix est indiqué en mensualités, en USD et par employé.

Application	Team	Business
LastPass	\$4	\$6
Dashlane	\$5	\$8
1Password	non	\$7.99

KeePass	non	non
Bitwarden	\$3	\$5
NordPass	non	\$3.50 ¹¹
Padloc	\$3.49	\$6.99 ¹²
Keeper	non	\$3.75 ¹³
Firefox	non	non

TABLE 2.4 – Tarifs pour les entreprises

2.4 Récapitulatif de l'étude

Nous allons résumer toutes les informations que nous avons recueillies dans ce chapitre-ci ; au final, nous constatons que les gestionnaires de mots de passe qui sont actuellement sur le marché (ici les plus populaires), sont assez complets au niveau des fonctionnalités proposées et ils sont adaptées pour tout type d'utilisation (personnelle, familiale ou professionnelle). Pour les gestionnaires payants, leurs prix sont assez abordables pour l'offre qu'ils proposent. Toutefois, même les gestionnaires en version gratuite, convient tout à fait à une utilisation quotidienne.

Au niveau des applications comparées, toutes ont leurs points positifs et leurs points négatifs (l'aspect sécuritaire sera discuté dans le chapitre *étude sécuritaire*).

LastPass propose une version gratuite avec les fonctionnalités classiques que l'on attend d'un gestionnaire de mot de passe. La limite est que l'application n'est accessible que depuis un seul type d'appareil, ils font la différence entre ordinateur (fixe et portable) et appareil mobile (téléphone, montre, tablette). La version payante offre le MFA ainsi qu'un dashboard (avec les alertes de sécurité et la surveillance sur les données compromises), ce qui est une fonctionnalité intéressante.

La version gratuite de **DashLane** propose un stockage jusqu'à 50 mots de passe, ce qui est au final assez limité mais il offre le 2FA ainsi que le partage sécurisé (jusqu'à 5 comptes). La version premium, permet l'utilisation d'un VPN ainsi qu'une synchronisation sur plusieurs appareils.

1Password est totalement payant mais est l'un des gestionnaires de mots de passe le plus populaire sur le marché.

KeePass est une application gratuite et open-source. Il propose une grande sélection de plugins assez utiles et variés, ce qui permet une grande offre, en plus d'être complètement

11. Il y a la possibilité d'établir un devis en fonction des besoins spécifiques de l'entreprise

12. voir 11

13. voir 11

gratuite.

bitwarden propose une version gratuite étonnement très complète avec un stockage illimité de mots de passe et un nombre illimité d'appareils. La version premium offre un 2FA avancé (notamment la connexion à l'aide d'une Yubikey) ainsi que des rapports de sécurité.

NordPass propose également une version gratuite complète qui permet le MFA ou encore la synchronisation entre plusieurs appareils, ce qui est très utile. Le premium propose l'aspect sécuritaire en plus. Cependant, c'est la solution gratuite la meilleure de tous les candidats sélectionnés.

Padloc n'est pas un gestionnaire très populaire mais il l'avantage d'être open-source et d'être disponible sur Github¹⁴. La version gratuite n'offre pas beaucoup de fonctionnalités mais il y a la possibilité de stocker un nombre illimité de secrets et d'y connecter un nombre illimité d'appareils. De plus, il est multi-plateformes.

Keeper est complètement payant mais a une offre très complète et est particulièrement bien adapté pour les entreprises.

Finalement, le gestionnaire de mots de passe proposé par **Firefox** est directement inclus avec le navigateur, ainsi ses fonctionnalités proposées sont assez basiques et pas très poussées, mais il propose les fonctionnalités attendues d'un gestionnaire, c'est-à-dire enregistrement, génération et synchronisation de mots de passe.

14. <https://github.com/padloc/padloc>

Chapitre 3

Informations préliminaires

Avant d'effectuer l'étude sécuritaire des gestionnaires de mots de passe, nous allons dédier un chapitre bref à l'explication de quelques notions de sécurité à propos des mots de passe.

3.1 Attaques sur mot de passe

Nous pouvons premièrement nous concentrer sur les différentes attaques qu'il est possible de faire sur des mots de passe afin de motiver l'utilisation d'un unique mot de passe par service et d'en créer un fort.

Une attaque populaire est le **brute force**. Cette dernière vise à tester x mots de passe différents avant de trouver le bon.

L'attaquant va tester systématiquement toutes les combinaisons possibles avec des lettres, chiffre et symboles. Le temps que va prendre l'attaque dépend de la complexité du mot de passe, c'est-à-dire le nombre de caractères. Plus le mot de passe est complexe, plus cela prendra du temps pour le brute forcer. Le nombre de temps de craquage peut aller de 2 secondes à des milliers d'années. Un outil connu pour cette attaque est Hashcat par exemple.

Une attaque également populaire est l'**attaque par dictionnaire**. Elle consiste à tester systématiquement chaque mot d'un dictionnaire en un mot de passe. L'attaquant sélectionne un fichier avec des milliers de mots du dictionnaire communs ou de mots populaires (comme par exemple des célébrités) et tente de trouver le mot de passe en testant chaque mot.

Cette attaque peut réussir si l'utilisateur utilise des mots communs comme par exemple "password" ou "iloveyou".

Comme attaque, nous pouvons également citer les **attaques *Rainbow Tables***. Cette dernière permet de craquer les hashes des mots de passe qui sont stockés dans la base de données. Certaines applications ne stockent pas les mots de passe en clair mais à la place, elle les stocke

en les chiffrant avec des hashes. Ainsi, la rainbow table fait référence à une table pré-calculée qui contient la valeur du hash du mot de passe pour chaque caractère en clair.

Cependant, dans les architectures modernes des applications, cette attaque n'est plus très efficace car la majorité utilisent un sel, qui est une valeur aléatoire, afin de stocker plus sûrement des mots de passe dans une base de données.

Finalement, nous pouvons citer l'attaque **Man-in-the-Middle** (MITM) qui pourrait permettre à un attaquant de se mettre entre l'utilisateur et le serveur et d'intercepter le mot de passe via le trafic.

3.2 Complexités

En prenant en compte les attaques sur mots de passe existantes, il est ainsi important de prendre des mesures afin d'éviter au maximum ces attaques. La mesure la plus importante est d'avoir un mot de passe complexe. Cela ne va pas stopper à 100% les attaques mais cela va venir les freiner, car plus le mot de passe est complexe et moins prévisible, plus l'attaque prendra du temps. En définition, la complexité d'un mot de passe est en fonction du nombre de types de caractères différents (chiffres, symboles, lettres), du nombre de caractères et de la variation des lettres (majuscule, minuscule). Afin de mettre en place de la complexité dans les mots de passe, il est nécessaire d'implémenter dans les applications des critères strictes lors de la création de mots de passe.

En se référant à l'étude de Hive Systems en 2022 [43], nous pouvons voir en fonction des critères de complexité combien de temps cela prend pour le brute forcer :

Nombre de caractères	Nombres	Minuscules	Majuscules	Chiffres, Minuscules, Majuscules	Chiffres, Minuscules, Majuscules, Symboles
4	Instantanément	Instantanément	Instantanément	Instantanément	Instantanément
5	Instantanément	Instantanément	Instantanément	Instantanément	Instantanément
6	Instantanément	Instantanément	Instantanément	Instantanément	Instantanément
7	Instantanément	Instantanément	2s	7s	31s
8	Instantanément	Instantanément	2min	7min	39min
9	Instantanément	10s	1h	7h	2 jours
10	Instantanément	4min	3 jours	3 semaines	5 mois
11	Instantanément	2h	5 mois	3 ans	34 ans
12	2s	2 jours	24 ans	200 ans	3'000 ans
13	19s	2 mois	1'000 ans	12'000 ans	202'000 ans
14	3min	4 ans	64'000 ans	750'000 ans	16 millions ans
15	32min	100 ans	3 millions ans	46 millions ans	1 billions ans
16	5h	3'000 ans	173 millions ans	3 billions ans	92 billions ans
17	2 jours	69'000 ans	9 billions ans	179 billions ans	7 trillions ans
18	3 semaines	2 millions ans	467 billions ans	11 trillions années	438 trillions ans

TABLE 3.1 – Temps pour craquer les mots de passe en fonction des critères

Ainsi, nous pouvons constater que la norme de 8 caractères, voire 6, qui est appliquée sur beaucoup d'applications en 2022 n'est plus assez résistante aux attaques de brute force. Les critères sont acceptables à partir des cellules en orange.

3.2.1 Entropie

D'après Okta[33], "L'entropie du mot de passe est une mesure de l'imprévisibilité, et donc de l'impossibilité de deviner, d'un mot de passe.". L'entropie mesure la probabilité que des attaques sur mots de passe (brute-force par exemple) fonctionnent et que l'attaquant puisse entrer dans le service. Elle permet ainsi d'évaluer sa force et sa complexité.

Il existe une formule qui permet de calculer l'entropie d'un mot de passe. Plus l'entropie est grande, plus le mot de passe est fort et imprévisible.

$$E = \log_2(R^L)$$

- R : le nombre possibles de caractères dans le mot de passe, par exemple pour les chiffres, ce serait 10 ([0-9])
- L : le nombre de caractère dans le mot de passe

Ainsi, en se basant sur le tableau 3.1 et sur une échelle proposée sur ce site [30], nous pouvons définir l'échelle suivante en fonction du nombre de bits d'entropie :

1. < 25 : mot de passe très faible
2. < 50 : mot de passe faible
3. < 75 : mot de passe raisonnable
4. < 100 : mot de passe fort
5. >= 100 : mot de passe très fort

3.3 2FA

Un autre point intéressant à aborder est l'authentification à double facteurs (2FA) ou l'authentification à multi-facteurs (MFA) qui permet d'ajouter une couche sécuritaire supplémentaire au mot de passe. Ces derniers consistent à ajouter un facteur supplémentaire en plus d'un simple mot de passe lors de la connexion d'un utilisateur à un service. Une liste non-exhaustive de quelques solutions disponibles :

- Facteurs biométriques (empreintes digitales, reconnaissance faciale)
- Code par SMS ou e-mail
- Tokens (YubiKey)
- One-time password (OTP)

Chapitre 4

Étude sécuritaire

Ce chapitre est dédié à toute l'analyse sécuritaire des gestionnaires de mots de passes en général. Nous allons dans un premier temps décrire comment ces applications sont sécurisées en fonction de leur type, puis présenter les différents algorithmes utilisés dans les candidats sélectionnés

4.1 Implémentation de la sécurité dans les gestionnaires de mots de passe

Dans cette section, afin de se baser sur des gestionnaires de mots de passe déjà existants et de pouvoir comparer les différentes sécurités implémentées, nous allons reprendre les 9 candidats sélectionnés dans la partie *étude de marché*, c'est-à-dire ; *LastPass*, *Dashlane*, *1Password*, *KeePass*, *Bitwarden*, *NordPass*, *Padloc*, *Keeper* et *Firefox*. Toutes les informations citées sont basées sur les *security whitepapers* des constructeurs[25][12][1][22][5][36][23][31].

Les gestionnaires de mots de passe sélectionnés ont un fonctionnement plutôt similaire, au final cette méthode est plutôt classique dans les architectures des applications. Un *master password* (qui est seulement connu par l'utilisateur) est généré ou entré par l'utilisateur et va permettre le déverrouillage de l'application et le chiffrement / déchiffrement en local de toutes les données stockées.

À part pour les gestionnaires en local qui gèrent la sécurité différemment, les gestionnaires qui utilisent la synchronisation avec le cloud mettent en avant le *Zero-knowledge encryption*. C'est une méthode qui va permettre un chiffrement *end-to-end* et qui va sécuriser au mieux les données personnelles et sensibles des utilisateurs, des serveurs du constructeur. En sachant que toutes les données sont stockées dans le cloud du provider, afin d'éviter que n'importe qui puisse y avoir accès, toutes les données sont chiffrées avant d'être envoyées au serveur. Ainsi, les données sont uniquement déchiffrées sur le device de l'utilisateur et la clé de chiffrement

reste également en local.

Nous allons décrire dans les sous-sections suivantes comment la sécurité est implémentée dans les gestionnaires de mots de passe en fonction de leur type afin d'aller un peu plus en détail. Nous nous concentrons sur 3 types différents de gestionnaires de mots de passe :

1. **Browser-based**, qui sont des gestionnaires de mots de passe directement intégré dans le navigateur (Google Chrome, Firefox, etc.)
2. **Cloud/local-based**, qui fonctionnent en local et avec le cloud et peuvent être multi-plateformes. Ce sont des applications tierces.
3. **Local-based**, qui fonctionnent uniquement en local et qui n'ont aucune donnée synchronisée entre plusieurs devices.

4.1.1 Les gestionnaires browser-based

Les gestionnaires de mots de passe *browser-based* proposent des fonctionnalités classiques et ne sont pas très poussés. Ce sont des applications légères qui sont pensées pour faciliter au mieux les internautes.[46]

Par défaut, ces applications n'activent pas de *master password* et certaines n'en proposent même pas pour ajouter un chiffrement supplémentaire. Toutefois, ils proposent la fonctionnalité de 2FA. Afin d'illustrer au mieux nos propos, nous allons nous baser sur le gestionnaire intégré Firefox.

Firefox fonctionne localement ou avec la synchronisation qui demande l'utilisation du cloud. Si la synchronisation n'est pas activée, les mots de passes stockés sont directement chiffrés sur l'appareil et sont ajoutés à un fichier *logins.json* qui se trouve dans le répertoire de l'utilisateur.

Il propose également la fonctionnalité d'ajouter un master password afin d'avoir une couche sécuritaire supplémentaire (par défaut, cette fonction est désactivée), ainsi, sans un mot de passe principal et sans l'activation du 2FA, les mots de passe sont accessibles en clair sur le navigateur dès le moment où on a accès à l'appareil et que l'utilisateur est connecté à son compte (ce qui est fortement probable, car le compte reste connecté, même après une fin de session). Toutefois, lors de l'ajout d'un master password, ce dernier est uniquement défini en local et n'est pas synchronisé entre profils ou appareils, mais il chiffrera toutes les données en local.

Au niveau du fonctionnement[2] (sans master password), les identifiants sont chiffrés à l'aide de 3DES-CBC et sont directement stockés dans un fichier JSON *logins.json* encodés en ASN.1 puis en Base64. La clé de chiffrement est stockée dans une base de donnée *key4.db*. Il existe actuellement des outils pour déchiffrer les mots de passe du gestionnaire.

Nous allons illustrer ci-dessous, comment se passe le déchiffrement d'un mot de passe d'une

entrée stockée en local dans le gestionnaire de mots de passe Firefox. La référence de ce schéma est un GitHub[8] qui explique comment sont protégés les mots de passe.

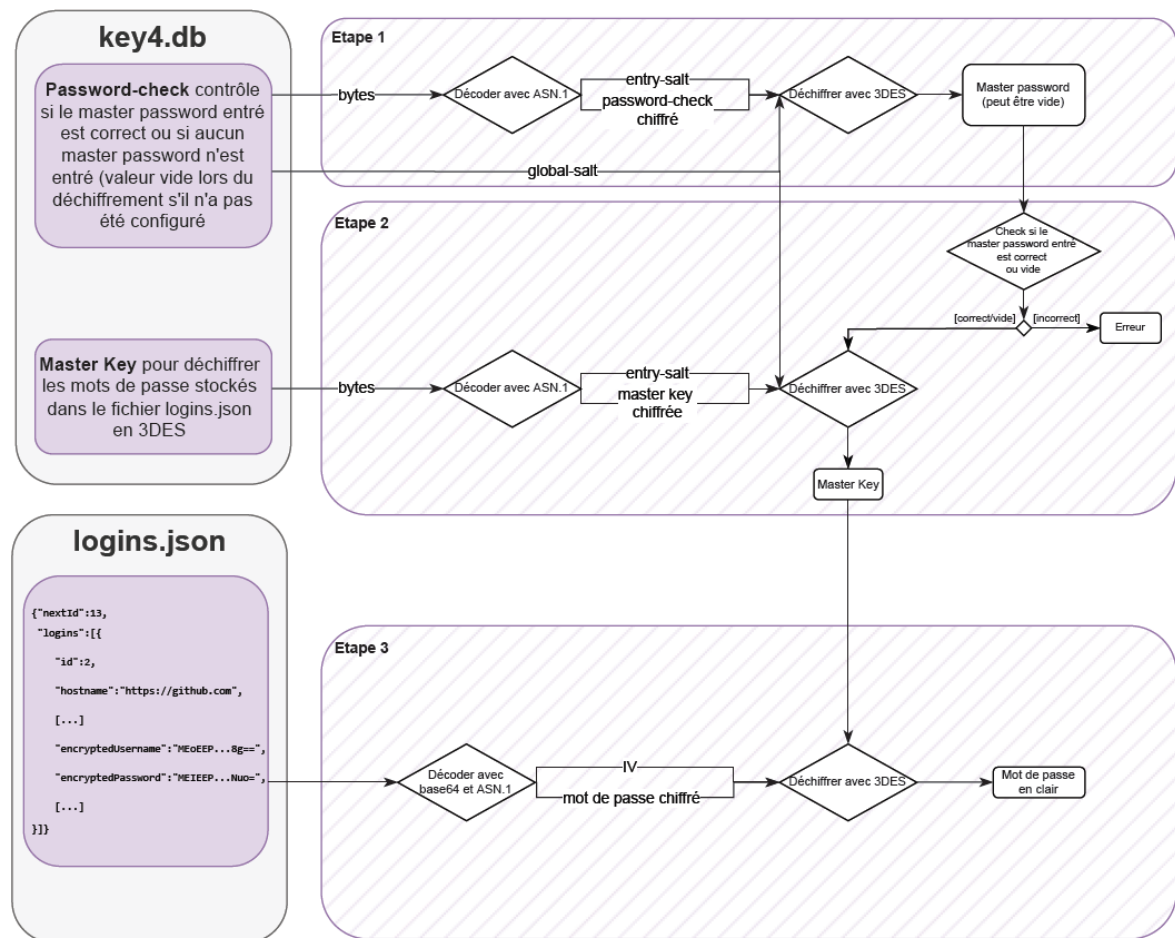


FIGURE 4.1 – Déchiffrement d'un mot de passe en local sur Firefox

Si l'option de synchronisation (*Firefox Sync*) est activée[41], les données stockées sur les serveurs Mozilla seront chiffrées. Nous pouvons nous baser sur le schéma suivant qui explique le processus de synchronisation des données avec les serveurs Mozilla :

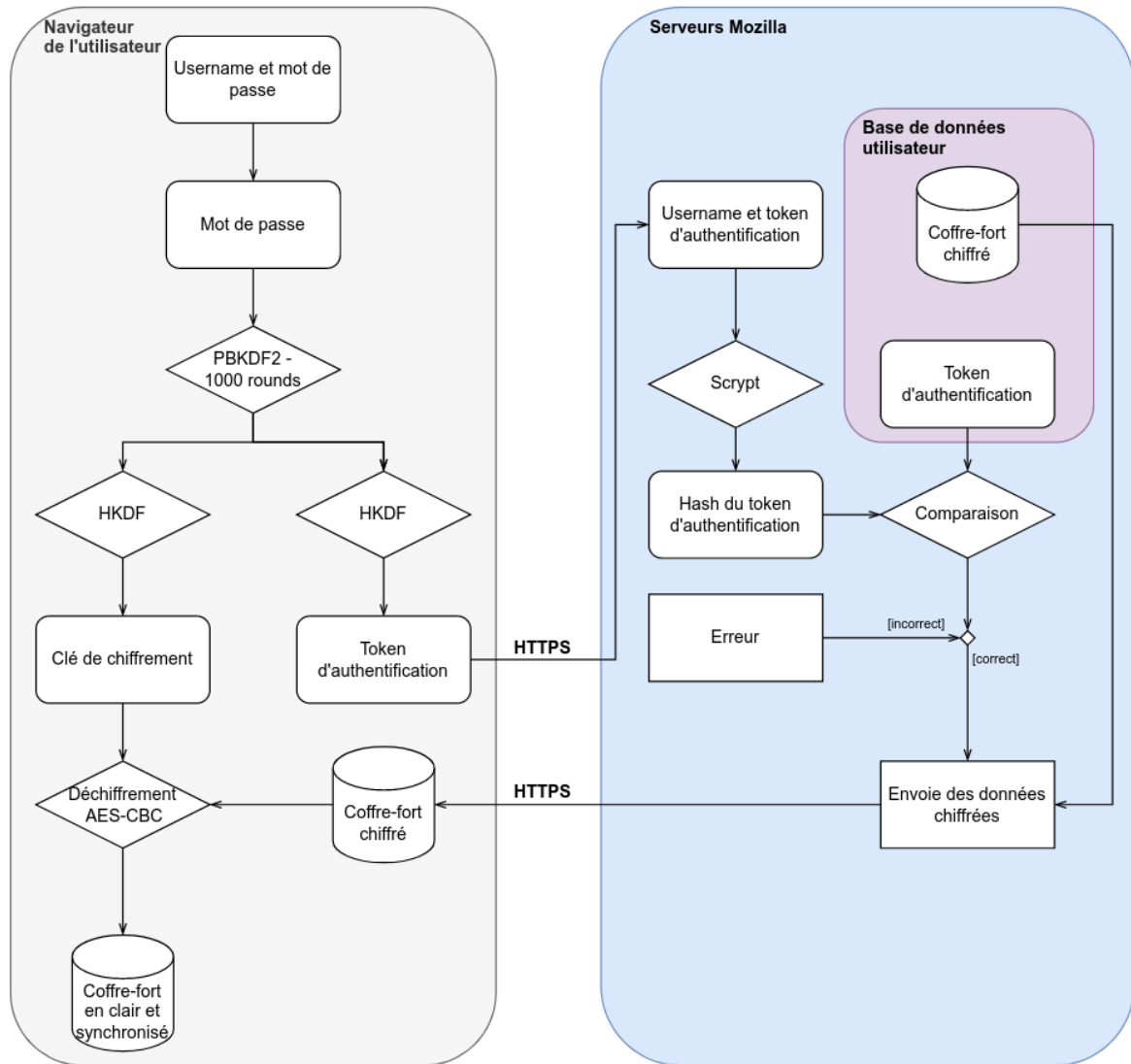


FIGURE 4.2 – Schéma de synchronisation de données sur Firefox

Les données chiffrées sont protégées avec HMAC-SHA256 afin de pouvoir authentifier les données avant de les déchiffrer. La méthode utilisée est Encrypt-then-MAC. Cela permet de protéger l'intégrité de toutes les données chiffrées.

Comme cité précédemment, lors d'un ajout de master password afin de protéger les données localement, Firefox va tout d'abord chiffrer les données localement (le stockage se fait comme expliqué plus haut) et va les re-chiffrer lors de la synchronisation avec le processus expliqué à l'aide du schéma.

Lorsque les serveurs Mozilla envoient les données à l'utilisateur pour la synchronisation, il envoie toutes les données stockées (tous les identifiants enregistrés), même si qu'une modification n'a été effectuée.

Au final, ce design cryptographique est plutôt classique au sein des gestionnaires de mots de passe qui utilisent les serveurs du constructeur pour stocker les données.

Les autres gestionnaires de mots de passe browser-based fonctionnent en général de la même manière avec les données synchronisées. Chrome stocke également les identifiants dans un fichier local protégé avec DPAPI (Microsoft's Data Protection API).

4.1.2 Les gestionnaires local-based

Pour les gestionnaires qui fonctionnent uniquement en local, nous allons nous baser sur l'implémentation sécuritaire de **KeePass**, qui est open-source, ce qui peut faciliter à comprendre tout le concept. Dans cette section, nous n'allons pas détailler toute l'architecture mais rester plutôt en surface. D'autres gestionnaires de mots de passe fonctionnent également en *offline* avec un stockage local, cependant lors de la première connexion, il y a quand même des informations envoyées aux serveurs du constructeur (comme 1Password par exemple), ce qui n'est pas le cas pour l'application de base KeePass.

Etant donné que KeePass ne fonctionne qu'en local, il est important de bien gérer la mémoire et le stockage de données sensibles.

Toutes les données se trouvent dans un fichier de base de données spécial *.kdbx*. Ce fichier contient toutes les données du gestionnaire (mots de passe, usernames, etc.) et est chiffré (également compressé en GZIP si on le souhaite). Dans la version de KeePass 2.x, les chiffrements supportés sont AES256-CBC et ChaCha20.

La base de données est stockée où l'utilisateur le souhaite (avec une possibilité de la stocker dans un cloud), c'est pourquoi la sécurité repose sur la complexité du mot de passe. Elle est structurée avec un header et un contenu[26][18].

Dans le header, sont stockées différentes informations ; un UUID indiquant le cipher, une indication si le fichier est compressé, différentes seed (voir 4.4), l'IV pour le chiffrement et des bytes pour l'authentification (générés aléatoirement lors de la sauvegarde de la base de données). Le corps du fichier contient des blocs hachés avec HMAC-SHA256 et des blocs de données qui ont un format XML lorsqu'ils sont déchiffrés.

Afin d'expliquer au mieux la structure de la base de données, nous avons établi un schéma qui présente tous les champs du header et du contenu de la base de données.

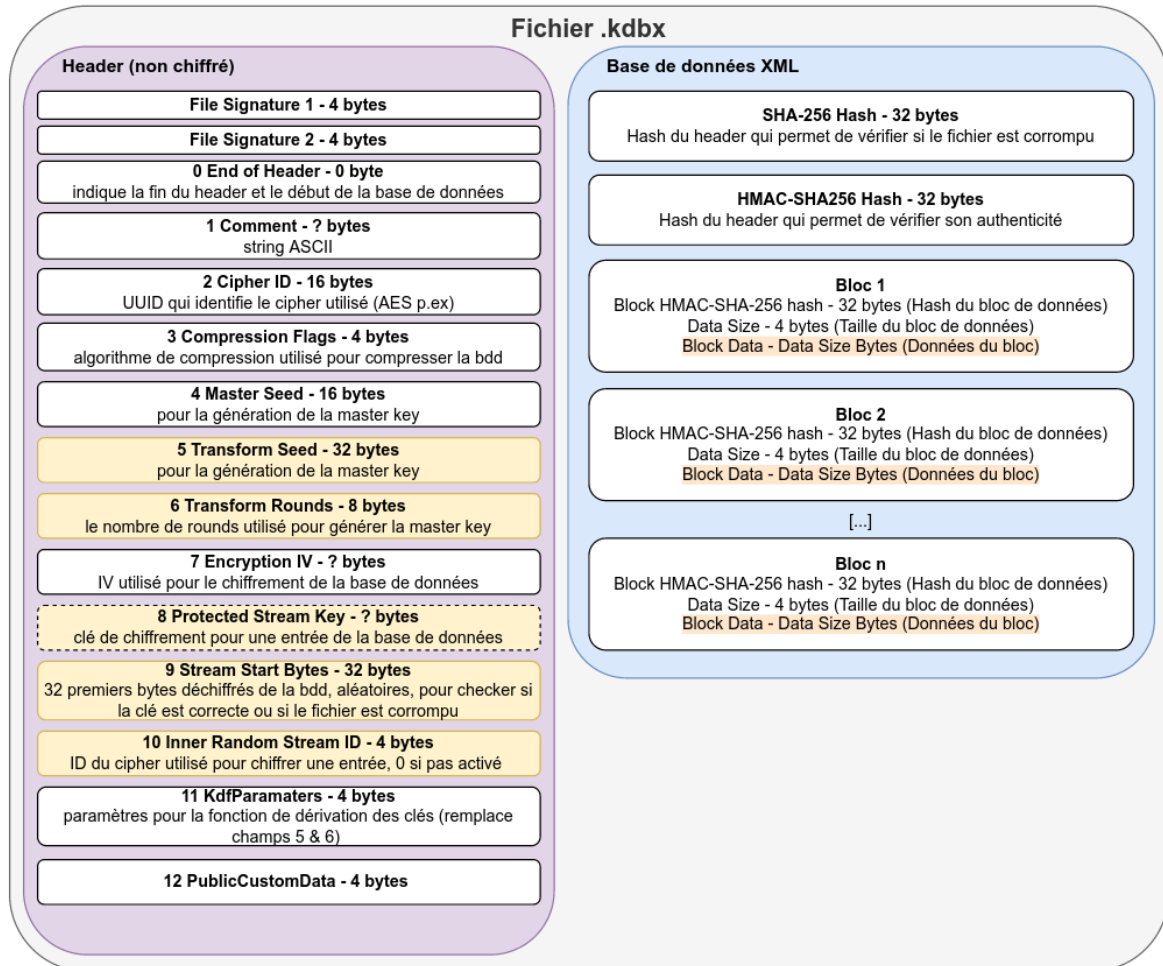


FIGURE 4.3 – Contenu du fichier .kdbx de KeePass

Nous avons représenté la structure de la version 4 du fichier *.kdbx*[21]. Le header contient des champs qui ne sont pas forcément présents dans chaque fichier (en traitillé sur le schéma). Ces derniers apparaissent si on souhaite effectuer un double chiffrement et chiffrer en plus chaque entrée de la base de données (chaque mot de passe stocké). De plus, les champs en jaune permettent la comptabilité entre la version 3.1 et 4 des fichiers *.kdbx*, cependant ils ne sont plus utilisés dans la version récente.

