



République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene
Faculté d'électronique et d'informatique

Projet pluridisciplinaire

Spécialité :

2^{ème} année

Tronc Commun Ingénieur d'Etat en Informatique

Thème :

Attaques sur les mots de passe

Réalisé par:

1. DOB Serine
2. NAIT-CHERIF Sabrinel
3. SALHI Racha
4. GUENDOUL Hayat
5. GHOMARI Djazia

Proposé par :

- Dr. Halim ZAIDI

N°groupe: 34

Remerciement

Avant tout, nous tenons à remercier Dieu qui nous a donné le courage, la force et la volonté nécessaires pour accomplir ce projet.

Nous adressons nos sincères remerciements à Monsieur Zaidi, notre superviseur, pour l'aide précieuse qu'il nous a apportée et les connaissances qu'il a su nous transmettre avec tant de pédagogie. Nous le remercions également pour sa présence constante et la qualité de ses conseils. Malgré le temps considérable qu'il nous a consacré, et même lors de nos nombreuses rencontres, il ne nous a jamais laissé sentir qu'il était préoccupé par ses propres besoins, mais il a toujours su nous guider avec bienveillance et patience chaque fois que nous rencontrions des difficultés. Nous adressons aussi nos sincères remerciements à tous nos professeurs de la faculté d'informatique pour leurs efforts constants et leur dévouement à notre égard.

Le Prophète (paix et salut sur lui) a dit : "Celui qui ne remercie pas les gens ne remercie pas Dieu."

Table des matières

Introduction Générale.....	5
Problématique	5
Objectif	5
Chapitre I : Cadre Théorique et Analyse.....	7
1.1. L'utilité des mots de passe	7
1.2. Comment protéger les mots de passe	7
1.3. Fonctions de hachage.....	7
MD5 (Message Digest Algorithm 5)	7
SHA-1 (Secure Hash Algorithm 1).....	8
SHA-256 (Secure Hash Algorithm 256-bit).....	8
SHA-512 (Secure Hash Algorithm 512-bit).....	8
Remarque	8
1.4. Attaques des mots de passe.....	8
Chapitre 2 : Conception et Implémentation	9
2.1. Force brute.....	9
2.2. Attaques par dictionnaire.....	13
2.3. Attaque qui combine Brute force et dictionnaire	16
2.4. Arc en ciel	18
.....	22
Chapitre III : Test et Exploitation.....	23
3.1. Office.....	23
3.2. Attaque sur les mots de passe sous Linux	25
3.3. Test de logiciel.....	26
3.5. Avantages de ce projet.....	28
3.6. Division des tâches	28
Conclusion générale.....	29
Bibliographie	30
Webographie	31

Table des figures

Figure 1: Temps de craquage des mots de passe par force brute.	12
Figure 2: Temps de Génération des Mots de Passe par Force brute/dictionnaire	17
Figure 3: Mécanisme de Stockage et de Récupération de Données Hachées	18
Figure 4: Espace de Stockage Requis pour 'Algorithme	20
Figure 5: Comparaison des méthodes d'attaques.....	22
Figure 6: Utilisation de John the Ripper pour Craquer des Mots de Passe Office.	24
Figure 7: Méthodologie d'Utilisation de Hashcat pour Fichiers Office..	25
Figure 8 :Attaque combiné brute force/dictionnaire.	26
Figure 9: Attaque force brute	26
Figure 10: Attaque Arce en ciel.....	27
Figure 11: Attaque par dictionnaire	27
Figure 12:Attaque par dictionnaire avec des variations	27
Figure 13:Attaque par dictionnaire avec des variations	28

Liste des tableaux

Tableau 1: Estimation des Combinaisons de Mots de Passe par Taille..	11
Tableau 2: Durée Maximale de craquer des Mots de Passe selon la Taille.	12
Tableau 3: tableau de variations	14

Introduction Générale

Dans le vaste domaine de l'informatique, la sécurité est devenue un enjeu crucial à l'ère numérique, touchant tant les entreprises que les particuliers. L'actualité est tristement marquée par une vague incessante de cyberattaques, de ransomwares et de fuites de données. Face à l'émergence du Dark Web et l'utilisation généralisée des cryptomonnaies pour des transactions illégales, les cybercriminels ont trouvé un terrain fertile pour leurs activités lucratives. En 2020 seulement, les pertes financières dues à ces attaques ont été estimées à près de 1000 milliards de dollars, représentant ainsi 1 % du PIB mondial. Cette menace ne se cantonne plus aux grandes entreprises ; elle cible également les administrations, les établissements d'enseignement, les hôpitaux et les particuliers. Même les systèmes industriels connectés (SCADA) ne sont pas à l'abri, subissant des attaques aux conséquences matérielles dramatiques.