La partie données du fichier contient deux champs hashés afin de vérifier l'authenticité du header. Ces champs ne sont pas chiffrés afin qu'on ne doive pas déchiffrer toutes les données pour vérifier que le master password entré est correct. Toutes les données sont séparées en blocs de données et chaque bloc possède un hash (non-chiffré) afin de contrôler l'authenticité du bloc. Le schéma *Encrypt-Then-Mac* est utilisé, c'est-à-dire que le hash est généré à l'aide

de données chiffrées, qui sont surlignées en orange sur le schéma ci-dessus.

Au niveau de l'utilisation de la mémoire du processus, lorsqu'il y a une interaction avec une entrée de la base de données, elle est chargée en mémoire. La protection de la mémoire s'applique uniquement aux données sensibles telles que la Master key et des mots de passe, le username ou les notes. Pour certaines opérations, comme la copie dans le presse-papier ou l'affichage des données en clair, KeePass doit rendre les données sensibles disponibles de manière non chiffrée dans la mémoire du processus pendant un court instant.

Dès le moment où l'application est verrouillée ou le processus arrêté, la mémoire est nettoyée afin qu'elle ne contienne aucune information sensible.

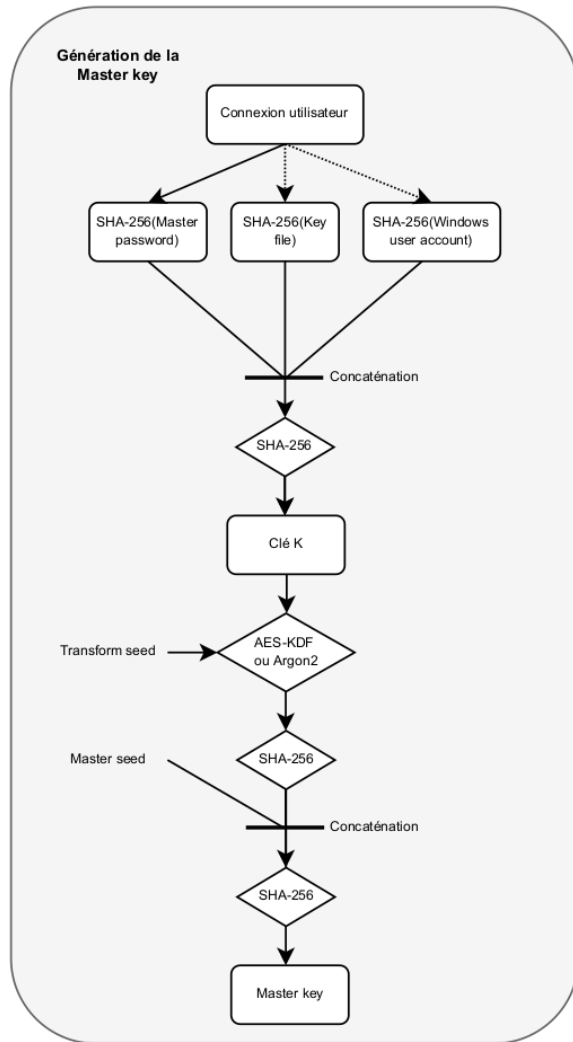


FIGURE 4.4 – Génération de la Master key sur Kee-pass

La Master key sur KeePass prend plusieurs arguments différents et est générée d'une manière assez complexe. L'utilisateur doit obligatoirement fournir un master password, et peut également utiliser un key file et / ou le compte utilisateur Windows.

Chaque composant est haché avec SHA-256 et sont concaténés et hachés ensemble. En sortie, on a une clé composite K qui va être dérivée à l'aide de AES-KDF ou Argon2. Les paramètres de ces deux algorithmes peuvent être configurés dans les paramètres de la base de données.

Chaque sortie est hachée avec SHA-256 et au final, on obtient la Master key qui permettra de déchiffrer la base de données.

Les deux différentes seeds utilisées dans la génération de la clé sont stockées dans le header de la base de données *.kdbx*.

Cette architecture permet de se protéger contre les attaques par dictionnaires et le brute-force de la Master key.

4.1.3 Les gestionnaires cloud/local-based

Les gestionnaires de mots de passes cloud/local-based ont tous une architecture similaire dû au fait que toutes les données sont stockées sur les serveurs du constructeur. Il y a des différences avec l'authentification de l'utilisateur et des algorithmes cryptographiques choisis, mais le design sécuritaire a la même base au niveau de la connexion de l'utilisateur et du chiffrement du gestionnaire.

Ainsi, afin d'expliquer un peu plus en détail le fonctionnement d'un gestionnaire cloud-based, nous allons prendre l'exemple de **LastPass**. Nous allons nous baser sur le schéma ci-dessous afin d'appuyer nos propos.

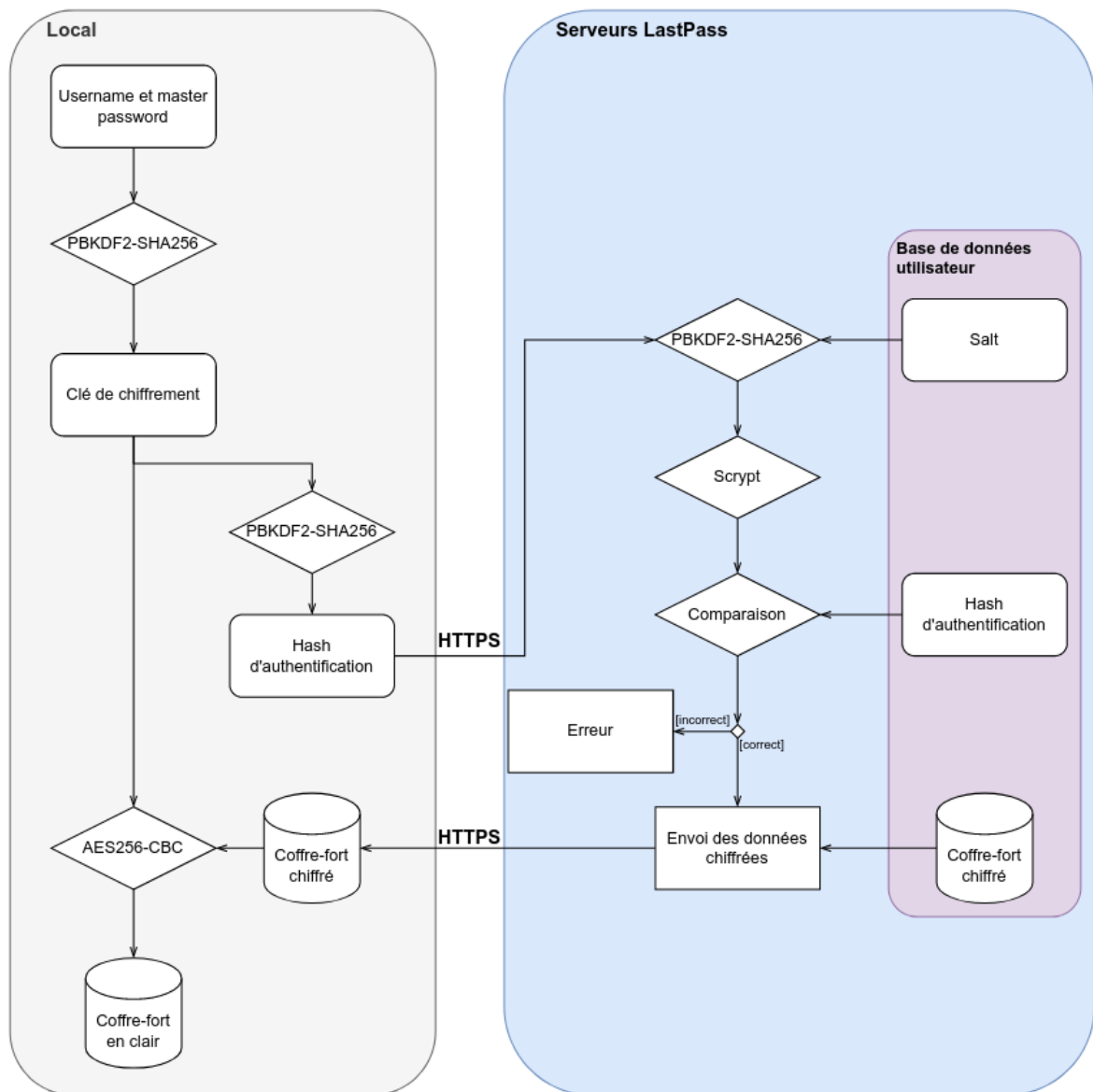


FIGURE 4.5 – Schéma de déchiffrement de LastPass

Afin de générer la clé de chiffrement, ce dernier dérive le master password en utilisant le nom d'utilisateur comme sel. Il utilise l'algorithme PBKDF2-SHA256 avec 100'000 rounds.

Du côté client, la sortie est hashée une fois de plus et est envoyée aux serveurs du provider afin de comparer ce hash d'authentification à celui qui est dans la base de données.

Toutes les données de l'utilisateur sont chiffrées avec AES-256-CBC avec la clé de chiffrement générée depuis le master password.

Les serveurs de LastPass stockeront la hash d'authentification et le coffre-fort de l'utilisateur chiffré. La clé de chiffrement reste sur le device en local dans la mémoire du processus.

Tous les gestionnaires de mots de passe incluent la fonctionnalité de 2FA (voire MFA). La façon dont est géré cette sécurité additionnelle dépend du constructeur et de la méthode utilisée, certains stockent une clé supplémentaire sur les serveurs, d'autres stockent en local les informations (notamment avec les facteurs biométriques). Lors de la génération de la clé de chiffrement, le master password et les facteurs supplémentaires sont combinés pour y dériver la clé.

4.2 Partage d'informations

La majorité des gestionnaires de mots de passe proposent le partage d'informations entre plusieurs utilisateurs. Nous n'allons pas entrer dans tous les détails dans cette sous-section afin de rester bref et d'expliquer le schéma général du partage de données.

Pour cette fonctionnalité, nous utilisons la cryptographie asymétrique avec RSA. Dès l'inscription et la création du coffre-fort de l'utilisateur, une paire de clés de 2048 bits [publique, privée] est générée.

La façon dont est géré le stockage des clés dépend du constructeur mais en général, la clé publique est envoyée aux serveurs, tandis que la clé privée est soit stockée avec les données personnelles de l'utilisateur, soit envoyée aux serveurs. Cette dernière (privée au serveur) est chiffrée afin de garantir sa protection et étant donné qu'elle est personnelle et n'est pas censé être transmise entre utilisateurs, cela est tout à fait concevable. Pour le chiffrement, soit la même clé pour le chiffrement des données est utilisée ou une clé symétrique supplémentaire est générée.

Afin d'illustrer concrètement le processus complet de partage, nous allons nous baser sur le gestionnaire **Dashlane** et nous montrerons un exemple où Alice souhaiterait partager un identifiant à un autre utilisateur, Bob.

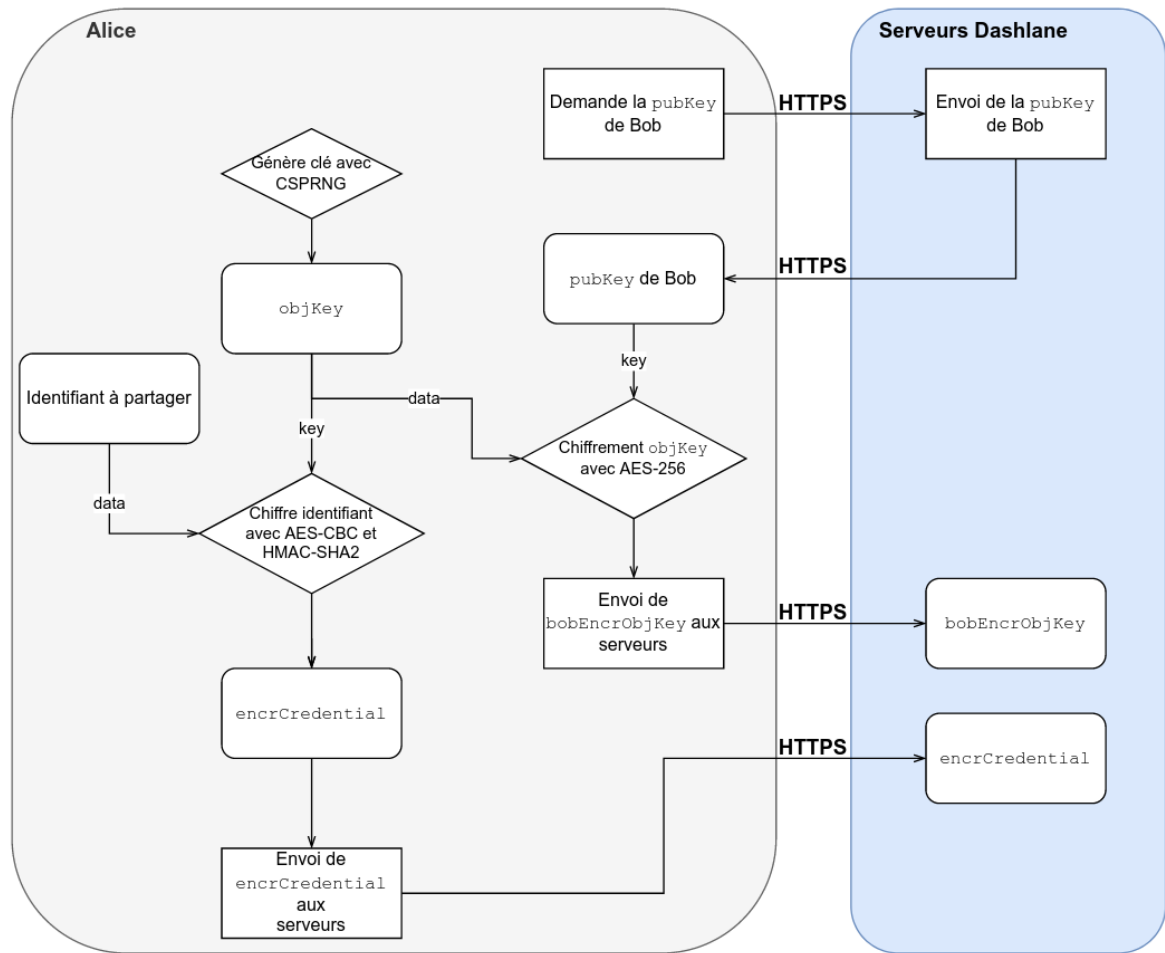


FIGURE 4.6 – Partage d'un identifiant sur Dashlane

Les différents éléments à considérer :

- **pubKey** est la clé publique de Bob et va servir à chiffrer l'information à partager
- **privKey** est la clé privée de Bob
- **objKey** est la clé symétrique AES-256 générée avec un CSPRNG¹ qui chiffre l'identifiant à partager
- **bobEncrObjKey** est la clé unique de l'objet chiffrée avec la clé publique de Bob
- **encrCredential** est l'identifiant à partager chiffré avec la clé générée à l'étape précédente

1. *Cryptographically secure pseudorandom number generator approprié pour la cryptographie* est un générateur de nombre pseudo-aléatoire

A chaque fois qu'un élément doit être partagé, on génère une clé symétrique aléatoirement qui est stockée sur les serveurs du provider à l'aide de la clé publique de l'utilisateur concerné. La clé est générée à l'aide d'un CSPRNG. Elle va permettre de chiffrer les identifiants avec AES-CBC et HMAC-SHA2, qui va permettre de garantir la confidentialité ainsi que l'intégrité en l'authentifiant.

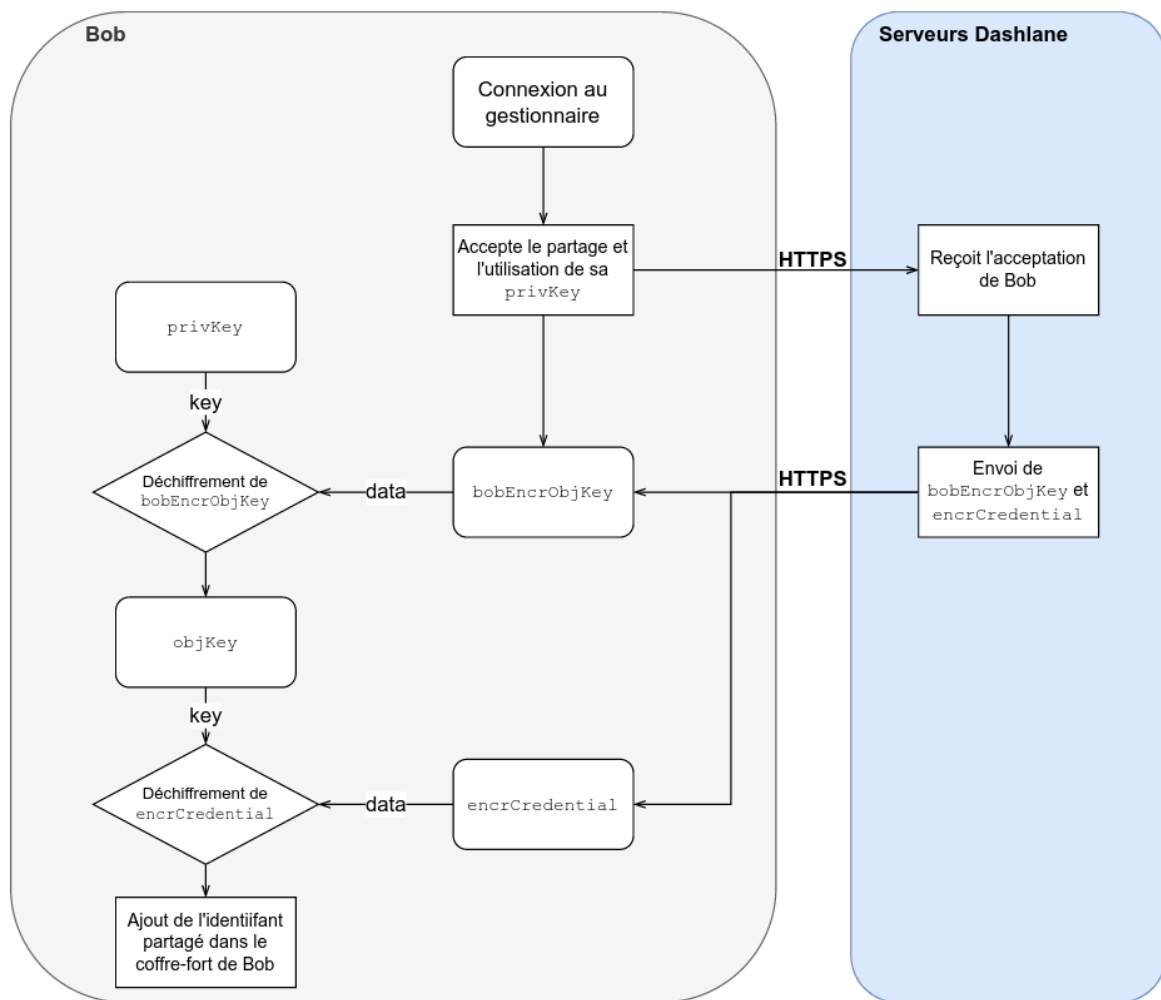


FIGURE 4.7 – Réception d'un partage d'identifiant sur Dashlane

Afin de récupérer le partage, Dashlane demande à l'utilisateur d'accepter l'envoi d'Alice pour qu'il accepte d'utiliser sa clé privée. Tout dépend de l'architecture de l'application, mais cette dernière doit d'abord être déchiffrée avant d'être utilisée pour déchiffrer l'identifiant envoyé par Alice.

Les identifiants partagés sont ajoutés finalement au coffre-fort de Bob. Ces derniers seront synchronisés par la suite sur tous les autres appareils de Bob dès qu'il sera connecté dessus.

Certains gestionnaires proposent de faire des dossiers partagés. Dans le cadre d'entreprises, cela est une fonctionnalité très utile. Le processus est le même que pour le partage avec une personne. Une clé pour le dossier à partager est générée et on chiffre chaque clé publique de tous les utilisateurs concernés à l'aide de cette dernière.

4.3 4 états du gestionnaire de mot de passe

Les gestionnaires de mots de passe ont généralement 4 états différents : *Not Configured*, *Not Running*, *Unlocked State* et *Locked State*. Nous allons expliquer plus en détails comment ils fonctionnent lorsqu'ils sont dans les différents états. Nous nous basons sur un article qui analyse le management des secrets[19].

4.3.1 État *Not Configured*

On définit cet état lorsque l'application n'a jamais été utilisée après son installation et n'a pas encore été configurée, c'est-à-dire inscription de l'utilisateur avec la création de son compte. Aucun secret n'est stocké en mémoire.

4.3.2 État *Not Running*

C'est l'état du password manager lorsqu'il n'a pas été lancé depuis le dernier redémarrage du système ou a été arrêté par un utilisateur mais qu'il a déjà été configuré. Dans cet état, le gestionnaire doit garantir qu'il n'y a aucune donnée sensible stockée sur le disque qui permettrait de compromettre le coffre-fort, comme une clé de chiffrement ou le master password.

4.3.3 État *Unlocked State*

Cet état indique que le gestionnaire fonctionne et donc, l'utilisateur a entré son master password afin de déchiffrer toutes les données afin d'avoir accès aux informations stockées. Le gestionnaire doit garantir qu'il n'est pas possible d'extraire d'information sensible de la mémoire. De plus, à part pour les mots de passe avec lesquels l'utilisateur a eu une interaction (affichage, accès ou copie), il ne devrait pas être possible d'extraire déchiffrés depuis la mémoire le reste des mots de passe.

4.3.4 État *Locked State*

Nous considérons cet état lorsque l'utilisateur a lancé le gestionnaire (déjà configuré) sans avoir encore entré le master password, qu'il a lui même verrouillé son gestionnaire ou qu'il y a eu un timeout de session. À ce moment, il ne devrait pas y avoir de données sensibles stockées sur le disque et sur la mémoire RAM afin d'éviter toute extraction.

Ci-dessous un schéma qui permet de résumer tout ce qui a été expliqué dans les sections précédentes :

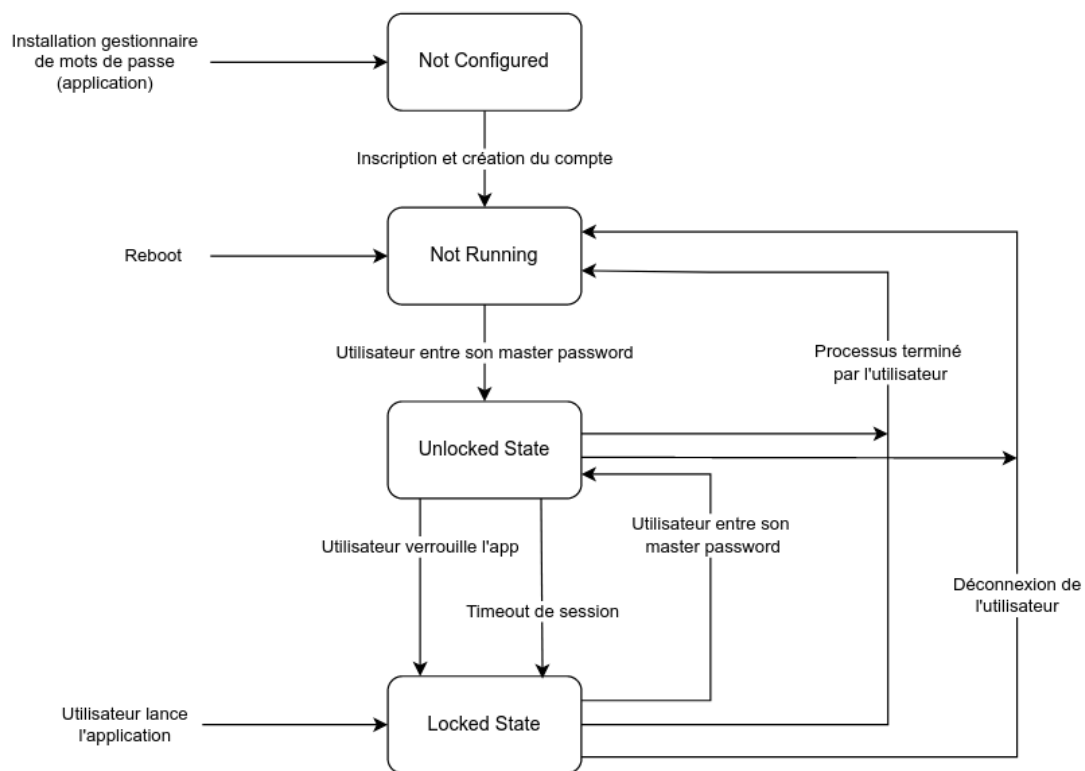


FIGURE 4.8 – Différents états d'un gestionnaire de mots de passe

4.4 Algorithmes cryptographiques

Afin d'avoir une bonne vue d'ensemble de l'architecture sécuritaire des gestionnaires de mots de passe sélectionnés pour notre étude, nous allons résumer les algorithmes cryptographiques utilisés pour le chiffrement des données, la dérivation des clés et l'authentification.

	Chiffrement des données	Dérivation des clés		Partage de données
LastPass	AES-CBC 256	PBKDF2-SHA2 100'000 rounds		RSA-2048
Dashlane	AES-256	Argon2d	PBKDF2-SHA2 200'000 rounds	RSA-2048
1Password	AES-GCM 256	PBKDF2-SHA2 100'000 rounds		RSA-OAEP 2048
KeePass	AES-CBC 256/ChaCha20	AES-KDF	Argon2	-
Bitwarden	AES-CBC 256	PBKDF2-SHA2 100'000 rounds		RSA-2048
NordPass	XChaCha20-Poly1305-IETF	Argon2id		? ²
Padloc	AES-GCM 256	PBKDF2-SHA2		RSA-PSS
Keeper	AES-256	PBKDF2		RSA
Firefox	AES-CBC/3DES-CBC	PBKDF2-SHA2 et HKDF 1000 rounds		-

TABLE 4.1 – Algorithmes cryptographiques des candidats

Nous pouvons remarquer que certains gestionnaire ont plusieurs algorithmes différents pour le même élément, ceci est dû au fait que l'utilisateur a le choix de configurer celui qu'il préfère et que l'application supporte les deux.

2. Information pas précisée sur leur documentation, toutefois il s'agit sûrement de RSA

Chapitre 5

Analyse de menaces

Ce chapitre a pour but d'identifier et analyser toutes les menaces existantes et / ou potentielles des *password managers* en les modélisant en suivant un certain processus afin d'avoir une meilleure vision des risques.

Puis, nous allons rédiger toutes les exigences sécuritaires que doivent respecter les gestionnaires de mots de passe afin que ces dernières garantissent une utilisation sûre qui évite des pertes, un vol de données, une usurpation d'identité, des accès aux services, etc.

5.1 Modélisation de menaces

Afin de modéliser correctement les menaces, nous allons suivre la norme ISO 27005[20]. Cela va nous permettre de séparer la modélisation en un processus avec plusieurs étapes comme suit :

1. Établissement du contexte, qui inclut
 - Objectifs des gestionnaires de mots de passe
 - Hypothèses et exigences de sécurité
 - Actifs à haute valeur
 - Data Flow Diagram
 - Définition des critères d'analyse
2. Identification des risques
 - Identification des biens
 - Identification des menaces
 - Identification des contrôles

- Identification des vulnérabilités
 - Identification des conséquences
3. Analyse des risques
 4. Évaluation des risques
 5. Traitement des risques
 6. Documentation

La dernière étape ne sera pas explicitement abordée car elle vise à documenter le modèle de menaces que nous allons établir.

Pour chaque étape du processus, nous allons aborder les 3 types de questionnaires que nous avons cités précédemment : local-based, browser-based et cloud/local-based. Étant donné que les applications browser-based et cloud/local-based fonctionnent les deux en local et en cloud avec la synchronisation de données, nous pouvons nous baser sur deux catégories uniquement ; les questionnaires **local-based** et **cloud/local-based**.

Nous allons également nous baser sur le processus de modélisation de menaces de OWASP[35].

Il est intéressant de mélanger les deux méthodologies car la norme ISO est plus axée sur une gestion des risques alors que OWASP se base sur une modélisation des menaces. Les deux ont des points communs notamment au niveau de la description et la décomposition de l'application, néanmoins la norme ISO va plus en profondeur en identifiant tous les risques possibles (vulnérabilités, menaces, conséquences, etc.). OWASP a également ses avantages en schématisant le flux de données à l'aide de DFDs (*Data Flow Diagram*) et en catégorisant les menaces avec le modèle STRIDE.

5.1.1 Définition des critères

Avant de commencer, nous allons définir les différents critères et niveaux pour évaluer l'impact des événements (les conséquences), la probabilité d'événement, la sévérité des vulnérabilités ainsi que l'évaluation des risques. Ces niveaux seront réutilisés pour l'identification des risques dans la suite de l'analyse de menaces.

Échelle	Description	Valeur
Aucune	La vulnérabilité ne présente aucune sévérité	0
Bas	La vulnérabilité a peu d'impact sur l'entreprise et demande un accès physique ou local au système	0.1-39
Moyen	En général, elle demande à l'attaquant d'être sur le même réseau que la victime et elle demande les privilèges utilisateurs pour effectuer une exploitation	4.0-6.9
Haut	La vulnérabilité est difficile à exploiter, peut être une élévation de privilèges et peut amener à une importante perte de données ou de temps d'arrêt	7.0-8.9
Critique	Exploitation de vulnérabilités au niveau root, l'attaquant n'a pas besoin d'être authentifié ou avoir des connaissances sur la victime	9.0-10.0

TABLE 5.1 – Scores de sévérité basés sur CVSS v3.1

Échelle	Description	Valeur
Négligeable	L'impact sur l'entreprise est négligeable	0
Mineur	L'effet sur les biens de l'entreprise est limité, comme entraîner des pertes financières mineures ou entraîner des dommages mineures aux actifs de l'entreprises	1
Modéré	L'effet sur les biens peut être grave, les effets causés sur les biens seront considérés comme importants	2
Important	L'effet sur les biens de l'entreprise peut être grave à catastrophique	3
Catastrophique	On s'attend à ce que la menace ait de multiples effets graves à catastrophiques sur les biens de l'entreprise	4

TABLE 5.2 – Impact des menaces sur l'entreprise

Échelle	Description	Valeur
Rare	La probabilité que l'événement arrive est rare	0
Peu probable	La probabilité que l'événement arrive est peu probable	1
Possible	La probabilité que l'événement arrive est possible	2
Probable	La probabilité que l'événement arrive est probable	3
Certain	La probabilité que l'événement arrive est certain	4

TABLE 5.3 – Probabilités d'événements

Pour l'évaluation des risques, il est important de prendre en considération la probabilité d'événements et les conséquences sur l'entreprise afin de définir une échelle d'évaluation de risques. Nous utilisons une matrice de risques basée sur une étude quantitative[49].

		Impact				
		0	1	2	3	4
		Négligeable	Mineur	Modéré	Important	Catastrophique
Probabilité	0 Rare	0	1	2	3	4
	1 Peu probable	1	2	3	4	5
	2 Possible	2	3	4	5	6
	3 Probable	3	4	5	6	7
	4 Certain	4	5	6	7	8

risque bas: 0-2
 risque moyen: 3-5
 risque haut: 6-8

FIGURE 5.1 – Critères d'évaluation des risques

5.1.2 Établissement du contexte

Dans cette section, nous allons comprendre les applications et comment interagissent les gestionnaires de mots de passe avec les entités externes. Au final, nous allons établir un *Data Flow Diagram* (DFD) qui va nous permettre de présenter tous les chemins différents du système en mettant en avant les vulnérabilités potentielles.

Objectifs du système

Comme cité plusieurs fois, l'objectif principal d'un gestionnaire de mots de passe est de stocker de manière **sûre** des informations sensibles, le plus souvent des identifiants, mais également la possibilité de stocker des notes, des informations bancaires, contrats, etc. Ils offrent également la fonctionnalité de générer des mots de passe forts et d'auto-compléter les champs de connexion.

Tout dépend, du gestionnaire de mots de passe, ils peuvent offrir une fonctionnalité de backup, afin de sauvegarder le coffre-fort et de garantir une perte minimale de données au cas où. Il existe également la fonctionnalité de partage de données entre utilisateurs.

Pour les applications qui fonctionnent avec le cloud et qui ne sont pas 100% offline, l'utilisateur a la possibilité de synchroniser ses données entre appareils (desktop, mobile, navigateur, smartwatch).

Hypothèses de sécurité

On peut émettre plusieurs hypothèses de sécurité afin de mettre en avant ce que doit assurer le gestionnaire de mots de passe pour être considéré comme sûr. Nous allons lister les hypothèses afin que cela soit le plus clair possible :

- Données du gestionnaire soient chiffrées et protégées en RAM, sur le disque ainsi que lors de communications avec les serveurs. Nous définissons les données comme étant le coffre-fort, avec tous les identifiants, les clés et le master password.
- Serveurs de confiance

Ceux-ci étant les plus importants afin de garantir la meilleure sécurité possible sur l'application, nous irons plus en détails dans la section sur les exigences sécuritaires.

Actifs de valeurs

Nous pouvons à présent définir les actifs (*assets*) des gestionnaires de mots de passe, qui représentent les éléments qui ont de la valeur et qui demandent une importante protection.

Tout d'abord, un actif à haute valeur est le **gestionnaire de mot passe** qui représente l'application. Cette dernière peut être sous forme d'application desktop, extension de navigateur ou application mobile. Un utilisateur ou un administrateur devrait pouvoir se connecter à l'application pour avoir accès à son coffre-fort.

Ensuite, un autre bien important est le **coffre-fort** qui contient tous les identifiants, notes, informations bancaires, etc. stockées par l'utilisateur, qui peut être représenté comme une base de données. Le bien principal sont donc les données, qui sont chiffrées et protégées. Un utilisateur devrait ainsi avoir l'habilité d'avoir accès à son coffre-fort en clair et de pouvoir y ajouter au minimum des identifiants.

Un actif qu'on peut également ajouter est le **master password** entré l'utilisateur pour déverrouiller le gestionnaire de mots de passe et avoir accès à son coffre-fort. Cet élément est très important car il va permettre de dériver les clés de chiffrement et ainsi, de pouvoir avoir accès au coffre-fort en clair.

Un autre actif à haute valeur sont les **clés**. Nous incluons dans cet actif les clés de chiffrement ainsi que les clés privées lors du partage de données entre utilisateurs. Nous pouvons ajouter le processus de génération de clés.

Dans le cadre de gestionnaires de mots de passe cloud/local-based qui ne fonctionnent pas

en mode offline, nous voulons protéger les données des **serveurs du constructeur**. Le processus de transfert de données, c'est-à-dire la communication, est également important car les données doivent impérativement être protégées. Nous pouvons inclure les fonctionnalités de synchronisation de données et le partage de données entre utilisateur, qui demandent de passer par les serveurs du constructeur.

Un asset important à ajouter sont les **données partagées** entre utilisateurs lors de la fonctionnalité de partage. La ou les données échangées doivent être sécurisées afin de ne pas les exposer à n'importe quel utilisateur.

Finalement, nous pouvons ajouter comme dernier actif, les fichier de **backups** qui devraient être chiffrés et protégés sur les devices de l'utilisateur ou des serveurs du provider.

À présent, pour effectuer une analyse de menaces la plus claire et structurée possible, nous allons séparer l'application en plusieurs parties comme suit :

- M1** Coffre-fort standalone, local-based (haut-niveau)
- M2** Fonctionnalité de backup
- M3** Fonctionnalité de synchronisation (pour les gestionnaires cloud-based)
- M4** Fonctionnalité de partage
- M5** Sécurité de l'application et ses composants (bas-niveau)

Pour chaque étape, nous identifierons et analyserons les risques pour couvrir tous les aspects et les fonctionnalités d'un gestionnaire de mots de passe. UN code d'identification a été ajouté afin de mieux organiser l'analyse.

Nous ferons un récapitulatif à la fin en évaluant tous les risques définis sur toutes les étapes et en listant toutes les contre-mesures nécessaires à prendre.

5.1.3 M1 Application haut-niveau

La première partie qu'on analyse, définit une application standalone toute simple, qui est un gestionnaire de mots de passe qui ne fonctionne qu'en local et qui interagit avec aucune entité externe. Premièrement, nous avons effectué un DFD (*Data Flow Diagram*) afin de constater les processus principaux de l'application :

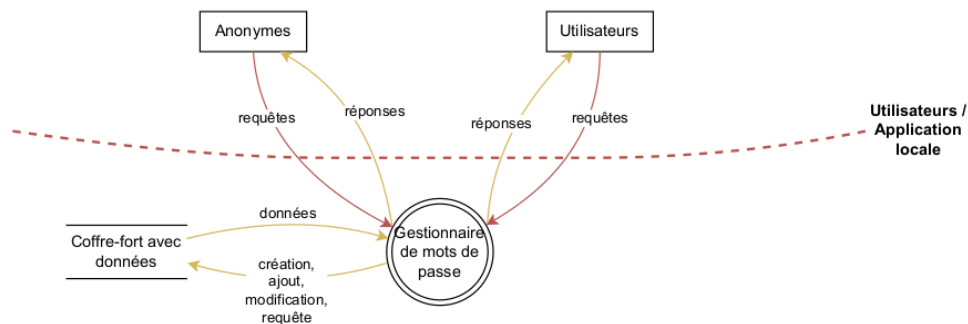


FIGURE 5.2 – Data Flow Diagram pour les gestionnaires M1

5.1.3.1 Identification des risques

Cette section va nous servir à déterminer ce qui peut se produire pour causer un dommage et à comprendre comment, où et pourquoi l'incident peut se produire.

Afin d'établir une étude complète, nous allons premièrement identifier les biens, menaces (avec les sources de menaces et les scénarios), les contrôles, les vulnérabilités ainsi que les conséquences sur l'entreprise.

Identification des biens

Nous avons déjà identifié les actifs plus-haut dans l'analyse, cependant nous allons identifier les actifs présents dans une application standalone. De plus, nous allons ajouter un code pour les identifier et les référencer plus tard dans l'identification des risques.

Code	Bien	Description
A1	Gestionnaire de mots de passe	L'application du gestionnaire de mots de passe
A2	Coffre-fort	Base de données avec toutes les informations stockées
A3	Master password	Mot de passe principal afin de déchiffrer le coffre-fort

TABLE 5.4 – Biens des gestionnaires de mots de passe M1

Identification des sources de menaces

Nous allons identifier toutes les sources de menaces possibles des gestionnaires de mots de passe avec leurs motivations (uniquement si c'est de source humaine) afin de pouvoir mettre en avant les différents scénarios de menaces.

Code	Source de menace	Motivations
S1	Hackers	Amusement, gloire, challenge, argent, ego
S2	Script kiddies	Curiosité, ego, argent, amusement
S3	Concurrent	Espionnage, réutilisation de contenu
S4	Cybercrime	Argent, destruction de données
S5	Accident humain	-
S6	Problèmes techniques	-

TABLE 5.5 – Sources de menaces des gestionnaires de mots de passe M1

Ci-dessous, se trouve un tableau avec différents scénarios d'attaques par rapport aux biens que nous avons défini précédemment. Nous nous basons sur le modèle STRIDE¹ afin de définir les types de menaces.

Code	Code du bien	Scénario de menace	Type de menace	Source de menace
2	T2	A1, A3	Key-logger installé sur le device de l'utilisateur et qui sniff le master password	<i>Elevation of privileges</i> S1, S2, S4
2	T2	A1	Utilisateur non-déconnecté de son compte et attaquant a accès au device	<i>Elevation of privileges</i> S1, S2, S4, S5
3	T1	A1	Mauvais mécanisme d'authentification	<i>Spoofing</i> S1, S2, S4, S6
4	T3	A1	Lecture du presse-papier	<i>Information disclosure</i> S1, S2, S4
5	T1	A1	Coffre-fort compromis dû à des identifiants volés sur d'autres sites	<i>Spoofing</i> S1, S2, S4, S5
6	T3	A2	Récupération du coffre-fort et déchiffrement dû à un algorithme trop simple	<i>Information disclosure</i> S1, S2, S4
7	T6	A2	Injection SQL dans la base de données utilisateur	<i>Tampering</i> S1, S2, S4
8	T1	A3	Brute-force sur le master password	<i>Spoofing</i> S1, S2, S4, S5
9	T1	A3	Phishing par mail pour récupérer le master password	<i>Spoofing</i> S1, S2, S3, S4

A1 = Gestionnaire de mots de passe | A2 = Coffre-fort | A3 = Master password

TABLE 5.6 – Scénarios et types d'attaques possibles sur applications M1

Identification des contrôles

Une étape importante lors de l'identification des risques est d'identifier les contrôles qui existent déjà sur les applications sur lesquelles nous basons notre analyse de risques. Étant donné, que notre étude est plutôt générale car elle ne se base pas sur un seul gestionnaire de mots de passe, nous allons énumérer les contrôles qui ont été entrepris afin de baisser le niveau de risque. Nous parlerons en détails des contrôles effectués lors de la seconde partie du travail.

1. https://owasp.org/www-community/Threat_Modeling_Process

Étant donné que dans cette partie, nous nous concentrons sur une application basique qui ne fonctionne qu'en local et que nous abordons pas le bas-niveau, nous n'irons pas en détails au niveau des contrôles.

Ce qui existe déjà pour éviter des attaques sur mots de passe sont des critères qui sont, pour la plupart des gestionnaires, imposés par le constructeur. Ces critères exigent par exemple d'avoir un certain nombre de caractères afin d'avoir un master password fort.

Au niveau de la protection du keylogger, aucune protection n'est réellement implémentée, à part utiliser l'application dans environnement sécurisé comme *Secure Desktop* sur Windows. Pour la protection du presse-papier, certaines applications (comme KeePass) mettent à disposition une fonctionnalité pour enlever les données sensibles du presse-papier et de la mémoire après un certain moment.

Pour le coffre-fort, la plupart des gestionnaires de mots de passe proposent un chiffrement fort du coffre-fort afin de protéger au mieux les données, afin que même si ce dernier est volé, la personne malveillante ne puisse pas le déchiffrer et récupérer les données stockées.

Identification des vulnérabilités

Pour chaque actif identifié au préalable, nous allons analyser les vulnérabilités qui pourraient être présentes. Nous allons également ajouter la sévérité de la vulnérabilité.

Bien	Vulnérabilité	Sévérité de la vulnérabilité
A1	Temps de session long	Critique
A1	Presse-papier non nettoyé après un certain temps	Moyen
A1	2FA pas proposé par défaut	Haut
A1	Aucune authentification effectuée	Moyen
A1	Génération de mots de passe faibles	Haut
A1	Manque de contrôle des entrées utilisateurs	Haut
A2	Possibilité d'avoir des entrées dans le coffre-fort qui ont le même mot de passe	Haut
A3	Master password trop faible	Haut
A3	Master password non unique	Bas

A1 = Gestionnaire de mots de passe

TABLE 5.7 – Vulnérabilités présentes dans les gestionnaires de mots de passe M1

Identifications des conséquences

Il est également important d'identifier les conséquences qui pourraient être causés par un incident. Nous allons définir par bien actif quelles sont les conséquences qu'ils pourraient y avoir sur l'entreprise lors d'une perte de confidentialité, intégrité ou disponibilité. Pour cela, nous allons nous référer au tableau qui définit différents scénarios de menaces. Nous allons également ajouter les éléments que nous souhaitons garantir pour chaque bien.

Bien	Conséquences	Garanties
Application (A1)	Perte de réputation et d'image	Disponibilité, intégrité
Base de données (A2)	Perte de réputation et d'image, perte de données	Confidentialité, intégrité des données
Master password (A3)	Perte de réputation et d'image, perte de données	Confidentialité, intégrité des données

TABLE 5.8 – Conséquences des menaces sur l'entreprise

5.1.3.2 Analyse des risques

L'objectif de cette section est d'estimer la probabilité des incidents et les conséquences pour les biens qu'ils menacent. Pour ce faire, nous allons faire une analyse de risque qualitative ce qui va nous permettre de faire une première analyse de risques assez générale afin de nous permettre d'aller plus en détails dans la seconde partie du travail de Bachelor.

Ainsi, nous allons analyser l'impact des conséquences ainsi que la probabilité que chaque scénario d'attaque identifié puisse se produire. Pour cela, nous allons utiliser plusieurs notations différentes ; pour les conséquences nous aurons les niveaux suivants :

Bien	Menace	Scénario	Impact des conséquences	Probabilité	Niveau de risque
A1, A3	T2	Key-logger installé sur le device de l'utilisateur et qui sniff le master password	Important	Possible	Moyen
A1	T2	Utilisateur non-déconnecté de son compte et attaquant a accès au device	Important	Probable	Haut
A1	T1	Mauvais mécanisme d'authentification	Modéré	Rare	Bas
A1	T3	Lecture du presse-papier	Modéré	Probable	Moyen
A1	T1	Coffre-fort compromis dû à des identifiants volés sur d'autres sites	Mineur	Possible	Moyen
A2	T3	Récupération de la base de données et déchiffrement dû à un algorithme trop simple	Important	Peu probable	Moyen
A2	T6	Injection SQL dans la base de données	Important	Peu probable	Moyen
A3	T1	Brute-force sur le master password	Important	Certain	Haut
A3	T1	Phishing par mail pour récupérer le master password	Modéré	Possible	Moyen

A1 = Gestionnaire de mots de passe | A2 = Base de données | A3 = Master password

T1 = Spoofing | T2 = Elevation of privileges | T3 = Information Disclosure | T4 = Denial of service | T5 = Equipment failure | T6 = Tampering

TABLE 5.9 – Analyse des risques de chaque scénario de menaces pour M1

5.1.4 M2 Backup

Cette partie concerne la fonctionnalité de backup des coffre-forts qui est offerte par tous les gestionnaires de mots de passe. Il y a plusieurs techniques de sauvegardes qui existent ; une solution proposée par les applications en local est l'exportation des données avec un fichier CSV (toutes les données sont en claires) ou un fichier chiffré. Sinon, il y a la possibilité d'effectuer des sauvegardes sur les serveurs du provider, seulement s'il s'agit d'un gestionnaire cloud-based. Un DFD a été défini ci-dessous afin de mieux comprendre le processus de sauvegarde :

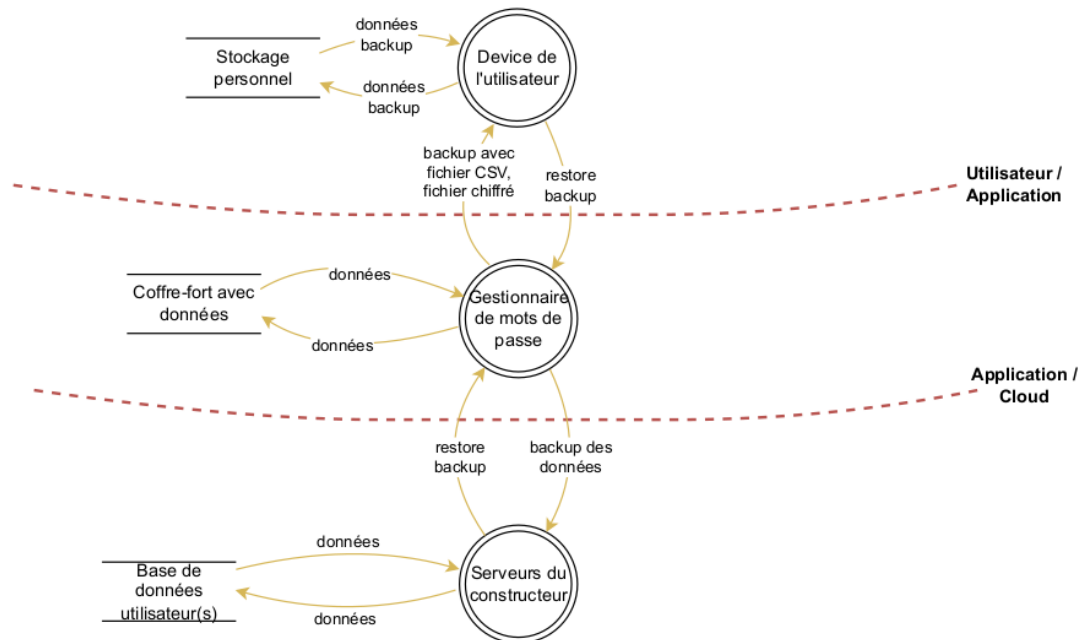


FIGURE 5.3 – Data Flow Diagram pour les applications M2

5.1.4.1 Identification des risques

Nous allons procéder exactement comme la section précédente, nous allons premièrement identifier les biens, menaces (avec les sources de menaces et les scénarios), les contrôles, les vulnérabilités ainsi que les conséquences sur l'entreprise afin d'établir une étude complète.

Identification des biens

Ci-dessous les biens présents lors du processus de sauvegarde de données dans un gestionnaire de mots de passe local-based ou cloud/local-based.

Code	Bien	Description
A1	Serveurs du constructeur	Serveurs qui stockent les données des utilisateur dans le cloud
A2	Sauvegardes	Fichiers ou emplacements contenant les sauvegardes des données

TABLE 5.10 – Biens des gestionnaires de mots de passe M2

Identification des sources de menaces

Nous allons identifier toutes les sources de menaces possibles qui pourraient survenir dans le cadre de backups de données d'un gestionnaire de mots de passe.

Code	Source de menace	Motivations
S1	Hackers	Amusement, gloire, challenge, argent, ego
S2	Script kiddies	Curiosité, ego, argent, amusement
S3	Concurrent	Espionnage, réutilisation de contenu
S4	Cybercrime	Argent, destruction de données
S5	Accident humain	-
S6	Problèmes techniques	-

TABLE 5.11 – Sources de menaces des gestionnaires de mots de passe M2

Ci-dessous, se trouve un tableau avec différents scénarios d'attaques. Nous utilisons le modèle STRIDE afin de définir tous les types de menaces.

	Code	Code du bien	Scénario de menace	Type de menace	Source de menace
1	T3	A1	Communication non chiffrée entre gestionnaire de mots de passe et serveurs	<i>Information Disclosure</i>	S1, S2, S4, S6
2	T3	A1	MITM (Interception de données non-chiffrées)	<i>Information Disclosure</i>	S1, S2, S4, S6
3	T4	A1	DoS	<i>Denial of service</i>	S1, S2, S4, S6
4	T5	A1	Perte des données utilisateurs dû à un serveur down	<i>Equipment failure</i>	S5, S6
5	T3	A2	Récupération du backup en clair sur le device du client	<i>Information Disclosure</i>	S1, S2, S3, S4, S6
6	T3	A2	Sauvegarde corrompue	<i>Information Disclosure</i>	S5, S6
7	T3	A2	Perte de la sauvegarde sur device utilisateur	<i>Information Disclosure</i>	S1, S2, S3, S4, S6
8	T3	A2	Vols de backups non protégés sur le serveur	<i>Information disclosure</i>	S1, S2, S4, S6

A1 = Serveurs du constructeur | A2 = Sauvegardes

TABLE 5.12 – Scénarios et types d'attaques possibles sur M2

Identification des contrôles

Dans cette partie, nous nous concentrons uniquement sur la fonctionnalité de backup, ainsi, la liste des contrôles existants n'est pas grande.

L'unique contrôle que l'on peut citer, qui est fait de manière indirect, est que la sauvegarde sur les serveurs du constructeur est entièrement chiffrée. Étant donné que dans leurs bases de

données utilisateurs, le coffre-fort est toujours chiffré grâce au *Zero-knowledge*, naturellement le backup le sera aussi. Cela nous garantit que les données ne sont pas connues des serveurs et sont protégées de quelconques attaques sur ces derniers.

Au niveau du device utilisateur, il est possible d'effectuer des contrôles mais c'est à l'utilisateur de s'assurer que les données sont stockées dans un emplacement sécurisé et à l'abri de toute manipulation malveillante.

Identification des vulnérabilités

Pour chaque actif identifié au préalable, nous allons analyser les vulnérabilités qui pourraient être présentes. Nous allons également ajouter la sévérité de la vulnérabilité.

Bien	Vulnérabilité	Sévérité de la vulnérabilité
A1	Backup non effectué sur les serveurs	Critique
A1	Communication non chiffrée	Critique
A1	Infrastructure des serveurs pas assez stable	Haut
A1	Infrastructure des serveurs du constructeur pas protégée à quelconques intrusions physiques	Bas
A2	Manque de chiffrement dans les sauvegardes sur serveurs ou device utilisateur	Critique
A2	Sauvegarde stockée dans un répertoire personnel non protégé	Moyen
A2	Sauvegarde CSV stockée dans un emplacement non sécurisé	Moyen

A1 = Serveurs du constructeur | A2 = Sauvegardes

TABLE 5.13 – Vulnérabilités présentes dans les questionnaires de mots de passe M2

Identifications des conséquences

Pour chaque bien défini plus haut, nous allons ajouter les conséquences qu'il pourrait y avoir dans le cas d'attaques, ainsi que ce que nous souhaitons garantir.