Pour contrer cette menace croissante, la collaboration entre les professionnels de la sécurité informatique est essentielle. Les entreprises se rendent compte qu'elles doivent investir dans la protection de leurs systèmes, en comprenant que la sécurité des données est cruciale pour leur survie. Cela se traduit souvent par le recours à des experts externes pour tester leurs réseaux et systèmes d'information, afin d'identifier et de corriger les failles de sécurité.

Dans ce contexte de cybermenaces omniprésentes, les attaques visant les mots de passe constituent l'une des menaces les plus courantes et persistantes dans le domaine de la sécurité informatique. Les mots de passe sont souvent la première ligne de défense pour protéger l'accès à des comptes sensibles, des données confidentielles et des systèmes critiques.

Problématique

Dans un monde de plus en plus numérique, la sécurité des mots de passe reste un maillon essentiel de la protection des informations sensibles. Bien que des avancées en matière de cybersécurité, les mots de passe continuent d'être vulnérables aux attaques des pirates informatiques. Les techniques d'attaque deviennent de plus en plus efficaces, posant un défi constant pour les utilisateurs.

- Quelles sont les méthodes de protection des mots de passe actuellement utilisées ?
- Quelles sont les méthodes d'attaque et sont-elles efficaces ?

Objectif

Ce projet a pour objectif de comprendre les mécanismes de protection des mots de passe et d'analyser les différentes techniques d'attaque utilisées par les pirates informatiques. Nous évaluerons l'efficacité et la complexité de ces attaques, en nous concentrant sur des méthodes courantes telles que les attaques par force brute et les attaques par dictionnaire.

Pour ce faire, nous développerons un système avancé capable de simuler des attaques de mots de passe en utilisant diverses techniques et plusieurs fonctions de hachage, telles que MD5 et SHA-256. Ce système pourra être utilisé pour évaluer la sécurité des mots de passe, fournissant ainsi des informations précieuses pour améliorer la sécurité des utilisateurs individuels et des entreprises.

Chapitre I : Cadre Théorique et Analyse

Dans ce chapitre, nous allons examiner la manière dont les mots de passe sont protégés et étudier les différentes fonctions de hachage utilisées à cet effet. Nous explorerons également quelques-unes des méthodes d'attaque de mots de passe les plus utilisées.

1.1. L'utilité des mots de passe

Les mots de passe jouent un rôle essentiel en matière de sécurité, constituant des éléments cruciaux pour protéger l'accès à divers systèmes, comptes et informations sensibles. Ils servent de barrière d'entrée, permettant uniquement aux utilisateurs autorisés d'accéder aux ressources sensibles.

En somme, Les mots de passe jouent un rôle crucial en empêchant l'accès non autorisé et en garantissant la confidentialité des informations sensibles, ce qui en fait un élément essentiel pour la sécurité des systèmes informatiques.

1.2. Comment protéger les mots de passe

Pour garantir la sécurité des mots de passe, les méthodes de stockage diffèrent de celles des données ordinaires. Les mots de passe sont hachés à l'aide de fonctions de hachage spécifiques telles que MD5, SHA-1 et SHA-256. Ces fonctions de hachage appliquent des algorithmes complexes aux mots de passe, produisant une chaîne de caractères unique et difficile à déchiffrer.

Les fonctions de hachage sont caractérisées par leur nature non réversible, ce qui signifie qu'il est impossible de retrouver les mots de passe originaux à partir du hachage. Cependant, il est possible de calculer le hachage de chaque mot de passe. Cette propriété assure que même si le hachage est compromis, les mots de passe d'origine restent sécurisés, car ils ne peuvent pas être récupérés à partir du hachage.

1.3. Fonctions de hachage

Une fonction de hachage est typiquement une fonction qui, pour un ensemble de très grande taille (théoriquement infini) et de nature très diversifiée, va renvoyer des résultats aux spécifications précises (en général des chaînes de caractère de taille limitée ou fixe) optimisées pour des applications particulières. Une fonction de hachage est uniformisée, tester, mesurer, efficace, universaliser, applicable et déterministe.

Voici les fonctions de hachage qu'on a utilisé dans notre logiciel :

MD5 (Message Digest Algorithm 5)

Définition : MD5 est un algorithme de hachage cryptographique qui produit un condensat de 128 bits (16 octets) à partir d'une entrée de taille arbitraire.

Utilisation : Historiquement utilisé pour vérifier l'intégrité des données et pour le stockage sécurisé de mots de passe. Cependant, il est aujourd'hui considéré comme peu sûr pour de telles applications en raison de vulnérabilités connues.