Bien	Conséquences	Garanties
Serveurs du constructeur (A1)	Perte de réputation et d'image, pertes de données	Disponibilité, intégrité, confidentialité
Sauvegardes (A2)	Perte de réputation et d'image, perte de données	Confidentialité, intégrité des données

TABLE 5.14 – Conséquences des menaces sur l'entreprise

5.1.4.2 Analyse des risques

Ainsi, dans cette partie, nous allons analyser l'impact des conséquences ainsi que la probabilité que chaque scénario d'attaque identifié puisse se produire. Pour cela, nous allons utiliser plusieurs notations différentes ; pour les conséquences nous aurons les niveaux suivants :

Bien	Menace	Scénario	Impact des conséquences	Probabilité	Niveau de risque
A1	T3	Communication non chiffrée entre gestionnaire de mots de passe et serveurs	Catastrophique	Peu probable	Moyen
A1	T3	MITM (Interception de données non-chiffrées)	Catastrophique	Peu probable	Moyen
A1	T4	DoS	Modéré	Possible	Moyen
A1	T5	Perte des données utilisateurs dû à un serveur down	Catastrophique	Peu probable	Moyen
A2	T3	Récupération du backup en clair sur le device du client	Modéré	Probable	Moyen
A2	T3	Sauvegarde corrompue	Modéré	Rare	Bas
A2	T3	Perte de la sauvegarde sur device utilisateur	Mineur	Possible	Moyen
A2	T3	Vols de backups non protégés sur le serveur	Catastrophique	Peu probable	Moyen

A1 = Serveurs du constructeur | A2 = Sauvegardes

T1 = Spoofing | T2 = Elevation of privileges | T3 = Information Disclosure | T4 = Denial of service | T5 = Equipment failure | T6 = Tampering

TABLE 5.15 – Analyse des risques de chaque scénario de menace sur M2

5.1.5 M3 Synchronisation

Cette section effectue une analyse de menaces sur la fonctionnalité de synchronisation des gestionnaires de mots de passe. Elle permet de synchroniser les données d'un utilisateur sur un ou plusieurs de ses devices. Cette option concerne uniquement les gestionnaires de mots de passe qui fonctionnent avec le cloud. Ainsi, l'application communique en continu avec les serveurs du constructeur. De plus, nous allons effectuer une analyse sur les fonctionnalités qu'un gestionnaire cloud-based peut proposer, par exemple l'auto-complétion des champs de formulaires.

Nous avons créé un DFD afin de comprendre comment les communications fonctionnent :

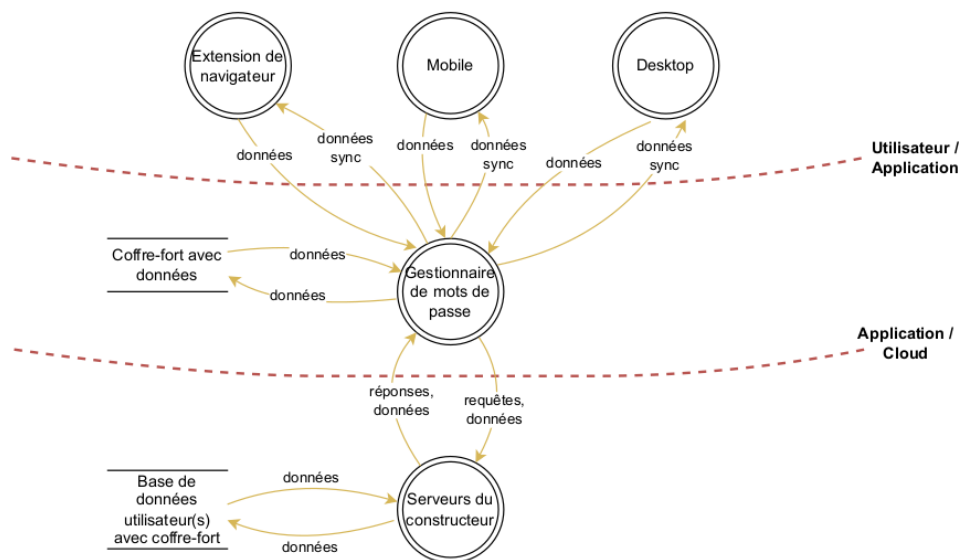


FIGURE 5.4 – Data Flow Diagram pour les synchronisations

5.1.5.1 Identification des risques

Nous allons procéder exactement comme la section précédente, nous allons premièrement identifier les biens, menaces (avec les sources de menaces et les scénarios), les contrôles, les vulnérabilités ainsi que les conséquences sur l'entreprise afin d'établir une étude complète.

Identification des biens

Ci-dessous les biens présents lors du processus de sauvegarde de données dans un gestionnaire de mots de passe cloud-based.

Code	Bien	Description
A1	Gestionnaire de mots de passe	L'application du gestionnaire de mots de passe
A2	Coffre-fort	Base de données avec toutes les informations stockées
A3	Serveurs du constructeur	Serveurs qui stockent les données des utilisateur dans le cloud

TABLE 5.16 – Biens des gestionnaires de mots de passe M3

Identification des sources de menaces

Nous allons identifier toutes les sources de menaces possibles qui pourraient survenir dans le cadre de synchronisations de données d'un gestionnaire de mots de passe.

Code	Source de menace	Motivations
S1	Hackers	Amusement, gloire, challenge, argent, ego
S2	Script kiddies	Curiosité, ego, argent, amusement
S3	Concurrent	Espionnage, réutilisation de contenu
S4	Cybercrime	Argent, destruction de données
S5	Accident humain	-
S6	Problèmes techniques	-

TABLE 5.17 – Sources de menaces des gestionnaires de mots de passe M3

Ci-dessous, se trouve un tableau avec différents scénarios d'attaques. Nous utilisons le modèle STRIDE afin de définir tous les types de menaces.

	Code	Code du bien	Scénario de menace	Type de menace	Source de menace
1	T1	A1	Phishing en imitant une page de connexion	<i>Spoofing</i>	S1, S2, S3, S4
2	T1	A1	Attaque XSS avec les champs de connexion auto-complétés	<i>Spoofing</i>	S1, S2, S4
3	T3	A1	Clickjacking	<i>Information Disclosure</i>	S1, S2, S4
4	T3	A3	Communication non chiffrée entre gestionnaire de mots de passe et serveurs	<i>Information Disclosure</i>	S1, S2, S4, S6
5	T3	A3, A2	MITM (Interception de données non-chiffrées)	<i>Information Disclosure</i>	S1, S2, S4, S6
6	T4	A3	DoS	<i>Denial of service</i>	S1, S2, S4, S6
7	T5	A3	Perte des données utilisateurs dû à un serveur down	<i>Equipment failure</i>	S5, S6
8	T3	A3	Vols de données non protégées sur le serveur	<i>Information disclosure</i>	S1, S2, S4, S6

A1 = Gestionnaire de mots de passe | A2 = Coffre-fort | A3 = Serveurs du constructeur

TABLE 5.18 – Scénarios et types d'attaques possibles sur M3

Identification des contrôles

Il est possible de citer quelques contrôles concernant la fonctionnalité de synchronisation de données.

Premièrement, pour la communication avec les serveurs, tous les gestionnaires de mots de passe cloud-based utilisent le protocole HTTPS afin de garantir une communication chiffrée.

Les données synchronisées sur le device de l'utilisateur sont chiffrées et peuvent être déchiffrées dès le moment où l'utilisateur a entré le bon master password lors de sa connexion au gestionnaire de mots de passe.

Finalement, pour l'auto-complétion des champs de connexion, certains s'assurent que l'utilisateur valide s'il veut vraiment remplir ce champ afin d'éviter d'entrer les données sensibles dans des pages web clonées malveillantes.

Identification des vulnérabilités

Pour chaque actif identifié au préalable, nous allons analyser les vulnérabilités qui pourraient être présentes. Nous allons également ajouter la sévérité de la vulnérabilité.

Bien	Vulnérabilité	Sévérité de la vulnérabilité
A1	Auto-complétion de champs sans une interaction avec l'utilisateur	Moyen
A1	Utilisation de critères de correspondance faibles lors de l'auto-complétion	Moyen
A1	Règles d'auto-complétion HTTP ne différencie pas HTTP et HTTPS	Haut
A2	Utilisation d'algorithmes de chiffrement plus recommandés ou trop faibles	Haut
A3	Communication non chiffrée	Critique
A3	Infrastructure des serveurs pas assez stable	Haut
A3	Infrastructure des serveurs du constructeur pas protégée à quelques intrusions physiques	Bas

A1 = Gestionnaire de mots de passe | A2 = Coffre-fort | A3 = Serveurs du constructeur

TABLE 5.19 – Vulnérabilités présentes dans les gestionnaires de mots de passe M3

Identifications des conséquences

Pour chaque bien défini plus haut, nous allons ajouter les conséquences qu'il pourrait y avoir dans le cas d'attaques, ainsi que ce que nous souhaitons garantir.

Bien	Conséquences	Garanties
Application (A1)	Perte de réputation et d'image	Disponibilité, intégrité
Base de données (A2)	Perte de réputation et d'image, perte de données	Confidentialité, intégrité des données
Serveurs du constructeur (A3)	Perte de réputation et d'image, perte de données	Disponibilité, confidentialité, intégrité des données

TABLE 5.20 – Conséquences des menaces sur l'entreprise

5.1.5.2 Analyse des risques

Ainsi, dans cette partie, nous allons analyser l'impact des conséquences ainsi que la probabilité que chaque scénario d'attaque identifié puisse se produire. Pour cela, nous allons utiliser plusieurs notations différentes ; pour les conséquences nous aurons les niveaux suivants :

Bien	Menace	Scénario	Impact des conséquences	Probabilité	Niveau de risque
A1	T1	Phishing en imitant une page de connexion	Modéré	Possible	Moyen
A1	T1	Attaque XSS avec les champs de connexion auto-complétés	Important	Possible	Moyen
A1	T3	Clickjacking	Modéré	Possible	Moyen
A3	T3	Communication non chiffrée entre gestionnaire de mots de passe et serveurs	Catastrophique	Peu probable	Moyen
A3, A2	T3	MITM (Interception de données non-chiffrées)	Catastrophique	Peu probable	Moyen
A3	T4	DoS	Modéré	Possible	Moyen
A3	T5	Perte des données utilisateurs dû à un serveur down	Catastrophique	Peu probable	Moyen
A3	T3	Vols de données non protégées sur le serveur	Important	Rare	Moyen

A1 = Gestionnaire de mots de passe | A2 = Coffre-fort | A3 = Serveurs du constructeur

T1 = Spoofing | T2 = Elevation of privileges | T3 = Information Disclosure | T4 = Denial of service | T5 = Equipment failure | T6 = Tampering

TABLE 5.21 – Analyse des risques de chaque scénario de menaces sur M3

5.1.6 M4 Partage

Nous consacrons cette section à la fonctionnalité de partage de données qui est présente sur les gestionnaires de mots de passe cloud/local-based. Afin de mieux comprendre le processus, nous avons effectué un DFD qui explique comment un objet d'un utilisateur est partagé vers un autre utilisateur :

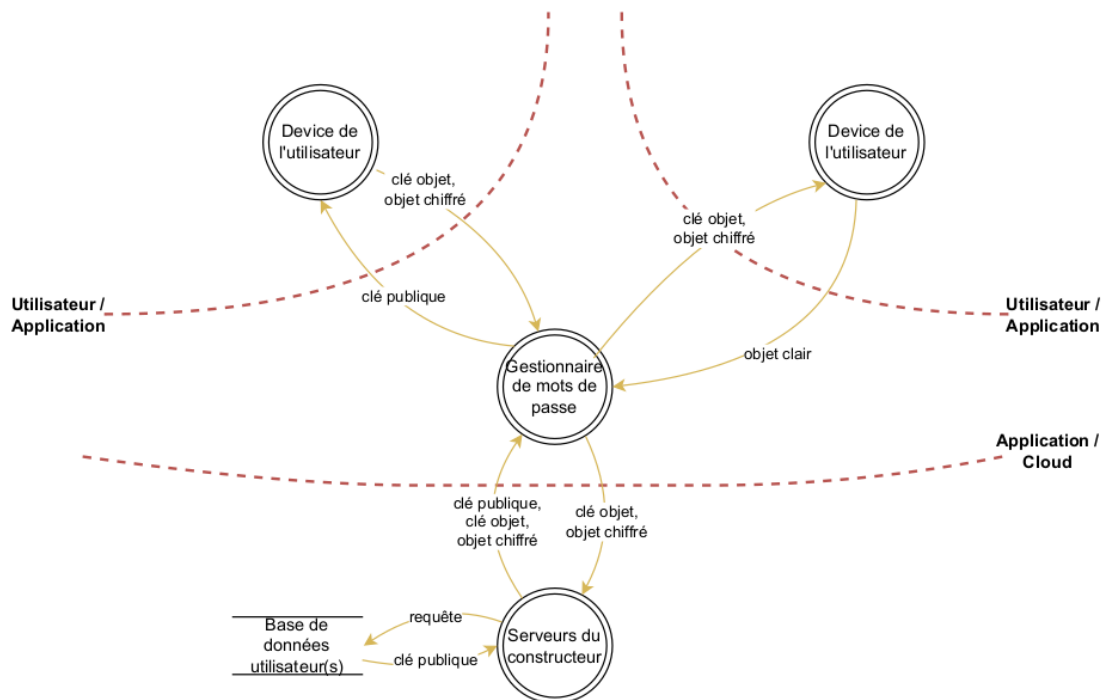


FIGURE 5.5 – Data Flow Diagram pour les partages de données

5.1.6.1 Identification des risques

Nous allons procéder exactement comme la section précédente, nous allons premièrement identifier les biens, menaces (avec les sources de menaces et les scénarios), les contrôles, les vulnérabilités ainsi que les conséquences sur l'entreprise afin d'établir une étude complète.

Identification des biens

Ci-dessous les biens présents lors du processus de partage entre deux utilisateurs.

Code	Bien	Description
A1	Serveurs du constructeur	Serveurs qui stockent les données des utilisateur dans le cloud
A2	Données partagées	Données qui sont partagées entre utilisateurs

TABLE 5.22 – Biens des gestionnaires de mots de passe M4

Identification des sources de menaces

Nous allons identifier toutes les sources de menaces possibles qui pourraient survenir dans le cadre de partage de données d'un gestionnaire de mots de passe.

Code	Source de menace	Motivations
S1	Hackers	Amusement, gloire, challenge, argent, ego
S2	Script kiddies	Curiosité, ego, argent, amusement
S3	Concurrent	Espionnage, réutilisation de contenu
S4	Cybercrime	Argent, destruction de données
S5	Accident humain	-
S6	Problèmes techniques	-

TABLE 5.23 – Sources de menaces des gestionnaires de mots de passe M4

Ci-dessous, se trouve un tableau avec différents scénarios d'attaques. Nous utilisons le modèle STRIDE afin de définir tous les types de menaces.

	Code	Code du bien	Scénario de menace	Type de menace	Source de menace
1	T3	A1	Communication non chiffrée entre gestionnaire de mots de passe et serveurs	<i>Information Disclosure</i>	S1, S2, S4, S6
2	T3	A1	MITM (Interception de données non-chiffrées)	<i>Information Disclosure</i>	S1, S2, S4, S6
3	T1	A1	Utilisateur auquel on partage des données n'est pas le bon	<i>Spoofing</i>	S1, S2, S4, S6
4	T3	A2	Donnée partagée non-chiffrée sur les serveurs du constructeur	<i>Information Disclosure</i>	S1, S2, S4, S6

A1 = Serveurs du constructeur | A2 = Données partagées

TABLE 5.24 – Scénarios et types d'attaques possibles sur M4

Identification des contrôles

Au niveau des contrôles effectuées lors du partage de données entre utilisateurs, il en existe quelques-uns.

Nous n'allons pas citer les contrôles que nous avons déjà pu expliquer dans les autres parties à propos de la communication des serveurs et de l'architecture *Zero-knowledge*, néanmoins nous pouvons ajouter qu'ils s'assurent que même les données à partager sont chiffrées sur les serveurs et que la clé pour chiffrer l'objet est bien chiffrée avec la clé publique de l'utilisateur.

Les serveurs du constructeur contrôlent également que l'utilisateur a qui on demande sa clé

publique est vraiment le bon afin d'éviter que des attaquants se fassent passer pour quelqu'un d'autre. Sur certains gestionnaires de mots de passe, ils s'assurent que la clé corresponde à l'e-mail adresse de l'utilisateur (cependant l'identité de la personne n'est pas contrôlée).

Identification des vulnérabilités

Pour chaque actif identifié au préalable, nous allons analyser les vulnérabilités qui pourraient être présentes. Nous allons également ajouter la sévérité de la vulnérabilité.

Bien	Vulnérabilité	Sévérité de la vulnérabilité
A1	Communication non chiffrée	Critique
A1	Manque de chiffrement lorsque les données sont envoyées sur les serveurs	Critique
A1	Manque de contrôle au niveau de l'identité de la personne	Moyen
A2	Non chiffrement de données sensibles	Critique

A1 = Serveurs du constructeur | A2 = Données partagées

TABLE 5.25 – Vulnérabilités présentes dans les gestionnaires de mots de passe M4

Identifications des conséquences

Pour chaque bien défini plus haut, nous allons ajouter les conséquences qu'il pourrait y avoir dans le cas d'attaques, ainsi que ce que nous souhaitons garantir.

Bien	Conséquences	Garanties
Serveurs du constructeur (A1)	Perte de réputation et d'image, pertes de données	Disponibilité, intégrité, confidentialité
Données partagées (A2)	Perte de réputation et d'image, perte de données	Confidentialité, intégrité des données

TABLE 5.26 – Conséquences des menaces sur l'entreprise

5.1.6.2 Analyse des risques

Ainsi, dans cette partie, nous allons analyser l'impact des conséquences ainsi que la probabilité que chaque scénario d'attaque identifié puisse se produire. Pour cela, nous allons utiliser plusieurs notations différentes ; pour les conséquences nous aurons les niveaux suivants :

Bien	Menace	Scénario	Impact des conséquences	Probabilité	Niveau de risque
A1	T3	Communication non chiffrée entre gestionnaire de mots de passe et serveurs	Catastrophique	Peu probable	Moyen
A1	T3	MITM (Interception de données non-chiffrées)	Catastrophique	Peu probable	Moyen
A1	T1	Utilisateur auquel on partage des données n'est pas le bon	Important	Probable	Haut
A2	T3	Donnée partagée non-chiffrée sur les serveurs du constructeur	Important	Peu probable	Moyen

A1 = Serveurs du constructeur | A2 = Données partagées

T1 = Spoofing | T2 = Elevation of privileges | T3 = Information Disclosure | T4 = Denial of service | T5 = Equipment failure | T6 = Tampering

TABLE 5.27 – Analyse des risques de chaque scénario de menaces sur M4

5.1.7 M5 Application bas-niveau

La dernière partie de cette analyse de menaces concerne la sécurité des gestionnaires de mots de passe et de tous ses composants, donc tout le bas-niveau. Cela inclut le stockage des données sensibles (clés, coffre-fort, master password) en mémoire RAM ainsi que sur le disque (fichiers de sauvegardes). Puis, nous aborderons les processus de chiffrement et génération des clés.

Ci-dessous un schéma représentant l'architecture bas-niveau des applications.

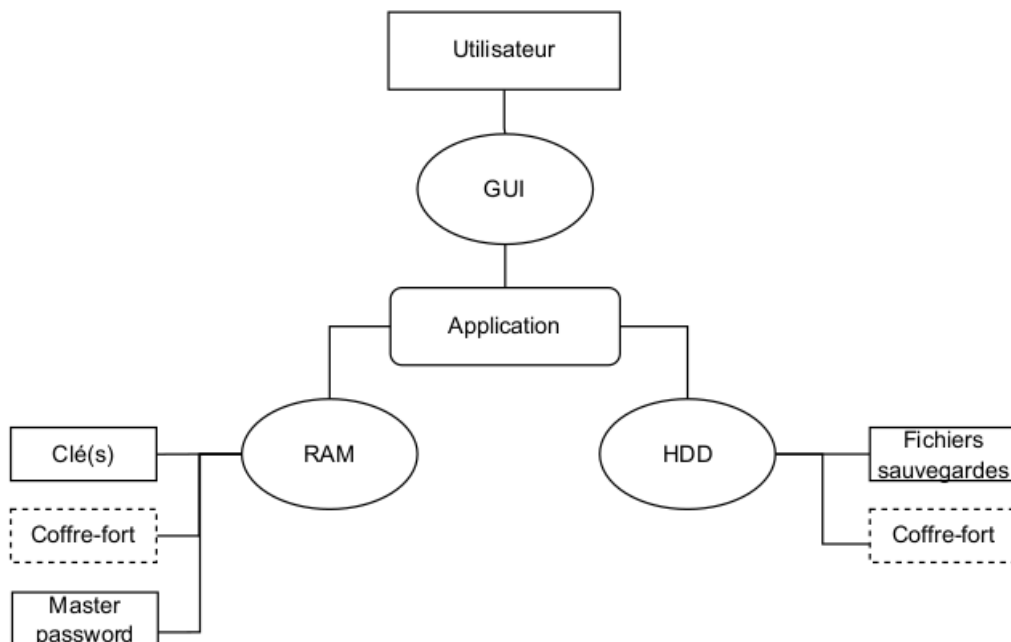


FIGURE 5.6 – Schéma bas-niveau des gestionnaires de mots de passe

5.1.7.1 Identification des risques

Nous allons procéder exactement comme la section précédente, nous allons premièrement identifier les biens, menaces (avec les sources de menaces et les scénarios), les contrôles, les vulnérabilités ainsi que les conséquences sur l'entreprise afin d'établir une étude complète.

Identification des biens

Ci-dessous les biens concernant la sécurité bas-niveau des gestionnaires de mots de passe.

Code	Bien	Description
A1	Coffre-fort	Base de données avec toutes les informations stockées
A2	Master password	Mot de passe principal afin de déchiffrer le coffre-fort
A3	Clés	Clé(s) de chiffrement et privée(s) de l'utilisateur

TABLE 5.28 – Biens des gestionnaires de mots de passe M5

Identification des sources de menaces

Nous allons identifier toutes les sources de menaces possibles qui pourraient survenir au niveau bas-niveau de l'application.

Code	Source de menace	Motivations
S1	Hackers	Amusement, gloire, challenge, argent, ego
S2	Script kiddies	Curiosité, ego, argent, amusement
S3	Concurrent	Espionnage, réutilisation de contenu
S4	Cybercrime	Argent, destruction de données
S5	Accident humain	-
S6	Problèmes techniques	-

TABLE 5.29 – Sources de menaces des gestionnaires de mots de passe M5

Ci-dessous, se trouve un tableau avec différents scénarios d'attaques. Nous utilisons le modèle STRIDE afin de définir tous les types de menaces.

	Code	Code du bien	Scénario de menace	Type de menace	Source de menace
1	T3	A1	Récupération de données sensibles en clair sur le disque	<i>Information disclosure</i>	S1, S2, S4
2	T3	A1	Récupération du coffre-fort et déchiffrement dû à un algorithme trop simple	<i>Information disclosure</i>	S1, S2, S4
3	T3	A1, A2, A3	Récupération de données sensibles en mémoire	<i>Information Disclosure</i>	S1, S2, S4
4	T3	A3	Récupération de la clé de chiffrement en mémoire	<i>Information Disclosure</i>	S1, S2, S4
5	T3	A3	Brute-force de clé	<i>Information Disclosure</i>	S1, S2, S4

A1 = Coffre-fort | A2 = Master password | A3 = Clés

TABLE 5.30 – Scénarios et types d'attaques possibles sur M5

Identification des contrôles

Le contrôle qui est présent sur toutes les applications est que chaque utilisateur possède une clé de chiffrement différente, qui est dérivée avec son master password. Même si ce dernier est le même, la clé ne sera pas pareille dû au salt qui est soit aléatoire soit le username, qui est unique. Les algorithmes choisis sont forts (voir 4.1) et sont recommandés pour la dérivation des clés. Ainsi, la clé est protégée contre le brute-force ou les attaques par dictionnaire.

Néanmoins, il est quand même important d’avoir un master password fort ², ce qui n’est pas demandé dans tous les questionnaires de mots de passe lors de l’inscription de l’utilisateur.

Pour le chiffrement des données, les questionnaires de mots de passe utilisent AES-256. L’algorithme est également encore recommandé en 2022.

Au niveau de la protection mémoire, certains proposent une protection des données sensibles dans le processus (avec DPAPI ou Chacha20) et les données sont effacées sur le disque dès que le processus s’arrête. En se basant sur une étude[19], la mémoire est en général bien protégée lorsque le questionnaire dans l’état *Not Running*, cependant dès le moment où le questionnaire est dans l’état *Unlock* ou *Locked*, les secrets et le master password sont plus exposés.

Identification des vulnérabilités

Pour chaque actif identifié au préalable, nous allons analyser les vulnérabilités qui pourraient être présentes. Nous allons également ajouter la sévérité de la vulnérabilité.

Bien	Vulnérabilité	Sévérité de la vulnérabilité
A1	Utilisation d’algorithmes de chiffrement plus recommandés ou trop faibles	Haut
A1, A2, A3	Aucune protection de la mémoire du processus	Haut
A1, A2, A3	Option <i>remember me</i>	Haut
A1, A2, A3	Données non nettoyées de la mémoire lors de l’arrêt du processus	Haut
A3	Dérivation des clés trop simple	Moyen

A1 = Coffre-fort | A2 = Master password | A3 = Clés

TABLE 5.31 – Vulnérabilités présentes dans les questionnaires de mots de passe M5

Identifications des conséquences

Pour chaque bien défini plus haut, nous allons ajouter les conséquences qu’il pourrait y avoir dans le cas d’attaques, ainsi que ce que nous souhaitons garantir.

Bien	Conséquences	Garanties
Coffre-fort (A1)	Perte de réputation et d’image, pertes de données	Intégrité, confidentialité
Master password (A2)	Perte de réputation et d’image, perte de données	Confidentialité, intégrité des données
Clés (A3)	Perte de réputation et d’image, perte de données	Confidentialité, intégrité

TABLE 5.32 – Conséquences des menaces sur l’entreprise

2. Ce qu’on considère un mot de passe fort ; au moins une majuscule, une minuscule, un chiffre et 11 caractères

5.1.7.2 Analyse des risques

Ainsi, dans cette partie, nous allons analyser l'impact des conséquences ainsi que la probabilité que chaque scénario d'attaque identifié puisse se produire. Pour cela, nous allons utiliser plusieurs notations différentes ; pour les conséquences nous aurons les niveaux suivants :

Bien	Menace	Scénario	Impact des conséquences	Probabilité	Niveau de risque
A1	T3	Récupération des données sensibles en clair sur le disque	Important	Probable	Haut
A1	T3	Récupération du coffre-fort et déchiffrement dû à un algorithme trop simple	Important	Peu probable	Moyen
A1, A2, A3	T3	Récupération de données sensibles en mémoire	Important	Probable	Haut
A3	T3	Récupération de la clé de chiffrement en mémoire	Important	Probable	Haut
A3	T3	Brute-force de clé	Important	Rare	Moyen

A1 = Coffre-fort | A2 = Master password | A3 = Clés
T1 = Spoofing | T2 = Elevation of privileges | T3 = Information Disclosure | T4 = Denial of service | T5 = Equipment failure | T6 = Tampering

TABLE 5.33 – Analyse des risques de chaque scénario de menace sur M5

5.1.8 Évaluation des risques

Cette étape sert à évaluer si les risques sont acceptables ou s'ils ont besoin d'un traitement supplémentaires, c'est-à-dire une mitigation. Nous avons défini à l'étape précédente le niveau de risque de chaque scénario de menace. En résumé, nous avons :

- Bas : 2
- Moyen : 17
- Haut : 6

Ainsi, nous pouvons définir que tous les scénarios de menaces ayant un niveau de risque bas peuvent être acceptés par l'entreprise, néanmoins ces derniers ne doivent pas être oubliés et mis de côté. Pour les niveaux de risques moyen et haut, il est nécessaire d'appliquer une mitigation et de surveiller les menaces.

5.1.9 Traitement des risques

Dans cette section, nous allons définir les contre-mesures à entreprendre afin d'éviter au maximum les menaces que nous avons identifiées plus haut. Comme précisé dans le paragraphe de l'identification des contrôles, beaucoup de choses sont déjà mises en place par les constructeurs et sont efficaces contre les attaques. Néanmoins, nous pouvons ajouter quelques contre-mesures qui ne sont pas ou peu effectuées par les quelques candidats que nous avons sélectionnés au préalable. Ainsi, nous allons sélectionner les risques qui ont été qualifiés comme moyen ou haut et nous allons proposer des contre-mesures possibles.