SHA-1 (Secure Hash Algorithm 1)

Définition : SHA-1 est un algorithme de hachage cryptographique qui produit un condensat de 160 bits (20 octets) à partir d'une entrée de taille arbitraire.

Utilisation : Autrefois largement utilisé pour la vérification de l'intégrité des données et les signatures numériques, mais il est désormais obsolète en raison de vulnérabilités découvertes, et son utilisation est déconseillée.

SHA-256 (Secure Hash Algorithm 256-bit)

Définition : SHA-256 est un algorithme de hachage cryptographique qui produit un condensat de 256 bits (32 octets) à partir d'une entrée de taille arbitraire.

Utilisation : Très utilisé dans de nombreuses applications sécurisées, la cryptomonnaie, la vérification de l'intégrité des fichiers, etc. Il est considéré comme sûr pour la plupart des applications cryptographiques.

SHA-512 (Secure Hash Algorithm 512-bit)

Définition : SHA-512 est un algorithme de hachage cryptographique qui produit un condensat de 512 bits (64 octets) à partir d'une entrée de taille arbitraire.

Utilisation : Principalement utilisé dans des applications où une sécurité accrue est nécessaire, les systèmes de gestion de mot de passe sécurisé, et d'autres applications où une longueur de condensat plus grande est préférable pour des raisons de sécurité.

Remarque

Il existe des collisions, où deux entrées différentes produisent le même condensat (hash). En d'autres termes, deux jeux de données différents génèrent le même résultat de hachage.

Les fonctions de hachage sécurisées minimisent les risques de collisions, mais elles ne peuvent pas les éliminer complètement. Des algorithmes plus faibles ou obsolètes comme MD5 et SHA-1 sont particulièrement vulnérables aux collisions, tandis que des algorithmes plus récents et plus robustes comme SHA-256 et SHA-512 sont conçus pour être résistants aux collisions.

Dans le cadre de notre projet nous avons essayé de trouver une collision mais malheureusement ce n'est pas le cas.

1.4. Attaques des mots de passe

Les fonctions de hachage sont irréversibles, rendant impossible la récupération des mots de passe originaux à partir de leur hachage. Cependant, les attaquants disposent de méthodes spécifiques pour déterminer les mots de passe à partir de leur hachage. Parmi ces méthodes d'attaque, nous trouvons l'attaque par dictionnaire, l'attaque par force brute, et l'attaque par table arc-en-ciel. Dans le prochain chapitre, nous examinerons ces attaques en détail et les mettrons en œuvre pour mieux comprendre leur fonctionnement et leur efficacité.

Chapitre 2 : Conception et Implémentation

Dans ce chapitre, nous abordons de manière approfondie l'implémentation des attaques informatiques en détaillant leur fonctionnement, les algorithmes utilisés, leur complexité et leur efficacité. Nous avons développé ces attaques en Python et conçu un système appelé <<Arc-hash>> qui regroupe différentes méthodes d'attaque (force brute, dictionnaire, combinaison force brute-dictionnaire, arc-en-ciel), utilisant quatre fonctions de hachage distinctes (MD5, SHA-1, SHA-256 et SHA-512), Pourquoi <<Arc-hash>> ?

- Arc : Représente la réversibilité.

- Hash : Fait référence aux fonctions de hachage.

Ainsi, notre programme, nommé <<Arc-hash>>, tente de retrouver le mot de passe à partir de son hachage, ce qui signifie qu'il réalise une forme de réversibilité des fonctions de hachage.

2.1.Force brute

Une attaque par force brute utilise la méthode de l'essai-erreur pour deviner les identifiants d'un utilisateur.

Méthodologie

Essai-Erreur : Les attaquants essaient systématiquement toutes les combinaisons possibles de mots de passe ou de clés.

Automatisation : consiste à utiliser des scripts ou des logiciels automatisés qui tester des centaines de milliers, voire des millions de combinaisons en peu de temps.

Puissance de Calcul : Pour être efficaces, ces attaques nécessitent souvent une puissance de calcul élevée. Les attaquants utilisent des ordinateurs puissants ou des réseaux de machines (botnets) pour accélérer le processus.

2.1.1. Types d'attaques par force brute

Chaque attaque par force brute peut utiliser différentes méthodes pour découvrir des données confidentielles. Il existe plusieurs types de méthodes de force brute, notamment :

a) Attaques par force brute simples

L'attaque par force brute simple consiste à tester toutes les combinaisons possibles une à une, pour trouver un mot de passe ou une clé.

Après que l'utilisateur a fourni le hachage, le code vérifie si la taille du hachage correspond à la méthode de hachage choisie. Si c'est le cas, le code commence à générer des combinaisons de chaîne de caractères de taille 6 à partir d'une liste prédéfinie. Pour chaque combinaison possible, le code calcule le hachage et le compare avec le hachage fourni au début. Si les hachages sont égaux,

le code arrête la génération et affiche le mot de passe correspondant. Sinon, il continue de générer de nouvelles combinaisons.