Menace	Contre-mesures
Master password faible	- Ajouter des contrôles lors de l'inscription de l'utilisateur en lui demandant un mot de passe fort - Proposer un master password fort généré aléatoirement par l'application
Temps de session trop long	- Ajouter un temps d'expiration de session et verrouiller le gestionnaire après ce temps
Presse-papier non nettoyé	- Après un temps court, nettoyer le presse-papier de l'utilisateur
Brute-force	- Ajouter un nombre d'essais maximum - Ajouter une politique de master password forte
Phishing en clonant une page de connexion	- Appliquer un meilleur critère de match lors de l'auto-complétion des champs de connexion - Demander à l'utilisateur de valider l'auto-complétion - Sensibilisation auprès des utilisateurs
Clickjacking	- Bloquer l'utilisation de Javascript à l'aide d'API sécurisées
Communication non-chiffrée	- Ajouter HTTPS pour la communication avec le serveur - Éviter d'envoyer le master password ou des clés de chiffrement ou privée au serveur - Chiffrer toutes les données avant de les envoyer dans le cloud
Dérivation des clés trop simple	- Utiliser un algorithme recommandé et fort (PBKDF2-SHA2 par exemple) - Une clé de chiffrement unique par utilisateur pour éviter le brute force
Perte de données	- Si cloud-based / browser-based, effectuer des backups sur les serveurs chaque nuit - Mettre à disposition la fonctionnalité de backup pour les particuliers pour les faire en local - Exportation des données CSV et les stocker dans un endroit chiffrer et sécurisé
Manque de protection dans la mémoire	- Nettoyage de mémoire lorsque le gestionnaire est dans l'état locked - Protection des données sensible avec des APIs pour limiter l'exposition de secrets
Option remember me	- Désactivation de mode offline avec les gestionnaires cloud-based et browser-based
Manque de protection du disque	- Utiliser un chiffrement du disque entier

TABLE 5.34 – Contre-mesures possibles pour les gestionnaires

5.2 Exigences sécuritaires à respecter

Dans cette dernière section de l'analyse de menaces, nous allons établir une liste d'exigences sécuritaires que les gestionnaires de mots de passe doivent respecter afin de se définir sécurisé et afin de protéger les données sensibles au maximum. Nous allons premièrement identifier toutes les exigences générales puis nous allons les définir plus en détails afin d'expliquer ce que nous souhaitons garantir. Pour les identifier, nous allons nous référencer à notre analyse de menaces puis nous allons également nous baser sur le standard de OWASP *Application Security Verification Standard*[34].

Type	Description de l'exigence	Commentaire
Chiffrement	Le coffre-fort de l'utilisateur doit impérativement être chiffré avec un algorithme ainsi qu'une taille de clé qui sont recommandés	Nous pouvons nous baser sur les recommandations du report de ECRYPT[13]. Au mieux, l'algorithme devrait être encore recommandé pour une utilisation future.
	Le chiffrement doit être end-to-end pour assurer la protection de données lors du transport dans le cloud	Concerne les gestionnaires qui fonctionnent avec le cloud et font de la synchronisations de données
	Le chiffrement doit être fait côté client, donc en local	Afin d'éviter que des données claires soient stockés sur les serveurs
	L'architecture Zero-knowledge doit être respectée	Cette exigence est identique à la précédente
Clés de chiffrement	La communication entre l'utilisateur et les serveurs doit être chiffrée avec un algorithme recommandé	HTTTPS est la meilleure solution de chiffrement
	La dérivation de clés doit se baser sur des algorithmes Password-Based Key Derivation et doit être recommandé	Les plus courants sont Argon2 et PBKDF2 (associé à une fonction de hash)
	Le salt doit être unique	Si on utilise le username ou l'adresse e-mail, il faut absolument s'assurer que ce dernier soit unique lors de l'inscription de l'utilisateur
	Le salt doit être aléatoire (si on n'utilise pas le username)	Utiliser un pseudo-nombre généré aléatoirement pour s'assurer qu'il soit correctement aléatoire
Authentification	Les clés et la salt doivent être stockés localement dans la mémoire du processus	
	Les clés de chiffrement ne doivent jamais être envoyées aux serveurs et rester en local	Pour les gestionnaires qui utilisent le cloud
	Authentifier les données avec un schéma sûr, comme Encrypt-then-MAC	Nécessaire pour les gestionnaires qui ne fonctionnent qu'en local
	Authentifier les utilisateurs auprès des serveurs du constructeur	Pour les gestionnaires qui fonctionnent également avec le cloud
Master password	Hash d'authentification stocké dans la base de données des serveurs doit être chiffré	Hash utilisé pour le comparer avec celui que le client envoie afin d'approuver l'authentification
	Le master password doit respecter des critères lors de l'inscription de l'utilisateur	Pour que le master password soit fort et qu'on évite au maximum des attaques sur mots de passe
	Le master password ne devrait jamais être stocké en mémoire	À moins que des précautions de protection de mémoire aient été prises et qu'on ne puisse pas le voir en clair
	La fonctionnalité Remember me est fortement déconseillée	Dû au fait que le master password puisse être stocké en mémoire
	Demander obligatoirement le 2FA ou MFA à l'utilisateur lors de son inscription	Cela permet d'ajouter une couche sécuritaire

	Le master password devrait rester localement et ne devrait jamais être transmis aux serveurs	Pour les gestionnaires qui fonctionnent avec le cloud
Session	Un temps de session doit être ajouté obligatoirement	Le gestionnaire se place dans un état Locked après ce temps
Mémoire	Dans l'état Not Running, aucune donnée qui puisse faire compromettre la base de données sur le disque ne doit être stockée sur le disque	Par exemple, le master password ou une clé de chiffrement dans un fichier de configuration
	Dans l'état Unlocked, il devrait pas être possible d'extraire le master password de la mémoire	
	Dans l'état Unlocked, il ne devrait pas être possible d'extraire de la mémoire des éléments déchiffrés avec lesquels on n'a pas interagi	Une interaction peut être afficher, copier ou encore l'accès aux mots de passe
	Dans l'état Locked, il ne devrait pas être possible d'extraire n'importe quelle donnée sensible	Toutes les données doivent être effacées de la mémoire
Partage de données	Les clés privées ne devraient être accessibles qu'aux utilisateurs concernés	
	Les clés privées doivent rester en local et ne jamais être envoyé aux serveurs	
	Les clés publiques et les identités liées doivent être vérifiées	Afin de s'assurer que l'utilisateur a qui on demande sa clé est le bon
Backup	L'élément a partager et sa clé de chiffrement doivent être chiffrés localement	
	Les sauvegardes stockées sur les serveurs doivent être chiffrées	
	Le presse-papier doit être nettoyé après un court instant (10-40s), l'historique du presse-papier doit également être nettoyé	Pour éviter le sniffing de presse-papier
Attaques web-based	Pour l'auto-complétion des champs, il est nécessaire de demander l'autorisation à l'utilisateur pour remplir le champs	Cela peut faire éviter des attaques de phishing
	Les critères de matching pour les URLs lors de l'auto-complétion doivent être stricts	
	Éviter d'ignorer les sous-domaines de l'URL	Afin de s'assurer qu'il n'est pas possible de voler les identifiants du domaine parent
	Il faudrait faire la différence entre HTTP et HTTPS lors du remplissage automatique des champs	Pour éviter une attaque MITM et de remplir une page clônée HTTP d'un site initialement HTTPS
	Il faut bloquer l'exécution Javascript	

TABLE 5.35 – Exigences sécuritaires

Chapitre 6

Sélection des candidats

Dans ce chapitre, nous allons faire la sélection des gestionnaires de mots de passe que nous analyserons dans le chapitre suivant. Nous définirons les critères des sélections afin de correctement les choisir, puis nous ferons notre choix en fonction des candidats sélectionnés. Afin d'avoir une bonne vue d'ensemble sur les fonctionnalités et la sécurité de chaque application, nous allons reprendre les 9 gestionnaires analysés dans les chapitres précédents.

6.1 Critères de sélection

Cette section va ainsi présenter les critères de sélections pour les candidats avec lesquels on fera une analyse sécuritaire détaillée.

Open-source - un critère assez important car un gestionnaire de mots de passe open-source pourrait être plus sûr qu'un gestionnaire closed-source dû au fait qu'importe quel utilisateur peut auditer le code source indépendamment et peut reporter des failles aux constructeurs. Les applications closed-source comptent à 100% sur l'équipe de développement et pourrait faire face à plus d'attaques.

Types - nous allons sélectionner des gestionnaires des 3 types différents afin d'avoir un éventail complet de gestionnaires de mots de passe existants. Pour faire un rappel, les différents types sont : cloud-based, local-based et browser-based.

Gratuit - il est bien de prendre en compte ce critère afin d'analyser si un gestionnaire proposant des fonctionnalités gratuites est plus faible qu'un gestionnaire complètement payant. De plus, il pourrait être intéressant de comparer la sécurité d'un application qui propose des fonctionnalités gratuites et payantes.

Nombre d'utilisateurs - il est intéressant d'ajouter le critère de la popularité afin de se rendre compte de l'impact de quelconques vulnérabilités présentes sur l'application et de

constater malgré une grande popularité, s'il y a l'existence de failles non-corrigées ou non-prises en compte.

Partage de données - cette fonctionnalité est assez critique, dû au fait que des données sont partagées entre plusieurs utilisateurs, il est impératif que les données soient uniquement accessibles par les bonnes personnes de confiance. Ainsi, il serait intéressant de sélectionner au moins un candidat qui propose cette fonctionnalité pour évaluer la sécurité.

Choix cryptographiques - ce critère concerne les choix cryptographiques utilisés pour le chiffrement des données par exemple, ou la dérivation des clés. Il sera important d'évaluer si ces choix sont suffisant. Ainsi, nous allons sélectionner des candidats qui ont effectuer des choix cryptographiques différents.

Plateforme supportée - l'analyse sécuritaire se fera dans un premier temps sur un ordinateur avec Windows 11, donc les applications doivent être disponibles pour cet OS. La raison pour laquelle nous faisons ce choix est que c'est un des OS les plus populaires sur le marché.

Failles connues - pour quelques gestionnaires de mots de passe, des vulnérabilités ont été découvertes. Pour certaines, des corrections ont été faites aux applications mais certaines n'ont pas été corrigée malgré un report vers les constructeurs des gestionnaires. Ainsi, ceci est un aspect important à prendre en compte afin de constater si les failles ont été corrigées après leurs découvertes.

6.2 Choix

Afin d'avoir une bonne vue d'ensemble sur tous les candidats et de faire une sélection pertinente, nous allons établir un tableau récapitulatif. Lors du choix, nous allons tenter de couvrir tous les critères proposés avant, afin de faire une analyse complète et diverse sur tout ce qui existe sur le marché actuellement.

Type	Open-source?	Gratuit?	Nombre d'utilisateurs	Partage de données	Choix cryptographiques	Disponible sur Windows 11 ?	Faibles ou faiblesses connues ?
Dashlane	Non	Oui	15'000'000	Oui	AES 256 Argon2d / PBKDF2-SHA2	Oui	Oui
LastPass	Non	Oui	33'000'000	Oui	AES-CBC 256 Argon2d / PBKDF2-SHA2	Oui	Oui
KeePass	Oui	Oui	60'000'000	Non	AES-CBC 256 / Chacha20 AES-KDF / Argon2	Oui	Oui
1Password	Non	Non	15'000'000	Oui	AES-GCM 256 PBKDF2-SHA2	Oui	Oui
NordPass	Non	Oui	14'000'000	Oui	XChaCha20 Argon2id	Oui	-
Bitwarden	Oui	Oui	15-20'000'000	Oui	AES-CBC 256 PBKDF2-SHA2	Oui	Oui
Keeper	Non	Non	1'000'000	Oui	AES 256 PBKDF2	Oui	Oui
Padloc	Oui	Oui	-	Oui	AES-GCM 256 PBKDF2-SHA2	Oui	Oui
Firefox	Oui	Oui	-	Non	AES-CBC / 3DES-CBC PBKDF2-SHA2 / HKDF	Oui	Oui

FIGURE 6.1 – Comparatif des candidats pour la sélection

Nous allons donc choisir 3 candidats pour avoir une bonne vue d'ensemble sur tout ce qui existe.

Ainsi, nous allons sélectionner :

- LastPass
- KeePass
- Firefox

Chaque chapitre dédié aux gestionnaires de mots de passe sélectionnés se basera sur les exigences sécuritaires que nous avons définis au chapitre précédent et sur les vulnérabilités et faiblesses déjà connues.

À chaque début de chapitre, nous établirons les critères d'évaluation de la sécurité de l'application. Les critères seront à chaque fois différents en fonction du type ou des fonctionnalités proposées.

Chapitre 7

LastPass

Ce chapitre sera dédié à l'analyse sécuritaire de l'application LastPass. Nous allons utiliser l'extension de navigateur sur Google Chrome. Il existe également des applications desktop, cependant les versions ne sont pas proposées sur le site officiel car ils mettent en avant uniquement l'extension de navigateur. Nous n'allons donc pas nous concentrer sur l'application desktop et uniquement analyser l'extension de navigateur.

7.1 Environnement

Logiciel	Version
Windows 11	10.0.22621
Google Chrome	106.0.5249.119
LastPass Extension	4.102.1

TABLE 7.1 – Environnement utilisé pour LastPass

7.2 Critères d'analyse

Comme précisé précédemment, nous allons dans un premier temps définir tous les critères que nous souhaitons analyser afin d'évaluer la sécurité du gestionnaire LastPass. Ainsi, nous avons établi les critères suivants :

- Master password
- Choix cryptographiques
 - Chiffrement
 - Dérivation des clés

- Authentification
- Fonctionnalités proposées
 - Génération de mots de passe
 - Auto-complétion de champs
 - Presse-papier
- Stockage
- Mémoire

Pour chaque critère, nous analyserons sa sécurité et nous l'évaluerons afin d'indiquer s'il s'agit d'une fonctionnalité sûre ou d'une faiblesse qui aurait besoin d'être corrigé avec une contre-mesure.

Nous allons également citer quelconque faiblesse ou faille identifiée et connue pour chaque critère afin de se concentrer sur certains points lors de notre analyse et de constater si la faiblesse existe encore ou pas.

7.3 Master password

Lors de l'inscription de l'utilisateur, LastPass va demander d'entrer un master password afin de pouvoir créer le coffre-fort. Étant donné que LastPass dérive la clé de chiffrement à l'aide du username et du master password, il est important que ce dernier soit fort et aléatoire afin d'éviter des attaques par dictionnaire.

LastPass à ajouter les critères suivants afin de valider le master password entré par l'utilisateur :

- minimum de 12 caractères
- minimum de 1 chiffre
- minimum de 1 majuscule
- minimum de 1 minuscule
- pas l'adresse e-mail utilisée pour le username

Afin d'estimer la force du master password pour contrer une attaque de brute force, nous pouvons calculer la complexité d'un mot de passe avec les critères minimum pour réaliser s'ils sont suffisants. On sait que pour les 12 caractères exactement il y a 3 possibilités différentes qui sont des caractères alphanumériques sensibles à la casse. Donc 26 lettres minuscules, 26 lettres majuscules et 10 chiffres. Ainsi, nous pouvons déduire :

$$\begin{aligned}n &= 26 + 26 + 10 \\c &= n^{12} \\c &= 3.226266762 \times 10^{21}\end{aligned}$$

Nous constatons que le nombre de combinaisons est grand et que les critères semblent assez suffisant afin d'éviter des attaques.

En se basant sur l'étude de Hives Systems[43], avec une GPU récente et puissante (RTX 3090), il faudrait 200 ans à un attaquant pour brute force le master password. Ce qui est pour l'instant un nombre de temps acceptable mais il serait nécessaire d'y faire attention pour les mois à venir.

De plus, LastPass utilise l'algorithme PBKDF2-SHA2 avec 100'100 rounds pour dériver le master password et le stocker sur la base de données utilisateurs sur leurs serveurs. Ce dernier est premièrement dérivé avec l'adresse e-mail comme salt, puis une deuxième fois afin de créer un hash d'authentification. Il est stocké dans la base de données en utilisant un salt aléatoire (un différent par utilisateur) et en le hashant de nouveau avec PBKDF2-SHA2 100'100 rounds et Scrypt.

Ainsi, cela amène une couche sécuritaire non-négligeable qui protège un maximum le master password de l'utilisateur.

7.3.1 Entropie

Nous pouvons calculer l'entropie du master password minimum exigé par LastPass :

$$E = \log_2(62^{12})$$
$$E = 71.4503$$

D'après l'échelle que nous avons proposé dans la section 3.2.1, les mots de passe générés sont raisonnables, c'est-à-dire qu'ils sont acceptables, mais il serait nécessaire d'accompagner les master password d'une bonne implémentation de l'authentification de l'utilisateur.

7.3.2 Brute-force authentication

L'étude de 2020 de l'université de York[7], liste une faiblesse découverte lors de son analyse sur plusieurs gestionnaires de mots de passe, notamment LastPass. Il indique que LastPass a une partielle contre-mesure contre le brute-force sur les extensions de navigateur. Nous avons testé d'entrer plusieurs master passwords faux, dans un premier temps dans un mode en ligne et dans un second temps en mode hors-ligne.

En ligne

Lors d'environ 6 essais de master password faux, l'authentification est bloquée pendant 5 minutes et un e-mail est envoyé à l'utilisateur. Cette contre-mesure est efficace à contrer les attaques brute-force, comme les attaques par dictionnaire. Pour un attaquant, cela ferait à

peu près 72 essais par heure, ce qui prendrait énormément de temps pour brute-forcer. Ainsi, ces mesures ralentissent considérablement les attaques et peut les prévenir.

Hors-ligne

Lors de brute-force hors-ligne, il n'y a aucun nombre d'essais maximum. Ainsi, aucune protection n'a été ajoutée afin de contrer des attaques. Il serait nécessaire d'ajouter des contre-mesures afin d'éviter toute faille car pour une telle attaque, l'attaquant n'a pas besoin de privilèges ni de l'interaction de l'utilisateur. Néanmoins, l'attaque s'appuie, entre autres, sur la complexité du mot de passe. Ainsi, nous remarquons que la politique du master password proposée par LastPass permet de configurer des mots de passe fort, ce qui pourrait permettre de ralentir considérablement des attaques de brute-force.

7.4 Choix cryptographiques

7.4.1 Chiffrement

Pour le chiffrement de toutes les données du questionnaire de mots de passe, LastPass utilise AES-CBC 256. Cet algorithme est communément utilisé et est encore recommandé en 2022[13], cependant il est important d'utiliser un IV aléatoire et unique pour chaque élément chiffré afin d'éviter la réutilisation d'IV.

7.4.2 Dérivation des clés

Comme cité précédemment, pour la dérivation des clés afin de créer la clé de chiffrement et le hash d'authentification, LastPass utilise PBKDF2-SHA256 avec des rounds de 100'100 et 100'101 (par défaut, augmentable dans les paramètres de l'application). En se référant à Ecrypt, ils suggèrent un minimum de 40'000 rounds, donc suffisamment de rounds sont effectués sur LastPass. Le salt utilisé pour la dérivation du master password est le username, donc l'adresse e-mail de l'utilisateur. Pour bien, il serait mieux d'utiliser un salt généré aléatoirement avec un PRNG afin de s'assurer qu'il soit aléatoire et unique. Néanmoins, LastPass s'assure correctement que l'adresse e-mail est unique et n'est pas encore utilisée par un autre utilisateur, ainsi ils garantissent qu'aucune clé de chiffrement ne pourra être identique.

En 2022, il est recommandé d'utiliser Argon2 à la place de PBKDF2 pour les nouveaux systèmes car ce dernier n'est pas résistant à certaines attaques (GPU par exemple).[30]

Ainsi, il serait intéressant pour LastPass d'ajouter Argon2 comme algorithme supporté pour la création des clés.

7.4.3 Authentication

Pour authentifier un utilisateur, LastPass va générer un hash d'authentification en dérivant deux fois le master password avec PBKDF2-SHA2. Ce dernier sera par la suite stocké dans la base de données de l'utilisateur en l'hashant une fois de plus avec PBKDF2-SHA2 avec un salt aléatoire (généré par utilisateur). En plus, LastPass le hashe une fois de plus avec l'algorithme Scrypt, qui ajoute une protection supplémentaire aux attaques par dictionnaire et au brute force. Ces deux dernières manipulations sont effectuées du côté serveur.

7.5 Fonctionnalités proposées

Dans les sections suivantes, nous allons évaluer la sécurité dans quelques fonctionnalités proposées par LastPass.

7.5.1 Génération des mots de passe

LastPass offre la fonctionnalité de générer des mots de passe automatiquement lors de l'ajout d'une entrée mot de passe dans le gestionnaire. Il est important d'évaluer la force des mots de passe générés afin de s'assurer que leur sécurité est garantie.

LastPass propose la génération suivante :

Longueur supportée	Composition par défaut	Taille par défaut	Symboles proposés
1-99	[A-Za-z0-9]	12	*!#\$%&%@^

TABLE 7.2 – Fonctionnalités de LastPass pour la génération de mots de passe

Une option que LastPass ne demande pas par défaut lors de la génération d'un mot de passe est d'ajouter au minimum un caractère par set de caractères¹ mis à disposition. Ce qui pourrait faire baisser le taux d'aléatoire et générer des mots de passe moins forts.

Au niveau des des symboles proposés pour la génération, nous remarquons qu'il n'y a pas une grande variation de caractères proposés, ce qui pourrait amener moins d'aléatoire lors de la génération de mots de passe.

De plus, sur l'extension, il y a la possibilité de générer des mots de passe "facile à dire" ou "facile à lire" pour aider l'utilisateur à la mémorisation. Cette fonctionnalité pourrait également baisser la sécurité dans la génération de mots de passe si la longueur n'est pas suffisante (minimum 12).

1. Nous définissons un set de caractères par [A-Z] par exemple

Une étude réalisée en 2021[32] sur l'évaluation de la qualité de la génération de mots de passe, a testé l'aléatoire et la robustesse de la génération. Cette dernière a pris en compte les différents sets de caractères et la longueur du mot de passe. Ainsi, nous allons nous baser sur cette étude en effectuant des tests sur la génération de mots de passe sur LastPass. Pour ce faire, nous allons générer le pool de données suivant :

- lettres majuscules / minuscules (ll)
- lettres majuscules / minuscules + chiffres (lc)
- lettres majuscules / minuscules + symboles (ls)
- symboles + chiffres (sc)
- lettres majuscules / minuscules + symboles + chiffres (all)

Pour chaque composition, nous allons générer 10'000 mots de passe aléatoires et avec à chaque fois 12, 16 et 20 caractères. 12 caractères étant la valeur par défaut sur LastPass, nous avons décidé de commencer à cette valeur-ci pour évaluer sa robustesse.

Afin de récupérer tous les mots de passe générés par l'extension, nous avons créé un fichier Java avec Selenium, qui permet de créer des scripts pour automatiser le navigateur. Le script se trouve à cet endroit dans le rapport A.1².

Pour évaluer la qualité de l'aléatoire et la robustesse, nous allons utiliser un outil créé par Daniel Wheeler[47] qui vérifie la popularité d'un mot de passe selon plusieurs sources - des sets de mots de passe fuités, des noms communs, les mots communs sur Wikipédia. Le programme va également regarder des motifs communs, comme par exemple des dates, des séquences répétées ou du clavier. En fonction de tous ces facteurs, il va estimer le nombre d'essais qu'un attaquant aura besoin afin de trouver le mot de passe de l'utilisateur. Le résultat du nombre d'essais est présenté en 4 catégories d'attaques sur mots de passe différentes qui permettent de visualiser tous les résultats possibles :

- **Online Throttling** (100 essais/heure) est une attaque en ligne sur un service qui limite le nombre d'essais lors de l'authentification
- **Online No Throttling** (10/seconde) est une attaque en ligne où il n'y a pas de nombre d'essais maximum ou que l'attaquant a réussi à le contourner
- **Offline Slow Hashing** ($1e^4$ /seconde) est une attaque hors-ligne où les mots de passe ont des fonctions de hachage lentes (comme PBKDF2).
- **Offline Fast Hashing** ($1e^{10}$ /seconde) est une attaque hors-ligne où les mots de passe ont des fonctions de hachage rapides (comme MD5, SHA1).

2. Pour des raisons de place et de redondance, le script en entier n'a pas été ajouté, cependant pour chaque composition la logique est la même pour le code

Ainsi, nous avons utilisé l'outil pour Java³ afin d'évaluer la robustesse de tous les mots de passe générés dans l'étape précédente. Le code s'inspire grandement de l'outil original⁴, mais il y a quelques différences lors des estimations. Toutefois, nous avons choisi l'outil Java afin de pouvoir écrire un script qui teste tous les mots de passe générés.

Le scriptA.2 implémenté a pour but de récupérer les mots de passes les plus faiblement générés en fonction du résultat du nombre d'essais et du nombre de caractères. L'outil retourne le score du mot de passe en \log_{10} pour que les nombres ne soient pas trop élevés. Ainsi, l'étude propose cette échelle d'évaluation suivante en fonction du score du mot de passe :

1. < 3 (\log_{10}), mot de passe risqué et trop vite devinable
2. < 6 , très devinable et protégé contre les attaques en ligne throttled
3. < 8 , quelque peu devinable et protégé contre les attaques en ligne unthrottled
4. < 10 , suffisamment non-devinable et modérément protégé contre les attaques hors-ligne avec des fonctions lentes
5. ≥ 10 , non-devinable et forte protection contre les attaques hors-ligne avec des fonctions lentes

Pour les résultats A.1, nous allons parler de quelques scores intéressants. Néanmoins, en général les résultats sont plutôt bons. Très occasionnellement, le générateur produit des mots de passes faibles (environ 1 fois sur 400), ce qui peut être tout à fait normal pour un générateur vraiment aléatoire. Néanmoins, d'après nos tests, le plus petit score se situe au niveau 4/5, ce qui est tout de même un bon score et un mot de passe plutôt fort et aléatoire.

Pour les 3 longueurs de mots de passe différents, la composition où le nombre d'essais est le plus petit sont avec des symboles et des chiffres. Cette constatation est assez logique car son entropie est la plus basse de toutes les compositions.

Au niveau des attaques, pour 12 caractères, nous constatons que les mots de passes sont assez vulnérables aux attaques hors-ligne, car le temps d'essais est assez court. Pour 16 caractères, pour les mots de passe les plus faibles, ils sont toujours un peu vulnérables aux attaques hors-ligne avec des fonctions de hachage rapides. Finalement, dès 20 caractères, nous constatons que tous les mots de passe sont fortement protégés contre tous les types d'attaques.

7.5.2 Auto-complétion

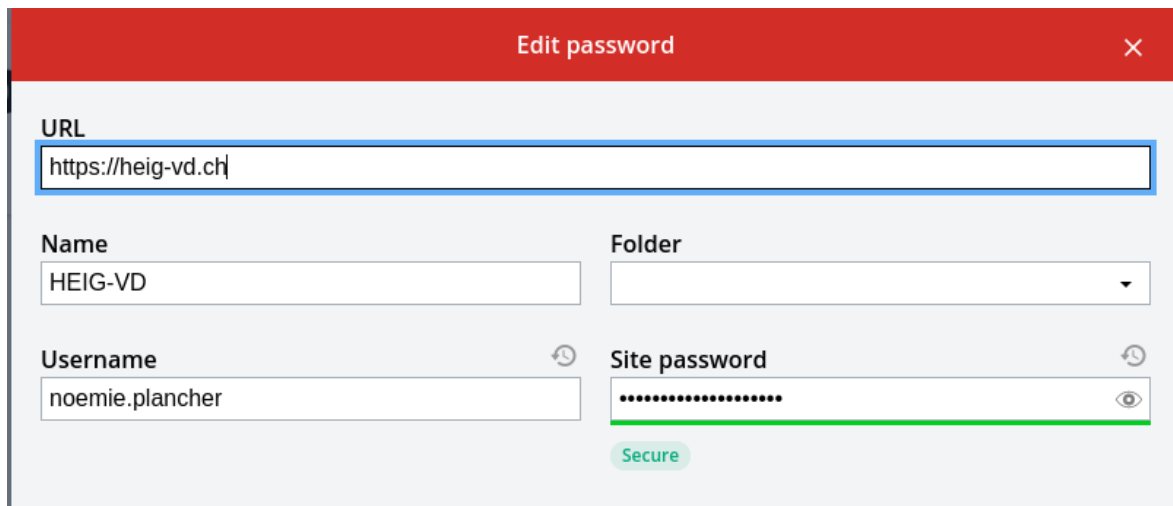
Afin de se référer à des faiblesses déjà connues de LastPass, nous allons nous appuyer sur une étude de 2013[27] et une plus récente de 2020[7] qui documentent des vulnérabilités découvertes sur des extensions de navigateur de gestionnaires de mots de passe, où LastPass a également été analysé. Une vulnérabilité qui est ressortie dans les deux études, est le fait

3. <https://github.com/nulab/zxcvbn4j>

4. <https://github.com/dropbox/zxcvbn>

que LastPass ignore les sous-domaines lors de la comparaison des origines, ce qui viole la *same-origin policy*.

Ainsi, nous avons testé si la faille existe encore en enregistrant des identifiants pour le nom de domaine parent `https://heig-vd.ch` :



The screenshot shows the 'Edit password' window in LastPass. The URL field is set to 'https://heig-vd.ch'. The Name field contains 'HEIG-VD', and the Folder field is empty. The Username field contains 'noemie.plancher', and the Site password field is masked with dots. A green 'Secure' indicator is visible below the password field.

FIGURE 7.1 – Identifiants pour heig-vd.ch enregistrés sur LastPass

Ensuite, nous avons testé de nous connecter à un sous-domaine avec `https://webmail.heig-vd.ch` :

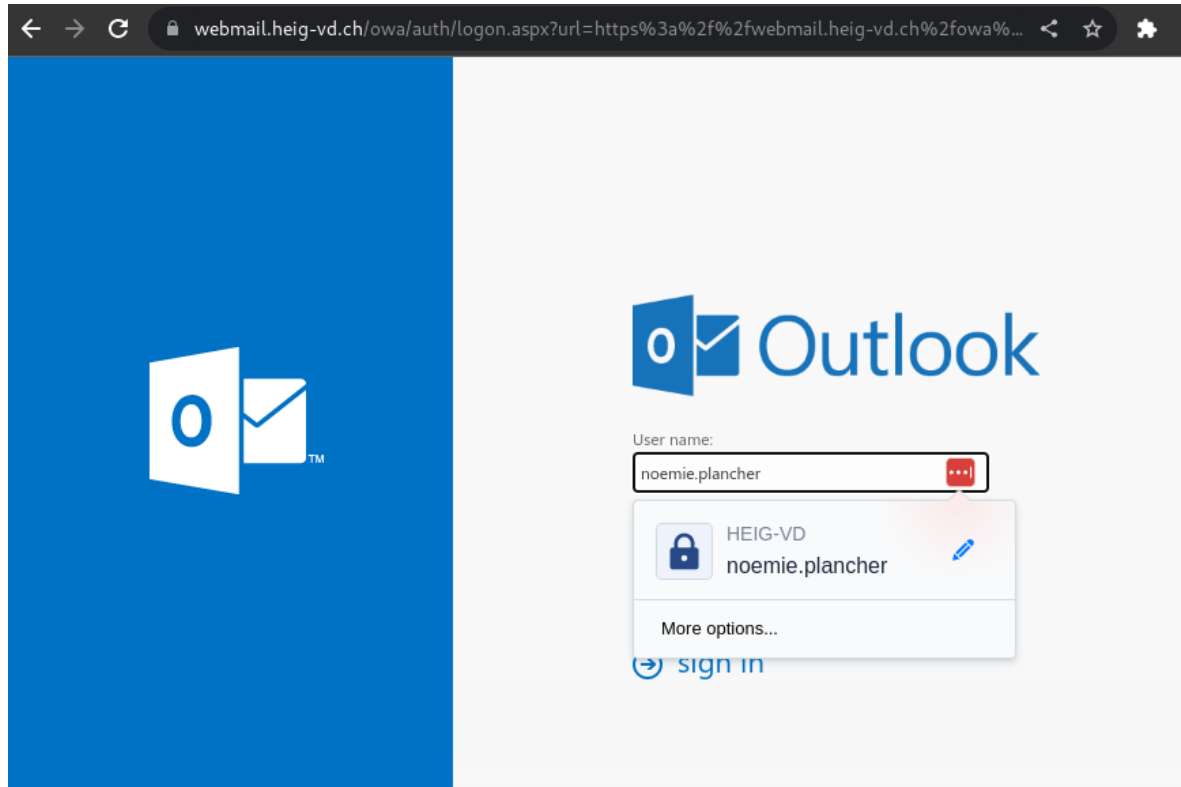


FIGURE 7.2 – Auto-complétion d'identifiants sur webmail.heig-vd.ch

Ainsi, nous constatons que LastPass propose les identifiants pour le sous-domaine car il les traite de la même manière.

Cette faiblesse pourrait être dangereuse car un attaquant pourrait potentiellement effectuer une attaque XSS en ajoutant un faux formulaire de connexion afin de récupérer les identifiants auto-complétés ou il pourrait utiliser l'HTML afin d'ajouter un formulaire de connexion sur le domaine client et ainsi voler des identifiants.

Néanmoins, sur LastPass, le résultat est aléatoire car la vulnérabilité n'apparaît pas pour chaque sous-domaine. En prenant notre exemple, avec le sous-domaine `https://gaps.heig-vd.ch`, les identifiants ne sont pas proposés :


← → ↻ 🔒 gaps.heig-vd.ch/consultation/ ⚙️ ☆ 📄 N

HEIG-VD HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD

GAPS

Bienvenue

Pour accéder aux différentes interfaces GAPS, il suffit de vous identifier ci-dessous avec votre compte Switch edu-ID :

Login with: 

HES-SO - University of Applied Sciences Western Switzerland

Continue

Mot de passe edu-ID oublié?

En cas de problème lors de l'identification, veuillez contacter le Helpdesk de votre institution.

Accès école (En cas de problèmes avec Switch edu-ID, compte local, utilisateur externe à la HES-SO ou chargés de cours)

Identifiant: (exemple: john.doe)

Mot de passe:

Entrer

Mot de passe HES-SO oublié ? | Mot de passe local oublié ?

En cas de problème **technique**, veuillez contacter l'équipe GAPS (gaps AT heig-vd POINT ch)

FIGURE 7.3 – Auto-complétion d'identifiants sur gaps.heig-vd.ch

Sinon, par défaut, LastPass auto-complète seulement les champs mais ne se connecte pas automatiquement (fonction modifiable dans les paramètres de l'entrée). Cela pourrait être critique car un attaquant pourrait effectuer une attaque XSS ou injection réseau et sans l'interaction de l'utilisateur, récupérer les identifiants qui ont été entrés dans le formulaire de connexion.

7.5.3 Presse-papier

La fonctionnalité de copier le mot de passe enregistré dans le presse-papier est souvent proposée lorsque le remplissage automatique de champs de connexion n'est pas proposée. Néanmoins, LastPass propose de copier le username, le mot de passe ou l'URL du site de connexion.

Par défaut, LastPass nettoie le presse-papier après 60 secondes. Ce temps est correct au niveau sécurité cependant, pour une meilleure protection contre le sniffing de presse-papier, ce temps pourrait être réduit à 30 secondes maximum. L'utilisateur a également la possibilité de décocher cette case et que LastPass ne nettoie plus le presse-papier. Il est certain qu'il

serait plus sécurisé d'enlever cette option afin d'éviter au maximum la perte ou le vol de données sensibles.

Toutefois, il est compliqué de se protéger à 100% de malwares de ce genre car à moins de ne pas permettre à l'utilisateur de copier ses mots de passe et d'effacer le presse-papier après un court instant, aucune protection n'est possible. Pour ce point, on ne peut que prévenir un maximum les utilisateurs qu'en leur exposant les risques et en leur demandant, par exemple, de verrouiller leur machine dès le moment où ils ne l'utilisent plus pour éviter qu'un attaquant sniffe leur presse-papier.

7.6 Stockage

LastPass fonctionne en ligne et hors-ligne[50], ainsi, il stocke des informations en local sur le disque de l'utilisateur afin d'avoir accès au coffre-fort et à toutes les données lorsque l'utilisateur n'est pas connecté à Internet. Tout dépend de l'OS et du navigateur, mais sur Windows 11 et Chrome, nous pouvons trouver une base de données SQLite à cet endroit : `%AppData%\Local\Google\Chrome\UserData\Default\databases\chrome-extension_hdokiejnpimakedhajhdcegplioahd_0`. Nous pouvons trouver plusieurs tables différentes, dont *LastPassData* où se trouvent les différentes entrées chiffrées du coffre-fort et le hash pour vérifier l'authenticité de la clé.

En se basant sur une démonstration d'attaque qui date de 2019[4], il explique comment la fonction *Remember Password* amène une vulnérabilité importante dans l'extension LastPass. Si la fonction de se rappeler le mot de passe et le mode hors-ligne sont activés, l'attaque peut réussir à récupérer le master password et le déchiffrer. Comme cité précédemment, une base de données est stockée en local chez l'utilisateur, ainsi c'est la table *LastPassSavedLogins2* qui nous intéresse. Elle contient les champs suivants : `username`, `password`, `last_login`, `protected`. En analysant le code Javascript `server.js`, qui se trouve dans le fichier `background.html` de l'extension, il est possible de voir comment est déchiffré le master password et ainsi écrire un script pour récupérer ce dernier en clair.

Toutefois, cette attaque n'a pas pu être effectuée sur la version actuelle de l'extension car plus aucune information n'est stockée dans la table *LastPassSavedLogins2*, malgré toutes les conditions remplies.

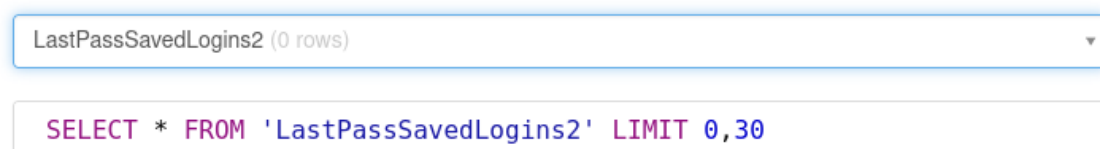


FIGURE 7.4 – Table *LastPassSavedLogins2* vide

Nous avons analysé en profondeur le code disponible de l'extension et nous n'avons pas trouvé de quelconque faiblesse qui nous permettrait de récupérer le master password enregistré. Les données qui transitent en local ont l'air d'être protégées en binaire, sans doute afin d'ajouter une couche de sécurité supplémentaire.

Dès le moment où le master password n'est pas facilement récupérable et mis directement disponible pour les attaquants, cela permet de ralentir les personnes malveillantes et il est ainsi moins problématique d'avoir des données chiffrées en local.

7.7 Mémoire

D'après le whitepaper de LastPass, ils utilisent les *Windows Crypto APIs*, mises à disposition par Windows, afin d'ajouter une couche supplémentaire de sécurité sur les appareils Windows. Ainsi, un test a été fait pour vérifier que le master password ou des informations sensibles, comme des mots de passe avec lesquels l'utilisateur aurait interagité, ne soient pas laissés en mémoire RAM lorsque le gestionnaire de mot de passe est verrouillé.

Ainsi, nous avons effectué un dump de la mémoire du processus sur le disque et dû au fait que la mémoire est partiellement gérée par Windows avec ses APIs cryptographiques, il y a beaucoup de données qui sont chiffrées et illisibles. Néanmoins, avec un peu de recherche à l'aide de mes usernames que je connais, nous avons pu trouver tous les mots de passe du coffre-fort en clair dans la mémoire du processus. Nous pouvons le constater ci-dessous avec une entrée pour tumblr en recherchant le string "passworddec" sur le dump :

```
1BC40309F440 39 31 34 38 35 35 38 30 39 37 31 36 39 22 0B 75 9148558097169".u
1BC40309F450 73 65 72 6E 61 6D 65 64 65 63 22 06 4E 6F 65 6D sernamedec".Noem
1BC40309F460 69 65 22 0B 70 61 73 73 77 6F 72 64 64 65 63 22 ie".passworddec"
1BC40309F470 0F 4C 61 63 72 79 70 74 30 63 72 69 67 30 6C 30 .Lacrypt0crig0l0
1BC40309F480 22 0E 70 61 73 73 77 6F 72 64 64 65 63 66 69 78 ".passworddecfix
1BC40309F490 22 0F 6C 61 63 72 79 70 74 30 63 72 69 67 30 6C ".lacrypt0crig0l
1BC40309F4A0 30 22 0E 72 65 61 6C 64 6F 6D 61 69 6E 32 6C 76 0".realdomain2lv
1BC40309F4B0 6C 22 0A 74 75 6D 62 6C 72 2E 63 6F 6D 22 0A 64 l".tumblr.com".d
1BC40309F4C0 6F 6D 61 69 6E 32 6C 76 6C 22 0A 74 75 6D 62 6C omain2lv1".tumbl
1BC40309F4D0 72 2E 63 6F 6D 22 05 67 65 6E 70 77 49 00 22 04 r.com".genpwI".
```

FIGURE 7.5 – Entrée du coffre-fort LastPass en clair

Cela veut donc signifier que LastPass charge tout le coffre-fort en mémoire dès que l'utilisateur se connecte et dès qu'elle est verrouillée, aucun nettoyage de la mémoire du processus n'est effectué. Toutefois, dès que le processus de l'application est stoppé, aucune donnée n'est visible en RAM.

Nous avons également effectué un autre test qui se base sur plusieurs attaques effectuées sur

LastPass d'une étude de 2013[42] et sur le travail de Bachelor de Léo Corthès[10]⁵. L'étude explique que l'extension utilise du javascript pour toutes les fonctionnalités, en incluant les opérations cryptographiques. Ainsi, la master key, qui est dérivée du master password et qui permet de chiffrer tout le coffre-fort, peut facilement se trouver dans le javascript sous le nom de variable `g_local_key`. Étant donné qu'on analyse une extension de navigateur, il est très commun d'avoir un fichier `background.html` qui permet d'invoquer tous les scripts qui tournent en arrière-plan. De ce fait, en utilisant les outils de développeur sur Chrome, nous avons pu debugger l'extension et chercher la master key pour afficher sa valeur.

Nous avons décidé d'effectuer ce test⁶ lorsque le gestionnaire était verrouillé afin de simuler correctement une attaque possible.

Premièrement, comme décrit dans le paragraphe plus haut, nous avons récupéré la valeur de `g_local_key`, qui représente la master key, à l'aide des outils de développeur de Chrome.

```
> g_local_key
< '\x8Cf\x1B1\x96;YsB{;Y64{ZsÅà\x1E|8+6yPÉUÄri'
> bin2hex(g_local_key)
< '8cce0a1bee963b5924d07b3ba5f4345b5a24c2e01e7c382b26ff50cb55c472ec'
```

FIGURE 7.6 – Master key en mémoire

Étant donné, que nous connaissons la valeur du master password (car il s'agit de mon compte) et que nous savons exactement comment LastPass dérive le master password (voir 4.1.3), nous pouvons tester si la clé est réellement la master key :

```
1 >>> masterkey = hashlib.pbkdf2_hmac('sha256', b'Followthewhiterabbit98', b'
   noemie.plancherel@gmail.com', 100100, 32)
2 >>> binascii.hexlify(masterkey)
3
4 b'8cce0a1bee963b5924d07b3ba5f4345b5a24c2e01e7c382b26ff50cb55c472ec'
```

Listing 7.1 – Calcul de la master key sur LastPass

Ainsi, nous constatons que les deux clés sont similaires, c'est-à-dire que la variable `g_local_key` est réellement la master key.

Avec cette information, nous pouvons à présent déchiffrer tout le coffre-fort. Comme précisé dans la section précédente, une base de données stocke tout le coffre-fort sur le disque de l'utilisateur. Pour cette partie, nous nous intéresserons à la table *LastPassData* et au champ *accts*.

5. Le travail étant confidentiel, il n'est pas possible de le visualiser, toutefois l'attaque expliquée s'appuie sur une étude de 2019[4]

6. Les informations qui vont suivre ne seront pas censurées car il s'agit de valeurs d'exemples et non des valeurs réelles

LastPassData (7 rows)			
SELECT * FROM 'LastPassData' LIMIT 0,30			Execute
id	username_hash	type	data
17	c93b966b0a96b5eec320260225a367c24ec25e09ff27c1407ceffa786...	bigicons	lp676f6f676c652e636fd:9104:VBORw0KGgoAAAANSUHEUgAAAL...
18	c93b966b0a96b5eec320260225a367c24ec25e09ff27c1407ceffa786...	sqicons	lp676f6f676c652e636fd:11108:VBORw0KGgoAAAANSUHEUgAAA...
22	c93b966b0a96b5eec320260225a367c24ec25e09ff27c1407ceffa786...	key	W4z9fiXqRZeEr48CyHkSwHJOpn8XM6bhV9mQPS0+NrzRZ6rt3zo...
23	c93b966b0a96b5eec320260225a367c24ec25e09ff27c1407ceffa786...	otp	57ece05b13bfa6aaa4af664f57ba2idd
24	c93b966b0a96b5eec320260225a367c24ec25e09ff27c1407ceffa786...	accts	iterations=100100;TFBBVgAAAAI1MEFUVIIAAAABMEVOQ1UAAAA...
25	c93b966b0a96b5eec320260225a367c24ec25e09ff27c1407ceffa786...	icons	lp209970029139377256.gif:472:R0IGODIhEAAQAKUAAESG9DSqV...
26	c93b966b0a96b5eec320260225a367c24ec25e09ff27c1407ceffa786...	rsakey	42EB63175434C6FB22CC6B3839CB73762ED49C01E17C0C9765A...

FIGURE 7.7 – Master key en mémoire

Nous pouvons remarquer que le champ commence par le nombre d'itérations nécessaires pour dériver la master key et des données encodées en Base64 suivent.

Les données en base64 représentent toutes les entrées du coffre-fort. L'étude de Derk Barten décrit la structure de chaque entrée qui possède des caractères de délimitations pour indiquer l'URL du site en clair, le username chiffré et le mot de passe chiffré.

Un script A.3 a été développé, inspiré par le celui proposé par Léo Corthès, afin de déchiffrer toutes les entrées du coffre-fort à l'aide de la master key récupérée à l'étape précédente.

Toutefois, ce script a quelques limitations, il fonctionne uniquement pour les entrées basiques. LastPass propose plusieurs champs lors de l'enregistrement d'un mot de passe (username, mot de passe, URL, titre, notes ou encore un dossier). Ainsi, il est compliqué de savoir quelles entrées ont quels champs configurés et étant donné que toute la structure stockée est impactée, il n'est pas possible de déchiffrer exactement tous les champs de chaque entrée, néanmoins un maximum d'informations peuvent être récupérées sur le script, ce qui permet déjà à un attaquant d'avoir accès à quelques identifiants.

Ces deux attaques de mémoire ont besoin de certaines conditions afin de pouvoir les exécuter sur les victimes.

Attaque #1 sur la mémoire du processus

- Le gestionnaire de mot de passe doit être dans un état *Locked* ou *Unlocked*

Attaque #2 sur la mémoire du processus

- Le gestionnaire de mot de passe doit être dans un état *Unlocked*
- Le gestionnaire doit avoir le mode offline activé (par défaut il est activé)
- Le 2FA ou MFA ne doivent pas être configurés

7.8 Récapitulatif de l'analyse

Dans cette dernière section, nous allons établir un récapitulatif de toute l'analyse en passant par tous les points abordés, en ajoutant une évaluation de sécurité et en conseillant des contre-mesures nécessaires dans le cas où la faille est importante.

Critère analysé	Commentaire	Contre-mesures nécessaires ?
Master password	Les critères du master password sont forts, cela permet de ralentir considérablement les attaques de brute-force sur le master password, cependant, l'entropie indique un mot de passe moyen, ainsi il faudrait y faire attention pour les mois à venir. Lorsque le gestionnaire est en mode hors-ligne, il n'y a aucun nombre maximum d'essais d'authentification, cela pourrait aider un attaquant à effectuer du brute-force, néanmoins étant donné que la politique des master password de LastPass créée des mots de passe forts, l'attaque prendrait du temps	Il serait également bien d'ajouter un nombre d'essais maximum de connexion, aussi lorsque le gestionnaire de mots de passe est hors-ligne, afin d'éviter au maximum toute attaque de brute-force
Chiffrement	L'algorithme utilisé pour le chiffrement du coffre-fort est fort	-
Dérivation des clés	PBKDF2-SHA2 est efficace pour la dérivation des clés. 100'100 rounds est également un nombre assez grand pour permettre de ralentir au mieux les attaques. Il serait bien d'ajouter un support pour Argon2 afin de pouvoir proposer les deux	-
Authentification	L'authentification auprès des serveurs est implémentée de sorte à ce qu'il soit difficile de se faire passer par l'utilisateur sans connaître son master password	-
Génération de mots de passe	Les résultats sont plutôt bons ; des mots de passes faibles sont très occasionnellement générés et ils ont de bons score au niveau du nombre d'essais pour les deviner.	Étant donné que par défaut, LastPass génère des mots de passe de 12 caractères, il serait bien de l'augmenter d'au moins 14 caractères. De plus, il serait bien d'ajouter des symboles en plus de ceux proposés afin de garantir un très bon aléatoire
Auto-complétion	Nous avons pu constater que des failles de 2013 sont toujours présentes sur LastPass, notamment en ignorant les sous-domaines lors de l'auto-complétion des champs. L'option auto-remplissage des champs peut également être dangereuse et être vulnérable à des attaques où un attaquant ajouterait un formulaire de connexion caché.	Une meilleure comparaison d'origine lors de l'auto-remplissage des champs de formulaire serait nécessaire pour éviter toutes attaques. De plus, l'option de auto-remplissage des champs étant activée par défaut lors d'ajout d'une entrée dans le coffre-fort, il serait mieux de la désactiver et de laisser le choix à l'utilisateur ou d'avoir une interaction avec ce dernier avant de remplir le champ pour qu'il puisse valider

	<p>Le timeout ajouté lors de la copie d'identifiants est correct. Peut-être que baisser ce temps à 30-40 secondes serait une meilleure solution, cependant il est compliqué de se protéger à 100% des attaques de sniffing, mais LastPass ne peut rien faire de plus.</p>	-
Stockage	<p>La base de données stockée local ne contient aucune information en clair qui serait facile de récupérer directement. La fonction de se rappeler du master password est sécurisée car il n'y a pas la possibilité de récupérer le master password directement dans une table de la base de données.</p>	<p>En liaison avec les failles découvertes dans la mémoire, il serait nécessaire de désactiver le mode offline par défaut afin d'éviter d'avoir des fichiers en local chez l'utilisateur. Cependant, sur LastPass afin de pouvoir désactiver ce mode, il faut avoir le 2FA ou MFA activé sur le gestionnaires. Ainsi, il faudrait également dissocier ces deux options.</p>
Mémoire	<p>Nous avons découvert deux failles assez critiques liées à la mémoire et également au stockage de la base de données en local. Premièrement, même quand le gestionnaire de mot de passe est verrouillé, on a la possibilité de récupérer tous les identifiants, avec mots de passe, en clair dans la mémoire du processus car LastPass semble charger tout le coffre-fort en mémoire et l'efface quand le processus est terminé. La seconde faille permet de récupérer la master key depuis les outils de développeurs de Chrome et de déchiffrer une grande partie du coffre-fort grâce aux données stockées en local.</p>	<p>La première faille est très critique, étant donné qu'un attaquant pourrait effectuer un dump de la mémoire assez facilement, même quand le gestionnaire de mots de passe est verrouillé et il n'a pas besoin de privilèges pour effectuer l'attaque. La sécurité à ajouter serait dans un premier temps de s'assurer de nettoyer la mémoire dès que le gestionnaire est verrouillé, puis s'assurer que les données sensibles, tels que les mots de passe, soient chiffrés dans la mémoire du processus, pour cela les OS mettent à disposition des APIs de cryptographie. De plus, un timeout de session pour verrouiller le gestionnaire après un certain temps serait une très bonne couche sécuritaire en plus. Pour la seconde attaque, la première contre-mesure à prendre serait de désactiver par défaut le mode offline. Puis au niveau des outils de développeurs de Chrome, il est assez difficile de gérer cette partie car cela touche à tout l'arrière-plan de l'extension qui est nécessaire quant à l'utilisation du gestionnaire.</p>
Divers	<p>Nous pouvons ajouter un commentaire sur un élément que nous n'avons pas analysé mais qui est très important à commenter ; par défaut, l'extension ne se verrouille jamais, même quand le processus se termine. C'est à l'utilisateur de se déconnecter manuellement. Cela pourrait être critique de ne jamais stopper la session, car un attaquant qui a accès au device déverrouillé de l'utilisateur, aura également accès au coffre-fort et à toutes les données en clair. Afin d'éviter au maximum un vol de données, il serait recommandé d'ajouter un timeout de session par défaut (car actuellement il est possible de le configurer dans les paramètres de LastPass).</p>	

■ Bonne sécurité, aucun contre-mesure à ajouter ■ Sécurité moyenne, points à faire attention et quelques recommandations au niveau des contre-mesures
 ■ Sécurité faible, contre-mesures à prendre ■ Sécurité très faible, contre-mesures à prendre rapidement car failles critiques

Chapitre 8

KeePass

Ce chapitre sera dédié à l'analyse sécuritaire de l'application KeePass. Nous allons utiliser l'application desktop sur Windows 11. KeePass fonctionne entièrement en local et aucune donnée ne transite sur le cloud.

Nous voulons préciser que KeePass est open-source, ce qui va nous permettre d'analyser le code en cas de vulnérabilités trouvées. De plus, cela nous permettra de proposer une correction du code afin de contribuer au gestionnaire de mots de passe.

8.1 Environnement

Logiciel	Version
Windows 11	10.0.22621
KeePass	2.52

TABLE 8.1 – Environnement utilisé pour KeePass

Pour cette analyse, nous allons utiliser KeePass sans plugin supplémentaire, afin d'analyser l'application basique sans aucun ajout. Nous procédons de cette manière-ci car il existe beaucoup de plugins différents qui permettent d'ajouter des fonctionnalités différentes, comme des backups, une couche sécuritaire supplémentaire, de l'export, etc¹. Cependant, chaque plugin est développé par des personnes externes à KeePass, ainsi il est préférable de ne pas les inclure dans le rapport.

De plus, KeePass propose deux versions de son application, une 1.x et une 2.x. La première propose beaucoup moins de fonctionnalités et est moins flexible que la deuxième, c'est pour

1. <https://keepass.info/plugins.html>

cela que nous nous basons sur la deuxième.

8.2 Critères d'analyse

L'établissement des critères d'analyse se basent sur l'analyse de menaces effectuée plus tôt dans le travail et sur les failles connues de KeePass afin qu'on puisse se baser dessus.

- Master password
- Choix cryptographiques
 - Chiffrement
 - Dérivation des clés
 - Authentification
- Fonctionnalités proposées
 - Génération de mots de passe
 - Presse-papier
- Stockage
- Mémoire

8.3 Master password

Quand l'utilisateur crée sa nouvelle base de données, où toutes les données du coffre-fort seront stockées dessus, KeePass demande de fournir un master password qui permettra d'y dériver la master de key et de chiffrer le coffre-fort.

Il n'y a pas de critères exigés par KeePass lors de la spécification, ainsi l'utilisateur pourrait ajouter un seul caractère et le master password serait validé. Néanmoins, KeePass indique qu'il s'agit d'un mot de passe faible, ce qui pourrait permettre d'éviter des master password trop faibles qui permettrait d'effectuer une attaque de brute-force :

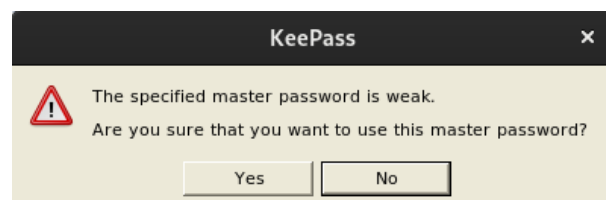


FIGURE 8.1 – Avertissement de master password faible de KeePass

Le gestionnaire indique également la qualité du mot de passe en indiquant le nombre de bits d'entropie et la force du mot de passe.

Il existe également des "options d'expert" qui permettent d'ajouter deux facteurs supplémentaires lors de la connexion. L'utilisateur peut fournir un *key file* qui est un fichier qui contient une clé. Il est possible d'ajouter le compte utilisateur Windows (si on est sur cette OS), ce qui permet de faire la base de données dépendante à l'utilisateur connecté. De ce fait, il sera uniquement possible d'ouvrir le coffre-fort si on est connecté avec cet utilisateur.

Pour chaque facteur ajouté, KeePass avertit **explicitement** que si on perd accès au facteur ou qu'on l'oublie, il est impossible de récupérer la base de données. Ainsi, ils conseillent de faire des backups pour le *key file* et que leur emplacement soit différent que celui du coffre-fort.

Donc, il n'existe aucun critère minimal exigé lors de la création du coffre-fort. Cependant, dans une entreprise par exemple, les administrateurs ont la possibilité de modifier le fichier de configuration et d'ajouter une taille et la qualité du master password minimum.

Un point très important à citer est qu'il est possible de ne pas configurer de master password, mais d'uniquement paramétrer soit un *key file* soit le *Windows Account User* soit les deux. Ainsi, cela pourrait être critique car un attaquant peu récupérer facilement le fichier avec la clé si ce dernier n'est pas dans un emplacement sécurisé et pour l'option de l'utilisateur Windows, si ce dernier est connecté sur la session de l'utilisateur, il pourra déverrouiller systématiquement le coffre-fort.

Bien que KeePass avertisse l'utilisateur qu'il a un master password trop faible, il serait recommandé d'ajouter une exigence quant à la qualité du mot de passe afin de s'assurer qu'il est fort. Car un particulier qui n'a pas ou peu de connaissances en sécurité informatique, il ne connaîtra pas les vulnérabilités liées à un mot de passe faible. Néanmoins, le fait que le gestionnaire est en uniquement en local, cela permet de freiner des attaques.

Il serait également recommandé de demander de toute manière un master password à l'utilisateur et de lui permettre d'ajouter des facteurs supplémentaires (comme le *key file* par exemple).

8.3.1 Brute-force authentication

Lorsque l'utilisateur doit entrer le master password de son coffre-fort, étant donné que KeePass fonctionne en local, un attaquant aurait la possibilité d'effectuer du brute-force du master password en testant toutes les possibilités possibles.

En testant plusieurs entrées incorrectes, nous avons remarqué que KeePass bloque le formulaire après 3 essais faux et ainsi afin de pouvoir retenter d'entrer le master password, il est nécessaire de ré-ouvrir la base de données ou de redémarrer l'application.

Cette méthode ne stoppe pas les attaquants mais permet de les freiner considérablement car

des manipulations sont nécessaires après seulement 3 essais.

8.4 Choix cryptographiques

8.4.1 Chiffrement

La version 2.x de KeePass propose deux algorithmes de chiffrement ; AES-CBC 256 et ChaCha20 256. Ces deux algorithmes sont recommandés et sont jugés comme très sécurisés[13].

L'IV utilisé pour chiffrer le coffre-fort est généré aléatoirement et stocké dans le header du fichier de la base de données. L'aléatoire permet ainsi de chiffrer plusieurs coffres-forts avec la même master key. Pour l'initialiser, KeePass utilise un CPRNG (cryptographically secure pseudorandom number generator)². Chaque générateur a besoin d'une source d'entropie pour l'initialiser, ainsi le gestionnaire crée une piscine d'entropie de différentes sources (notamment des nombres aléatoires générés par le fournisseur cryptographique du système, la date/heure actuelle et l'heure de mise en service, la position du curseur, etc.) afin d'avoir la meilleure entropie possible.

De plus, à chaque fois que la base de données est sauvegardée, c'est-à-dire si l'utilisateur a ajouté, modifié ou supprimé des entrées, un nouvel IV aléatoire est généré.

8.4.2 Dérivation des clés

Pour la dérivation de la master key, nous nous référons au schéma réalisé et expliqué plus tôt dans le travail 4.4. Nous remarquons que premièrement chaque facteur de connexion est hashé avec SHA-256 afin de créer une entrée (des données) pour la fonction de dérivation. La fonction de hachage SHA-256 est considérée comme sécurisée et recommandée dans le futur par ECRYPT. KeePass supporte deux algorithmes de dérivation de clés :

AES-KDF : par défaut, KeePass propose 60'000 itérations, qui sont modifiables par l'utilisateur. En sachant que plus il y a d'itérations, plus les attaques par dictionnaires sont complexes, ce nombre est suffisant.

Argon2 : le gestionnaire propose deux variantes de cet algorithme : Argon2d et Argon2id. Le premier protège mieux contre les attaques GPU, alors que le deuxième protège mieux contre les *side-channel attacks*³. Cette fonction est plus recommandée que AES-KDF car elle offre une meilleure résistance contre les attaques GPU et c'est un algorithme moderne (2015).

Puis la sortie des fonctions de dérivation est ensuite concaténée avec un sel aléatoire puis

2. Algorithme déterministe qui vise à générer une séquence de nombre imprévisibles pour un adversaire

3. Attaque qui recherche et exploite des failles dans l'implémentation, logicielle ou matérielle du système

compressée avec SHA-256 afin de créer la clé finale, la master key.

Cette méthode permet d'amener une grande protection contre les attaques par dictionnaire et les attaques par guessing. Plus les paramètres des fonctions de dérivation ont des valeurs élevées, plus la clé sera difficile et prendra du temps à deviner pour un attaquant.

8.4.3 Authentification

Comme expliqué dans la section 4.1.2, KeePass authentifie le header et les données du coffre-fort[38]. Le header authentifié à l'aide de HMAC-SHA-256 et est stocké directement après le header. Cela permet de vérifier que le fichier n'a pas été corrompu. De plus, un hash SHA-256 est stocké dans la base de données afin de vérifier que le header n'a pas été involontairement corrompu, sans connaître la master key.

Les données du coffre-fort sont authentifiées à l'aide d'un hash HMAC-SHA-256. KeePass sépare toutes les données du coffre-fort en blocs de 1 MB, ainsi chaque bloc chiffré est authentifié. Le schéma utilisé est *Encrypt-then-MAC*, qui est un schéma recommandé et qui permet de garantir l'intégrité des données claires et chiffrées. Le hash du bloc est généré avec les données chiffrées du bloc et une clé qui est différente pour chaque bloc, afin d'éviter toute réutilisation de clé. La clé pour HMAC est générée à l'aide de : $K_i := \text{SHA} - 512(iK)$ où K est une clé de 512 bits dérivée de la master key et du master seed (stocké dans le header du fichier KDBX).

8.5 Fonctionnalités proposées

8.5.1 Génération de mots de passe

Comme pour l'analyse de LastPass, nous allons également analyser la fonctionnalité de génération de mots de passe de KeePass[39]. Nous allons juger la force des mots de passe générés afin de garantir une bonne sécurité des entrées ajoutées par l'utilisateur.

KeePass propose la génération suivante :

Longueur supportée	Composition par défaut	Taille par défaut	Symboles proposés
> 1	[A-Za-z0-9]	20	!"#\$%&'()*+,-./:;<=>?@[\\]^_`{ }~

TABLE 8.2 – Fonctionnalités de KeePass pour la génération de mots de passe

Par défaut, KeePass ajoute au minimum un caractère par set de caractères choisi, ce qui permet de générer un mot de passe plus aléatoire. En plus de tous les sets de caractères proposés, il est possible d'inclure des caractères supplémentaires pour s'assurer qu'ils soient

présents dans la génération de mots de passe par exemple. Presque tous les caractères Unicode sont supportés par KeePass⁴. Cela permet d'avoir une grande variation de caractères différents, ce qui est un bon point pour la génération et l'aléatoire des mots de passe.

D'autres modes de générations sont également disponibles (comme l'ajout de règles de génération), mais nous ne nous concentrerons pas dessus pour cette partie.

Ainsi, nous allons effectuer les mêmes tests qu'avec LastPass en reprenant les mêmes compositions de mots de passe afin de pouvoir faire une comparaison entre les deux gestionnaires. Le pool de données suivant ont été généré :

- lettres majuscules / minuscules (ll)
- lettres majuscules / minuscules + chiffres (lc)
- lettres majuscules / minuscules + symboles (ls)
- symboles + chiffres (sc)
- lettres majuscules / minuscules + symboles + chiffres (all)

Nous avons également généré 10'000 mots de passe aléatoire par composition avec 12, 16 et 20 caractères pour évaluer leur robustesse.

Afin de récupérer tous les mots de passe générés par KeePass, nous avons utilisé un plugin, proposé par KeePass. L'outil *KPScript*⁵ permet d'effectuer du scripting avec KeePass. Il existe la fonctionnalité de *single command* où il est possible d'utiliser le plugin comme commande et où la fonctionnalité de génération de mots de passe est disponible. Le point avantageux est qu'il existe un paramètre *count* qui permet de générer *n* mots de passe. Le second paramètre *profile* permet de définir des profils de génération de mots de passe, par exemple, 12 caractères avec des majuscules et minuscules.

Voici la commande utilisée pour générer 10'000 mots de passe avec une composition all :

```
1 $ KPScript -c:GenPw -count:100000 -profile:"all"
```

Listing 8.1 – Génération de mots de passe sur KeePass

Pour tester l'aléatoire et la robustesse des mots de passe, nous prenons le même script que pour LastPass A.2, qui pour rappel utilise un outil *xxcvbn* qui permet d'évaluer la force du mots de passe et du nombre d'essais qu'un attaquant aura besoin pour le deviner. Ce dernier évalue le nombre d'essais en fonction de 4 attaques différentes (voir section 7.5.1). L'échelle d'évaluation de la génération est également la même (pour des questions de redondance, se référer au chapitre LastPass).

Pour les résultats B.1, nous constatons qu'en moyenne, les mots de passe générés sont bons. KeePass génère rarement des mots de passe un peu plus faibles que d'autres, néanmoins même ceux qui sont considérés plus faibles, ont un bon score contre les différentes attaques.

4. <https://keepass.info/help/base/pwgenerator.html#charset>

5. https://keepass.info/help/v2_dev/scr_index.html

Étant donné que KeePass propose beaucoup de symboles différents dans sa génération, les mots de passes générés qui en possèdent sont forts. Et ainsi, contrairement à LastPass, les mots de passes faibles ne dépendent pas vraiment de leur composition de caractères, nous remarquons que les résultats varient.

Au niveau du nombre de caractères, pour 12 caractères nous remarquons que les mots de passes sont fortement protégés contre les attaques en ligne, néanmoins un peu moins contre les attaques hors-ligne qui indiquent que pour des fonctions de hachage rapide, il faudrait moins de 3 secondes pour deviner le mot de passe. Dès 16 caractères, nous remarquons une meilleure protection au niveau des attaques hors-ligne, cependant ils sont modérément protégés contre les attaques avec hachage rapide. Pour 20 caractères, n'importe quel mot de passe, même un peu plus faible, est suffisamment protégé contre toute attaque.

Vu que KeePass est open-source, nous avons inspecté le code afin d'observer comment l'aléatoire dans les mots de passe était géré. Nous voyons qu'un stream de nombre aléatoire est créé :

```

1 private static CryptoRandomStream CreateRandomStream(byte[]
    pbAdditionalEntropy, out byte[] pbKey)
2 {
3     pbKey = CryptoRandom.Instance.GetRandomBytes(128);
4     // Mix in additional entropy
5     Debug.Assert(pbKey.Length >= 64);
6     if((pbAdditionalEntropy != null) && (pbAdditionalEntropy.Length > 0))
7     {
8         using(SHA512Managed h = new SHA512Managed())
9         {
10             byte[] pbHash = h.ComputeHash(pbAdditionalEntropy);
11             MemUtil.XorArray(pbHash, 0, pbKey, 0, pbHash.Length);
12         }
13     }
14     return new CryptoRandomStream(CrsAlgorithm.ChaCha20, pbKey);
15 }

```

Listing 8.2 – Fonction *CreateRandomStream* de KeePass

```

1 public CryptoRandomStream(CrsAlgorithm a, byte[] pbKey)
2 {
3     if(pbKey == null) { Debug.Assert(false); throw new
        ArgumentNullException("pbKey"); }
4     int cbKey = pbKey.Length;
5     if(cbKey <= 0)
6     {
7         Debug.Assert(false); // Need at least one byte
8         throw new ArgumentOutOfRangeException("pbKey");
9     }
10    m_crsAlgorithm = a;
11    if(a == CrsAlgorithm.ChaCha20)
12    {
13        byte[] pbKey32 = new byte[32]:

```

```
14     byte[] pbIV12 = new byte[12];
15     using (SHA512Managed h = new SHA512Managed())
16     {
17         byte[] pbHash = h.ComputeHash(pbKey);
18         Array.Copy(pbHash, pbKey32, 32);
19         Array.Copy(pbHash, 32, pbIV12, 0, 12);
20         MemUtil.ZeroByteArray(pbHash);
21     }
22     m_chacha20 = new ChaCha20Cipher(pbKey32, pbIV12, true);
23 }
24 [...]
25 }
```

Listing 8.3 – Constructeur *CryptoRandomStream* de KeePass

L'algorithme ChaCha20 est utilisé comme un CSPRNG (*cryptographically secure pseudorandom number generator*), car il est possible d'utiliser des algorithmes *stream cipher* pour ce genre de générateur[11]. En suivant les recommandations du NIST[3] concernant les générateurs de nombres aléatoires, les étapes suivantes sont recommandées :

- **Entropie** consiste à récupérer de l'entropie afin de pouvoir créer la seed aléatoire. KeePass va récupérer l'entropie de l'utilisateur de plusieurs sources différentes.
- **Seed** est utilisée pour instantier le CSPRNG et par la suite générer des bits aléatoires. Afin de la créer, il est recommandé d'utiliser une fonction de hachage afin de compresser les bits d'entropie récupérés précédemment. Dans le premier extrait de code, SHA-512 est utilisé pour compresser les bits. Il est également recommandé d'utiliser un nonce en plus de l'entropie, nous voyons que KeePass génère 128 bits aléatoires en supplément.
- **Re-seeding** cette partie est importante car elle permet de rafraîchir le pool d'entropie interne. Générer trop de sorties d'une même seed pourrait fournir suffisamment d'informations afin de prédire avec succès les futures sorties. Dans le second extrait du code, KeePass prend comme paramètre la seed précédente et la hache à nouveau avec SHA512. Cependant, elle ne prend pas en paramètre l'entropie de l'utilisateur, ce qui est recommandé par le NIST.
- **Génération** la dernière étape consiste à créer le générateur afin de récupérer des valeurs aléatoires. Le générateur est créé à l'aide de l'initialisation d'un cipher. Pour ce gestionnaire, ChaCha20 est utilisé pour générer les bits aléatoires. La clé est changée à chaque fois qu'un mot de passe est généré afin de garantir qu'une future clé compromise ne mette pas en danger des sorties précédentes. De plus, nous remarquons que le tableau qui contient la seed et qui n'est plus utilisé est effacé de la mémoire afin de s'assurer qu'aucun attaquant puisse y avoir accès et deviner les prochaines sorties du générateur.

Dernièrement, sur KeePass, chaque caractère du mot de passe est généré en appelant le cipher ChaCha20 créé précédemment.

Il est mieux d'utiliser des *stream ciphers* au lieu d'utiliser des fonctions PRNG/DRBG (l'algorithme Blum Blum Shub par exemple qui est uniquement utilisé pour la génération de nombres aléatoires), car ils permettent d'être plus simples à implémenter et plus rapides.

8.5.2 Presse-papier

Pour évaluer la sécurité de la fonctionnalité de presse-papier sur KeePass, nous allons effectuer plusieurs analyses différentes. Premièrement, nous allons expliquer comment le gestionnaire gère cette fonctionnalité, puis nous allons effectuer quelques tests et finalement, nous allons analyser le code de l'application.

Dans la version la plus récente, par défaut, KeePass nettoie le presse-papier après 12 secondes et nettoie également le contenu copié dès que KeePass est fermé. Ces deux options permettent d'amener une grande sécurité quant au sniffing de presse-papier par exemple. Le nombre de secondes est paramétrable, cependant 12 secondes est un nombre très suffisant.

De plus, une autre option activée par défaut demande à KeePass de ne pas stocker les éléments copiés dans l'historique de Windows. Une sécurité supplémentaire est également implémentée à propos des applications tierces de presse-papier, l'option *Use Clipboard Viewer Ignore clipboard format* permet à des applications d'ignorer le contenu copié par KeePass, car certaines sauvegardent le contenu du presse-papier et empêche de pouvoir le nettoyer.

Ainsi, nous avons effectué les tests suivants :

- ✓ Copier un mot de passe et tester si le contenu du presse-papier était nettoyé
- ✓ Copier un mot de passe et contrôler qu'il soit bien effacé de l'historique de Windows 11
- ✓ Copier un mot de passe, verrouiller le gestionnaire et contrôler que le mot de passe soit effacé du presse-papier après 12 secondes
- ✓ Copier un mot de passe, fermer l'application et vérifier que rien ne soit stocké dans le presse-papier
- ✓ Copier un mot de passe, copier un autre élément externe à KeePass avant les 12 secondes et vérifier que le mot de passe ne soit pas stocké dans l'historique de Windows
- ✓ Copier un mot de passe, verrouiller le gestionnaire, copier un autre élément externe à KeePass avant les 12 secondes et vérifier que le mot de passe ne soit pas stocké dans l'historique de Windows

Tous les tests ci-dessous ont été testés et ont tous été passés avec succès. Cela signifie que la fonctionnalité du presse-papier est très bien gérée par KeePass et qu'elle garantit une grande sécurité vis-à-vis de cette dernière. Néanmoins, le risque n'est pas à zéro, car un attaquant aurait toujours la possibilité d'ajouter un malware sur le device de l'utilisateur qui permettrait de récupérer les éléments du presse-papier, mais il est difficile de se protéger complètement de ces attaques.

Au niveau du code, en se référant au fichier *ClipboardUtil.Windows.cs*, KeePass n'utilise que des méthodes natives, c'est-à-dire qu'elle importe des fonctions de l'API Utilisateur Windows.

```
1 [DllImport("User32.dll", SetLastError = true)]  
2 internal static extern IntPtr SetClipboardData(uint uFormat, IntPtr hMem);
```

Listing 8.4 – Importation des méthodes natives de l'API de Windows

Ci-dessous, un exemple d'importation que nous pouvons retrouver afin d'importer la copie de données dans le presse-papier. Ainsi, pour Windows, KeePass se base dans un premier temps sur les méthodes qui sont déjà proposées par l'OS, car elles fonctionnent efficacement. Toutefois, des fonctions implémentées par le développeur de KeePass sont proposées afin d'ajouter un support si les méthodes de Windows ne fonctionnent pas.

8.6 Stockage

Étant donné que KeePass ne fonctionne qu'en local, le coffre-fort est stocké dans local sur l'appareil de l'utilisateur. Nous avons déjà expliqué comment KeePass gère le coffre-fort 4.3, néanmoins nous pouvons faire un bref rappel sur sa gestion.

Tout le coffre-fort est stocké dans une base de données *.kdbx* qui contient toutes les informations nécessaires quant à l'ouverture et le déchiffrement du coffre-fort. La base de données est séparée en deux parties ; un header et un contenu. Le header contient tous les bytes pour la dérivation de la master key et le chiffrement des données. Le contenu quant à lui, contient des hashes pour l'authentification du header, afin d'éviter qu'il soit involontairement corrompu et qu'il ne soit pas authentique. Il contient également des blocs de données qui sont chacun authentifiés à l'aide d'un hash HMAC-SHA256 et qui sont chiffrés. Dès que tous les blocs sont déchiffrés à l'aide de la master key, les blocs sont concaténés et cela forme une base de données XML qui contient tout le coffre-fort en clair.

La master key n'est jamais stockée dans la base de données pour éviter qu'un attaquant puisse la récupérer et déchiffrer toute la base de données, néanmoins si le master password est brute-forcé, l'attaquant peut récupérer le coffre-fort en clair.

Ainsi, nous avons suivi les différentes attaques du blog de harmj0y [16] afin d'attaquer la base de données et récupérer le master password. Nous allons tester les attaques suivantes :

- **Attaque #1** brute-force de master password
- **Attaque #2** déverrouiller le coffre-fort avec un facteur *Windows User Account* activé

Attaque #1

Dans cette partie, nous allons tenter de récupérer le master password du coffre-fort depuis la base de données. Hashcat 3.0.0⁶ propose un support pour les versions 1.x et 2.x de KeePass

6. <https://hashcat.net/forum/thread-5559.html>

afin de craquer les hashes de ces derniers. Ainsi, afin d'extraire un hash de la base de données de KeePass compatible avec Hashcat, un outil John The Ripper, `keepass2john` a été développé. De ce fait, ce dernier prend en paramètres le fichier `.kdbx` de la base de données.

Premièrement, nous allons chercher une base de données à attaquer et le binaire de l'application sur le device de l'utilisateur, car elle peut être stockée dans n'importe quel emplacement. Soit on utilise une commande Powershell :

```
1 Get-ChildItem -Path C:\Users\ -Include @("*kee*.exe", "*.kdbx") -Recurse -
   ErrorAction SilentlyContinue | Select-Object -Expand FullName | fl
```

Listing 8.5 – Commande Powershell pour chercher le fichier kdbx

Soit, on cherche le fichier de configuration XML de KeePass où les chemins des différentes base de données sont indiqués (et tous les autres facteurs de connexion). Sur Windows, le fichier se trouve à `C:\Users\user\AppData\Roaming\KeePass\KeePass.config.xml`.

Étant donné que l'outil John The Ripper a quelques années (2013) et n'a pas été mis à jour, il ne supporte que le chiffrement AES, et non ChaCha20 ou Argon2, nous avons donc configuré une base de données avec un master password faible et nous avons laissé les paramètres par défaut lors de la création de coffre-fort, qui sont AES. Nous avons décidé de configurer ces paramètres en se mettant au mieux dans la peau d'un particulier qui aurait peu de connaissances sur la cryptographie.

Dès que nous avons l'emplacement de la base de données, nous lançons l'outil de hash :

```
1 $ keepass2john Database.kdbx
2 Database:$keepass$*2*60000*0*
   fba5a877bc4231428a6ac61bb0db5af6ac92b4ecdd2f35bcc4bd3de7f76ffdd9*30
   de72f912159a9a666e3a624546a09761e4a8932a531d2f56d36d3e08c07dc0*74
   e5d2fb309d9f3ef81d4fbf78562b91*31
   af5d12f9d189defa02193b8d4c2db3d3c7f6cdd95ae7d37552c7121848605c*5
   f36c21ec05f1d97be40c4ce64d2870f2f74392e0ca3acd40304afaa6c7461df
```

Listing 8.6 – Outil keepass2john

Et puis, on utilise Hashcat pour récupérer le master password. On utilise avec un dictionnaire afin de trouver plus rapidement le master password :

```
1 $ hashcat -a 0 -m 13400 -o cracked_output.txt --outfile-format 2 CrackThis.
   hash rockyou.txt
2 Session.....: hashcat
3 Status.....: Cracked
4 Hash.Name.....: KeePass 1 (AES/Twofish) and KeePass 2 (AES)
5 Hash.Target.....: $keepass$*2*60000*222*6b4d1292a0c332f17dc1dfa744e24
   ...37704b
6 Time.Started.....: Mon Dec 5 13:42:32 2022 (5 mins, 40 secs)
7 Time.Estimated...: Mon Dec 5 13:48:12 2022 (0 secs)
8 Guess.Base.....: File (rockyou.txt)
9 Guess.Queue.....: 1/1 (100.00%)
10 Speed.#1.....: 284 H/s (14.02ms) @ Accel:256 Loops:128 Thr:1 Vec:8
```

```
11 Recovered.....: 1/1 (100.00%) Digests
12 Progress.....: 96256/14344384 (0.67%)
13 Rejected.....: 0/96256 (0.00%)
14 Restore.Point....: 94208/14344384 (0.66%)
15 Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:59904-60000
16 Candidates.#1....: 431987 -> master10
```

Listing 8.7 – Commande Hashcat

Nous constatons que Hashcat a correctement cracké le master password et l’a trouvé en 5 minutes, ce qui est plutôt rapide. Néanmoins, le master password était *password98*, ce qui est faible mais peut refléter le genre de mots de passe que les particuliers pourrait configurer.

Finalement, l’attaquant connaît l’emplacement de la base de données et le master password, ainsi il peut la déverrouiller avec facilité.

Attaque #2

Cette attaque demande d’avoir quelques informations au préalable et est donc plus difficile à reproduire. Cette attaque permet à un attaquant, qui aurait pu récupérer la base de données de l’utilisateur, son master password à l’aide d’un keylogger et le mot de passe Windows de ce dernier. Si l’option de connexion *Windows User Account* est activée, la personne malveillante ne pourra malheureusement pas ouvrir le coffre-fort.

Nous allons premièrement expliquer comment KeePass utilise ce facteur de connexion afin de créer la master key de l’utilisateur. Il se sert du DPAPI (*Windows Data Protection Application Programming Interface*)[48]. C’est une interface qui propose des fonctions cryptographiques qui permettent de chiffrer / déchiffrer des données sensibles DPAPI appelées *binary large objects* ou *blob*⁷. Les clés DPAPI, dérivées du mot de passe et utilisées pour chiffrer la master key, sont stockées sous `C:\Users\{username}\AppData\Roaming\Microsoft\Protect\{SID}`. La master key permet de chiffrer des données *blob*.

Dans le code de KeePass, dans le fichier *KcpUserAccount*, nous pouvons constater que KeePass crée une données protégées DPAPI à l’aide d’entropie, de l’utilisateur actuel et de données aléatoires. Ces données sont ensuite stockées dans un fichier *ProtectedUserKey.bin* qui est localisé sous `C:\Users\{username}\AppData\Roaming\KeePass\` sur Windows.

Étant donné que des utilisateur de KeePass ont déjà perdu leur master key incluant leur compte Windows, une procédure complète a été expliquée afin de recover le facteur de connexion *Windows User Account*⁸, et ainsi déverrouiller leur coffre-fort. Nous n’allons pas décrire toute la procédure car elle est complexe, cependant un script implémenté par harmj0y automatise tout le processus[17].

De ce fait, le script a besoin du répertoire avec le SID de l’utilisateur, le nom et le domaine de l’utilisateur puis le fichier de KeePass *ProtectedUserKey.bin*.

7. Données binaires stockées comme une seule entité

8. <https://sourceforge.net/p/keepass/wiki/Recover%20Windows%20User%20Account%20Credentials/>

— ajouter image script!!!

8.7 Mémoire

À propos de la gestion de la mémoire sur KeePass[40], comme nous l'avons également cité dans la section précédente, DPAPI est utilisé sur Windows afin de chiffrer les données sensibles en mémoire. KeePass considère que les données sensibles du coffre-fort sont la master key et les mots de passe des entrées. Ainsi, les usernames, les notes ou les URLs sont en clairs dans la mémoire du processus, car ces informations n'exposent pas des données importantes et sensibles.

En se basant sur une étude de 2019 sur le management de secrets dans les gestionnaires de mots de passe[19], des mots de passe en clair sont exposés dans la mémoire du processus
outil keethief

8.8 Récapitulatif de l'analyse

-lors de l'inscription KeePass indique quelques indices/aides sur les attaques de mots de passe ou sur la sécurité en général. good point pour les particuliers.

Chapitre 9

Firefox

9.1 Critères d'analyse

Chapitre 10

Conclusion

Bibliographie

- [1] 1Password. 1password security design, 2021.
- [2] Apr4h. Decrypting browser credentials for fun (but not profit), 2019.
- [3] Elaine Barker and John Kelsey. Recommendation for random number generation using deterministic random bit generators. Technical Report NIST Special Publication (SP) 800-90A Rev. 1, National Institute of Standards and Technology, Gaithersburg, MD, 2015.
- [4] Derk Barten. "client-side attacks on the lastpass browser extension", 2019.
- [5] bitwarden. Bitwarden security whitepaper, 2022.
- [6] bitwarden. World password day global survey full report, 2022.
- [7] Michael Carr and Siamak F. Shahandashti. "revisiting security vulnerabilities in commercial password managers", 2020.
- [8] Laurent Clévy. "firepwd.py, an open source tool to decrypt mozilla protected passwords", 2020.
- [9] Clifford Colby, Rae Hodge, and Attila Tomaschek. Best password manager to use for 2022, 2022.
- [10] Léo Corthès. "analyse forensique de la ram macos - extraction de clés de chiffrement", 2019.
- [11] cr.yp.to. "fast-key-erasure random-number generators : An effort to clean up several messes simultaneously", 2017.
- [12] Dashlane. Security white paper, 2022.
- [13] ECRYPT-CSA. Algorithms, key size and protocols report, 2018.
- [14] Elizabeth A. Gallagher. Choosing the right password manager. *Serials Review*, 45 :1–2, 84–87, 2019.
- [15] Eric Griffith. How to master google password manager, 2022.
- [16] harmj0y. "a case study in attacking keepass", 2016.
- [17] harmj0y. "restore-userdpapi.ps1", 2016.

- [18] Jingxin Hong Hengwei Zhang and Jun Hu. Analysis of encryption mechanism in keepass password safe 2.30. *2016 10th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, pages 43–46, 2016.
- [19] ISE. Password managers : Under the hood of secrets management, 2019.
- [20] Technologies de l'information - Techniques de sécurité - Gestion des risques liés à la sécurité de l'information. Standard, International Organization for Standardization, July 2018.
- [21] KeePass. "kdbx 4", 2017.
- [22] KeePass. Security, 2022.
- [23] Keeper. Keeper encryption model, 2022.
- [24] Michael Kurko. Best password managers, 2022.
- [25] LastPass. Technical whitepaper.
- [26] lgg. Keepass file format explained, 2017.
- [27] Blanchou Marc and Youn Paul. "password manager : Exposing password everywhere", 2013.
- [28] Paulius Masiliauskas. Most secure password managers in 2022, 2022.
- [29] William Melicher, Blase Ur, Sean M. Segreti, Saranga Komanduri, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Fast, lean, and accurate : Modeling password guessability using neural networks. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 175–191, Austin, TX, August 2016. USENIX Association.
- [30] Leo N. Idiomatic cryptography : Pbkdf2, sha, aes, 2022.
- [31] NordPass. "nordpass business whitepaper", 2022.
- [32] Timothy Oesch. "an analysis of modern password manager security and usage on desktop and mobile devices.", 2021.
- [33] Okta. "password entropy : The value of unpredictable passwords", 2021.
- [34] OWASP. *Application Security Verification Standard 4.0.3*, 2021.
- [35] Larry Conklin (OWASP). Threat modeling process, 2022.
- [36] Padloc. Security whitepaper, 2022.
- [37] PasswordManager. Password manager trust survey, 2020.
- [38] Dominik Reichl. "kdbx 4", 2017.
- [39] Dominik Reichl. "password generator", 2022.
- [40] Dominik Reichl. "process memory protection", 2022.
- [41] Tom Ritter. Private by design : How we built firefox sync, 2018.
- [42] Chuan Yue Rui Zhao and Kun Sun. "security analysis of two commercial browser and cloud based password managers", 2013.

- [43] Hives Systems. Are your password in the green, 2022.
- [44] Security.org Team. Password manager and vault 2021 annual report : Usage, awareness, and market size, 2021.
- [45] Verizon. 2022 data breach investigations report, 2022.
- [46] Liz Wegerer. Is your browser's password manager safe ?, 2022.
- [47] Daniel Lowe Wheeler. zxcvbn : Low-Budget password strength estimation. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 157–173, Austin, TX, August 2016. USENIX Association.
- [48] Wikipedia. "data protection api", 2022.
- [49] Richard Wood. "what's the difference between qualitative and quantitative risk analysis ?", 2019.
- [50] Fei Yu and Hao Yin. A security analysis of the authentication mechanism of password managers. In *2021 IEEE 21st International Conference on Communication Technology (ICCT)*, pages 865–869, 2021.

Table des figures

1	Planning du travail de Bachelor	xvi
4.1	Déchiffrement d'un mot de passe en local sur Firefox	19
4.2	Schéma de synchronisation de données sur Firefox	20
4.3	Contenu du fichier .kdbx de KeePass	22
4.4	Génération de la Master key sur KeePass	24
4.5	Schéma de déchiffrement de LastPass	25
4.6	Partage d'un identifiant sur Dashlane	27
4.7	Réception d'un partage d'identifiant sur Dashlane	28
4.8	Différents états d'un gestionnaire de mots de passe	30
5.1	Critères d'évaluation des risques	36
5.2	Data Flow Diagram pour les gestionnaires M1	39
5.3	Data Flow Diagram pour les applications M2	43
5.4	Data Flow Diagram pour les synchronisations	47
5.5	Data Flow Diagram pour les partages de données	51
5.6	Schéma bas-niveau des gestionnaires de mots de passe	54
6.1	Comparatif des candidats pour la sélection	63
7.1	Identifiants pour heig-vd.ch enregistrés sur LastPass	72
7.2	Auto-complétion d'identifiants sur webmail.heig-vd.ch	73
7.3	Auto-complétion d'identifiants sur gaps.heig-vd.ch	74
7.4	Table <i>LastPassSavedLogins2</i> vide	75

7.5	Entrée du coffre-fort LastPass en clair	76
7.6	Master key en mémoire	77
7.7	Master key en mémoire	78
8.1	Avertissement de master password faible de KeePass	84

Liste des tableaux

2.1	Fonctionnalités des candidats sélectionnés	8
2.2	Plateformes supportées par les différentes applications	9
2.3	Tarifs pour particuliers	10
2.4	Tarifs pour les entreprises	11
3.1	Temps pour craquer les mots de passe en fonction des critères	14
4.1	Algorithmes cryptographiques des candidats	31
5.1	Scores de sévérité basés sur CVSS v3.1	35
5.2	Impact des menaces sur l'entreprise	35
5.3	Probabilités d'événements	36
5.4	Biens des gestionnaires de mots de passe M1	39
5.5	Sources de menaces des gestionnaires de mots de passe M1	40
5.6	Scénarios et types d'attaques possibles sur applications M1	40
5.7	Vulnérabilités présentes dans les gestionnaires de mots de passe M1	41
5.8	Conséquences des menaces sur l'entreprise	42
5.9	Analyse des risques de chaque scénario de menaces pour M1	42
5.10	Biens des gestionnaires de mots de passe M2	44
5.11	Sources de menaces des gestionnaires de mots de passe M2	44
5.12	Scénarios et types d'attaques possibles sur M2	44
5.13	Vulnérabilités présentes dans les gestionnaires de mots de passe M2	45
5.14	Conséquences des menaces sur l'entreprise	45

5.15	Analyse des risques de chaque scénario de menace sur M2	46
5.16	Biens des gestionnaires de mots de passe M3	48
5.17	Sources de menaces des gestionnaires de mots de passe M3	48
5.18	Scénarios et types d'attaques possibles sur M3	48
5.19	Vulnérabilités présentes dans les gestionnaires de mots de passe M3	49
5.20	Conséquences des menaces sur l'entreprise	49
5.21	Analyse des risques de chaque scénario de menaces sur M3	50
5.22	Biens des gestionnaires de mots de passe M4	52
5.23	Sources de menaces des gestionnaires de mots de passe M4	52
5.24	Scénarios et types d'attaques possibles sur M4	52
5.25	Vulnérabilités présentes dans les gestionnaires de mots de passe M4	53
5.26	Conséquences des menaces sur l'entreprise	53
5.27	Analyse des risques de chaque scénario de menaces sur M4	53
5.28	Biens des gestionnaires de mots de passe M5	55
5.29	Sources de menaces des gestionnaires de mots de passe M5	55
5.30	Scénarios et types d'attaques possibles sur M5	55
5.31	Vulnérabilités présentes dans les gestionnaires de mots de passe M5	56
5.32	Conséquences des menaces sur l'entreprise	56
5.33	Analyse des risques de chaque scénario de menace sur M5	57
5.34	Contre-mesures possibles pour les gestionnaires	58
5.35	Exigences sécuritaires	60
7.1	Environnement utilisé pour LastPass	65
7.2	Fonctionnalités de LastPass pour la génération de mots de passe	69
8.1	Environnement utilisé pour KeePass	83
8.2	Fonctionnalités de KeePass pour la génération de mots de passe	87
A.1	Résultats de la robustesse des mots de générés sur LastPass	115
B.1	Résultats de la robustesse des mots de générés sur KeePass	119

C.1 Journal de travail	122
----------------------------------	-----

Liste des listings

7.1	Calcul de la master key sur LastPass	77
8.1	Génération de mots de passe sur KeePass	88
8.2	Fonction <i>CreateRandomStream</i> de KeePass	89
8.3	Constructeur <i>CryptoRandomStream</i> de KeePass	89
8.4	Importation des méthodes natives de l'API de Windows	92
8.5	Commande Powershell pour chercher le fichier kdbx	93
8.6	Outil keepass2john	93
8.7	Commande Hashcat	93
A.1	Script pour la génération de mots de passe LastPass	113
A.2	Script pour tester les mots de passe de LastPass	114
A.3	Script pour déchiffrer les entrées LastPass	115

Annexe A

Annexes LastPass

A.1 Génération de mots de passe

```

1 // Initialisation du ChromeDriver
2 System.setProperty("webdriver.chrome.driver", "chromedriver.exe");
3 WebDriver driver = new ChromeDriver();
4 // Liaison avec l'extension LastPass
5 driver.get("https://lastpass.com/?ac=1");
6
7 // Connexion au gestionnaire de mots de passe
8 WebElement email = driver.findElement(By.xpath("//*[@id=\"root\"]/div/form/
    div[1]/div/div/input"));
9 WebElement password = driver.findElement(By.xpath("//*[@id=\"root\"]/div/
    form/div[2]/div/input"));
10 email.clear();
11 email.sendKeys("$EMAIL");
12 password.clear();
13 password.sendKeys("$PASSWORD");
14 driver.findElement(By.className("euz05gu0")).click();
15
16 // Acces au menu de generation de mots de passe
17 driver.switchTo().frame("newvault");
18 driver.findElement(By.xpath("//*[@id=\"advancedMenuItem\"]/div/span")).click
    ();
19 driver.findElement(By.cssSelector("#generatePasswordMenuItem")).click();
20
21
22 // Configuration du nombre de caracteres qu'on souhaite, ici on en veut 12
23 WebElement numberCar = driver.findElement(By.xpath("//*[@id=\"lengthInput\"
    \"]"));
24 numberCar.clear();
25 numberCar.sendKeys("12");
26
27

```

```
28 // On coche ou decoche les cases pour avoir que des lettres minuscules et
    majuscules
29 WebElement digits = driver.findElement(By.xpath("//*[@id=\"
    generatePasswordDialogDropdownAdvancedOptions\" +
30 \"\"]/div/span[2]/div[3]/div/label")).click();
31
32 // Creation d'un fichier local
33 pour recuperer tous les mots de passe generes
34 File fout = new File("$OUTPUT_FILE");
35 FileOutputStream fos = new FileOutputStream(fout);
36 BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(fos));
37 // Generation de tous les mots de passe et ecriture dans le fichier local
    ligne par ligne
38 for (int i = 0; i < 10000; ++i) {
39     WebElement pass = driver.findElement(By.xpath("//*[@id=\"password\"]"));
40     bw.write(pass.getAttribute("value"));
41     bw.newLine();
42     WebElement refresh = driver.findElement(By.xpath("//*[@id=\"generateBtn\"]
        \""));
43     refresh.click();
44 }
45 bw.close();
46 driver.quit();
```

Listing A.1 – Script pour la génération de mots de passe LastPass

```
1 Zxcvbn zxcvbn = new Zxcvbn();
2 Vector<Double> guesses_min = new Vector<>();
3 Vector<String> weak_passwords = new Vector<>();
4
5 BufferedReader br = new BufferedReader(new FileReader("$INPUT_FILE"));
6
7 for (String line = br.readLine(); line != null; line = br.readLine()) {
8     Strength strength = zxcvbn.measure(line);
9     double guess = strength.getGuessesLog10();
10
11     if(guess <= Collections.min(guesses_min)) {
12         guesses_min.add(guess);
13         weak_passwords.add(line);
14     }
15 }
```

Listing A.2 – Script pour tester les mots de passe de LastPass

Nombre chars	Composition	Mot de passe	Nombre d'essais (log ₁₀)	Moyenne du nombre d'essais (log ₁₀)	Temps d'essai en fonction des attaques			
					Throttled online attack	Unthrottled online attack	Offline attack, slow hash	Offline attack, fast hash
12	ll	markZTkzuBfL	10.00000	11.95351	siècles	31 ans	12 jours	1 seconde
	sd	^!09!!7^5\$#@	8.00004	11.75474	siècles	4 mois	3 heures	>1 seconde
	ld	nQ5bwleUg3NE	9.85187	11.96165	siècles	22 ans	8 jours	>1 seconde
	ls	*IJNvertatw	9.95904	11.96757	siècles	28 ans	11 jours	>1 secondes
	all	QfQ!pK*2323	10.00000	11.97474	siècles	31 ans	12 jours	1 secondes
16	ll	ruXnanagdFVZTgtE	13.48979	15.93678	siècles	siècles	96 ans	51 minutes
	sd	#93&!!1949@%432!	12.22354	15.67387	siècles	siècles	5 ans	3 minutes
	ld	S3cR37t3sGj07GXX	13.28869	15.94858	siècles	siècles	60 ans	32 minutes
	ls	&lpP^AUbXKJqUDDR	13.47712	15.95871	siècles	siècles	93 ans	50 minutes
	all	CV1b8C6Gr55g0!ng	13.42160	15.96274	siècles	siècles	82 ans	44 minutes
20	ll	dZkflsLXTnWTrEwqIWfx	17.47712	19.92035	siècles	siècles	siècles	12 jours
	sd	&&&22051904!#@44%48\$	16.02168	19.59983	siècles	siècles	siècles	12 jours
	ld	UROJd82Hn2UJLin3xkj	18.04462	19.936259	siècles	siècles	siècles	33 ans
	ls	o!UEG\$W%AT!Uwh@tcPAH	17.77815	19.94897	siècles	siècles	siècles	2 ans
	all	AYAiCyds3d!nd44!3l	17.47712	20.00000	siècles	siècles	siècles	11 mois

TABLE A.1 – Résultats de la robustesse des mots de générés sur LastPass

A.2 Accès mémoire

```

1 #!/usr/bin/python
2 # Author : Leo Corthes
3
4 import base64
5 import re
6 import sys
7 import hashlib
8 from Crypto.Cipher import AES
9 from Crypto import Random
10 import string
11 import sqlite3
12 from sqlite3 import Error
13 from shutil import copyfile
14 from pathlib import Path
15
16 # Geto connection to local SQLite database
17 # http://www.sqlitetutorial.net/sqlite-python/sqlite-python-select/
18 def create_connection(db_file):
19     try:
20         conn = sqlite3.connect(db_file)
21         return conn
22     except Error as e:
23         print(e)
24     return None
25
26 # Select the field which contains the LastPass encrypted data
27 def selectEntries(conn):
28     cur = conn.cursor()
29     cur.execute("SELECT data FROM LastPassData where type='accts'")
30     rows = cur.fetchall()
31     for row in rows:

```

```
32     return str(row)
33
34 BLOCK_SIZE = 16
35 pad = lambda s: s + (BLOCK_SIZE - len(s) % BLOCK_SIZE) * chr(BLOCK_SIZE -
36     len(s) % BLOCK_SIZE)
37 unpad = lambda s: s[:-ord(s[len(s) - 1:])]
38 printset = set(string.printable)
39
40 # AES-CBC 256 encryption
41 # Inspired and modified from : https://www.quickprogrammingtips.com/python/aes-256-encryption-and-decryption-in-python.html
42 def encrypt(raw, password):
43     private_key=bytes.fromhex(KEY)
44
45     raw = pad(raw)
46     iv=IV
47     cipher = AES.new(private_key, AES.MODE_CBC, iv)
48     return base64.b64encode(iv + cipher.encrypt(raw.encode("utf-8")))
49
50 # AES-CBC 256 Decryption
51 def decrypt(enc, key):
52     private_key=bytes.fromhex(key)
53     enc = base64.b64decode(enc)
54     iv = enc[:16]
55     retVal=""
56     try:
57         cipher = AES.new(private_key, AES.MODE_CBC, iv)
58         retVal = unpad(cipher.decrypt(enc[16:]))
59     except:
60         pass
61     return retVal
62
63 # Look for hexadecimal strings and convert them into ASCII. The URL is
64 # stored this way
65 def lookForWebsiteName(fields):
66     b = re.findall(b"[0123456789ABCDEFabcdef]*", fields)
67     for e in b:
68         if len(e.decode()) > 3:
69             try:
70                 print("Website : " + bytes.fromhex(e.decode()).decode('utf-8'))
71             except:
72                 pass
73
74 # Decrypt the username. The encrypted username starts at the beginning of a
75 # field and ends with \0\0\0
76 def decryptUsername(key, field):
77     try:
78         # First part of the field contains the username
79         encUsername=field.split(b"\0\0\0")[0]
80         print("Username : " + decrypt(base64.b64encode(encUsername), key).decode())
81     except:
82         pass
```

```

78     except:
79         pass
80
81     # Decrypt the password. The encrypted password is either stored directly
82     # after the encrypted username.
83     # Or after the encrypted username with a "!" as separator.
84     # It can also be stored in a completely different field.
85     def decryptPassword(key, field):
86         try:
87             # Second part of the field contains the password
88
89             encPassword=field.split(b"\0\0\0")[1]
90             if '!' in str(encPassword):
91                 encPassword=encPassword.split(b"!")[1]
92                 print("Password : " + decrypt(base64.b64encode(encPassword), key).
93                     decode())
94             else :
95                 print("Password : " + decrypt(base64.b64encode(encPassword), key).
96                     decode())
97         except:
98             pass
99
100    # Decrypt whatever can be found in a field. It could be a username, a
101    # password, a note, an entry name, ...
102    def decryptChunk(key, field, name="Data"):
103        try :
104            chunk=field.split(b"\0\0\0")[0]
105            print(name + " : " + decrypt(base64.b64encode(chunk), key).decode())
106        except:
107            pass
108
109    def main():
110        if (len(sys.argv) == 2):
111            database = str(".config/google-chrome/Default/databases/chrome-
112                extension_hdokiejnpimakedhajhdlcegeplioahd_0/9")
113            key = sys.argv[1]
114        elif(len(sys.argv) == 3):
115            database = sys.argv[1]
116            key = sys.argv[2]
117        else :
118            print("You must provide the path to the database and the file containing
119                the potential keys as argument. If you provide nothing, default values
120                will be used.")
121            print("You must provide key that can be found in Javascript as
122                g_local_key")
123            exit()
124
125    # Connecting to the local database and getting the encrypted data
126    conn = sqlite3.connect(database)
127
128    with conn:

```

```
121 encData=selectEntries(conn)
122
123 # Decoding the data and separate each entry
124 encDataBin=base64.b64decode(encData)
125 entries=(encDataBin.split(b"ACCT"))
126 entries.pop(0)
127
128
129 # Iterating over all the entries
130 for entry in entries:
131     print("===== NEW ENTRY =====")
132     fields=[]
133     fields=entry.split(b"!!")
134
135     # Iterating over each field, trying to find some artefacts
136     for i in range(0, len(fields)):
137         lookForWebsiteName(fields[i])
138     for i in range(0, len(fields)):
139         decryptUsername(key, fields[i])
140     for i in range(0, len(fields)):
141         decryptPassword(key, fields[i])
142     for i in range(0, len(fields)):
143         decryptChunk(key, fields[i])
144
145 if __name__ == "__main__":
146     main()
```

Listing A.3 – Script pour déchiffrer les entrées LastPass

Annexe B

Annexes KeePass

B.1 Génération de mots de passe

Nombre chars	Composition	Mot de passe	Nombre d'essais (\log_{10})	Moyenne du nombre d'essais (\log_{10})	Temps d'essai en fonction des attaques			
					Throttled online attack	Unthrottled online attack	Offline attack, slow hash	Offline attack, fast hash
12	ll	thEyFYdFWaS	9.60852	11.94930	siècles	94 ans	1 mois	3 secondes
	sd) * + , & 54 < 8 . \$ /	9.50785	11.95862	siècles	10 ans	4 jours	> 1 seconde
	ld	y4YVElJmElJm	8.60211	11.95883	siècles	1 an	11 heures	> 1 seconde
	ls	o YRamsnrOfa	10.47856	11.98163	siècles	94 ans	1 mois	3 secondes
	all	thls{J\$2Vz%W	10.00000	11.98263	siècles	31 ans	12 jours	1 seconde
16	ll	nZtSABbxvyyTteb	13.44715	15.93299	siècles	siècles	87 ans	47 minutes
	sd	l :-09*751 ⁽¹⁾ \88+~	13.47712	15.94364	siècles	siècles	93 ans	50 minutes
	ld	FlXXM980rTZPsNTH	14.00000	15.94680	siècles	siècles	siècles	3 heures
	ls	sO CERg,=WAZPwt :	12.93449	15.97629	siècles	siècles	27 ans	14 minutes
	all	uLEW{i\$4}a!q&cog : @	14.06505	15.97639	siècles	siècles	siècles	3 heures
20	ll	HgcSqdWNuBYBYHBdrama	17.05690	19.92115	siècles	siècles	siècles	4 mois
	sd	*~ / : * & , { ' @ / : 8241971	17.32572	19.93353	siècles	siècles	siècles	8 mois
	ld	BZEWNUwa173r7dPKQUwL	17.29402	19.93458	siècles	siècles	siècles	7 mois
	ls	AF_zql\ Ri < x % . - % . A	19.97082	18.12100	siècles	siècles	siècles	4 ans
	all	3579 +s(:\$[3'% LTfm	18.00000	20.00000	siècles	siècles	siècles	3 ans

TABLE B.1 – Résultats de la robustesse des mots de générés sur KeePass

Annexe C

Journal de travail

TABLE C.1 – Journal de travail

Date	Description	Rech. [h]	Dev. [h]	Rapport [h]	Admin [h]
> 20.09.22	Discussion avec le professeur responsable, établissement du cahier des charges, introduction	7	0	10	4
20.09.2022	Update + organisation du TB, planing, relire le début du TB déjà commencé, avancement de l'étude du marché (fonctionnalités, plateformes, prix), lecture d'articles	2	0	5	1
21.09.2022	Recherches sur les statistiques des gestionnaires de mots de passe sur le marché, rédaction du chapitre étude de marché (terminé ce jour-ci)	3	0	3	0
22.09.2022	Introduction et organisation du chapitre étude sécuritaire, recherche et lecture sur les différentes implémentations sécuritaire des gestionnaires de mots de passe	3	0	1	1
23.09.2022	Recherche et lecture sur les différentes implémentations sécuritaire des gestionnaires de mots de passe et organisation du rapport	1	0	1	0
26.09.2022	Recherche et lecture sur les gestionnaires de mots de passe browser-based, rédaction dans le rapport à ce propos	4	0	1	0
27.09.2022	Recherche et lecture sur les gestionnaires de mots de passe browser-based et local-based, et rédaction dans le rapport	3	0	2	0
28.09.2022	Recherche et lecture sur les gestionnaires de mots de passe browser-based et local-based, et rédaction dans le rapport	4	0	1	0
29.09.2022	Recherche et lecture sur les gestionnaires de mots de passe local-based, et rédaction dans le rapport	5	0	3	0
30.09.2022	Rédaction de la section du partage d'informations, des 3 états du gestionnaires de mots de passe	1	0	4	0
03.10.2022	Fin du chapitre 3 sur analyse de menaces, lecture sur la modélisation de menaces et la norme que je souhaite suivre	7	0	1	0

Le journal de travail continue à la page suivante.

Date	Description	Rech. [h]	Dev. [h]	Rapport [h]	Admin [h]
04.10.2022	Début chapitre analyse de menaces et lecture de la norme choisie	6	0	2	0
05.10.2022	Lecture et rédaction établissement du contexte de l'analyse de menaces	5	0	3	0
06.10.2022	Lecture et rédaction établissement du contexte de l'analyse de menaces	2	0	6	0
07.10.2022	Lecture et rédaction identification des risques de l'analyse de menaces	5	0	3	0
09.10.20220	Lecture et rédaction identification des risques de l'analyse de menaces	2	0	2	0
10.10.20220	Lecture et rédaction identification des risques de l'analyse de menaces	3	0	5	0
11.10.20220	Rédaction et lecture analyses des risques et évaluation des risques	4	0	4	0
12.10.20220	Rédaction et lecture analyses des risques et évaluation des risques	3	0	4	0
13.10.20220	Rédaction du traitement des risques et lecture à ce propos avec les contremesures possibles	3	0	3	0
14.10.20220	Rédaction de la section exigences sécuritaires à respecter, relecture du TB, organisation du rapport et rendu intermédiaire du rapport	1	0	6	0
16.10.20220	Rédaction de la sélection des candidats	2	0	6	0
17.10.20220	Rédaction de la sélection des candidats	3	0	3	0
18.10.20220	Début de recherches spécifiques sur LastPass, organisation du chapitre	5	0	2	0
19.10.20220	Début analyse LastPass -> master password, choix cryptographiques, génération de mots de passe	2	0	6	0
20.10.20220	Analyse sur LastPass -> analyse de la qualité de la génération de mots de passe sur LastPass en codant un script	2	0	6	0

Le journal de travail continue à la page suivante.

Date	Description	Rech. [h]	Dev. [h]	Rapport [h]	Admin [h]
30.10.20220	Feedback sur le rendu intermédiaire	0	0	1	2
31.10.20220	Corrections du rendu intermédiaire	2	0	5	0
01.11.20220	Corrections du rendu intermédiaire	6	0	1	0
02.11.20220	Corrections du rendu intermédiaire	5	0	1	0
03.11.20220	Corrections du rendu intermédiaire	5	0	2	0
05.11.20220	Corrections du rendu intermédiaire	1	0	2	0
08.11.20220	Corrections du rendu intermédiaire	1	0	7	0
09.11.20220	Corrections du rendu intermédiaire	1	0	3	0
10.11.20220	Corrections du rendu intermédiaire	0	0	6	0
11.11.20220	Analyse de LastPass	5	0	3	0
13.11.20220	Analyse de LastPass	7	0	0	0
14.11.20220	Analyse de LastPass	5	0	3	0
15.11.20220	Analyse de LastPass -> génération de mots de passe	2	5	1	0