Voici un pseudo-code illustrant le processus de l'attaque par force brute pour trouver un mot de passe à partir de son hachage :

```
for length in range (6, 13):
    for combination in generate_combinations(characters, length):
        calculated_hash = calculate_hash(combination, hash_type)
        test_text.insert(tk.END, f"Testing combination: {combination},
Hash: {calculated_hash}\n")
        test_text.update()
        if calculated_hash == hash_value:
            result_text.insert(tk.END, f"Password found: {combination}\n")
            result_text.tag_configure("found", foreground="green")
            result_text.tag_add("found", "1.0", "end")
            return
        test_text.delete(1.0, tk.END)

result_text.insert(tk.END, "Password not found.\n")
result_text.tag_configure("not_found", foreground="red")
result_text.tag_add("not_found", "1.0", "end")
```

2.1.2. Avantage

a) Exhaustivité

L'attaque par force brute est exhaustive, ce qui signifie qu'elle garantit la découverte de la solution si elle est suffisamment persistante. Elle explore toutes les combinaisons possibles de caractères jusqu'à trouver la bonne.

b) Simplicité

C'est une technique relativement simple à mettre en œuvre.

c) Efficacité Contre les Mots de Passe Faibles

Elle est particulièrement efficace contre les mots de passe faibles, car ces derniers sont souvent trouvés très rapidement dans le cadre des premières tentatives.

2.1.3. Inconvénients

Temps et complexité

L'attaque par force brute peut prendre énormément de temps, en particulier pour les mots de passe ou les clés de chiffrement complexes avec un grand espace de recherche. Les mots de passe complexes sont généralement composés de divers caractères, y compris des lettres majuscules et minuscules, des chiffres, et éventuellement des caractères spéciaux. La présence de cette diversité

augmente exponentiellement le nombre total de combinaisons possibles, rendant l'attaque par force brute plus difficile et prenant plus de temps.

De plus, la longueur du mot de passe est un facteur critique. Plus le mot de passe est long, plus il y a de combinaisons possibles à tester, ce qui augmente de manière exponentielle le temps nécessaire pour mener à bien une attaque par force brute. Par exemple, un mot de passe de 6 caractères à un espace de recherche beaucoup plus petit qu'un mot de passe de 8 caractères.

Voici une estimation de nombre de combinaisons possibles d'un mot de passe de différentes tailles en utilisant l'ensemble de caractères donné dans la liste :

Tableau 1: Estimation des Combinaisons de Mots de Passe par Taille.

Taille de mot de passe	Les valeurs
6	$77^6 \approx 2.084 \times 10^{11}$
7	1.604×10^{13}
8	1.235×10^{15}
9	9.515×10^{16}
10	7.326×10^{18}
11	5.641×10^{20}
12	4.343×10^{22}

```
characters =  
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!-  
_ "#$%&* , . = ? ^ @ '
```

Dans notre cas, le nombre de caractères est 77 de (26 lettres minuscules + 26 lettres majuscules + 10 chiffres + 15 caractères spéciaux).

a) Estimation du temps maximum requis pour découvrir des mots de passe de tailles différentes :

L'objectif principal de cette estimation est de déterminer le temps maximum nécessaire pour trouver un mot de passe. Cela permet d'avoir une idée du temps nécessaire pour mener à bien une attaque par force brute sur des mots de passe de différentes longueurs et complexités.

Nous avons développé un code qui génère toutes les combinaisons possibles de caractères pour des mots de passe de différentes longueurs, allant de 6 à 12 caractères. Ensuite, nous avons compté le nombre de combinaisons générées dans un délai d'une minute. Après plusieurs tests pour confirmer cette valeur, nous avons constaté qu'environ 252 281 449 combinaisons sont générées en une minute.

En utilisant cette valeur comme référence, nous avons calculé le temps nécessaire pour générer toutes les combinaisons possibles pour chaque taille de mot de passe, de 6 à 10 caractères. Les résultats obtenus sont consignés dans un tableau pour évaluer la complexité de la génération de mots de passe en fonction de leur taille.

Tableau 2: Durée Maximale de craquer des Mots de Passe selon la Taille.

Taille de mot de passe	6	7	8	9	10
Temps Maximum (en jours)	0.57	44.17	3401.55	261920.03	20167842.58

Ces données fournissent un aperçu du temps nécessaire pour trouver des mots de passe de différentes tailles, ce qui peut aider à évaluer la robustesse des systèmes de sécurité en fonction de la complexité des mots de passe.

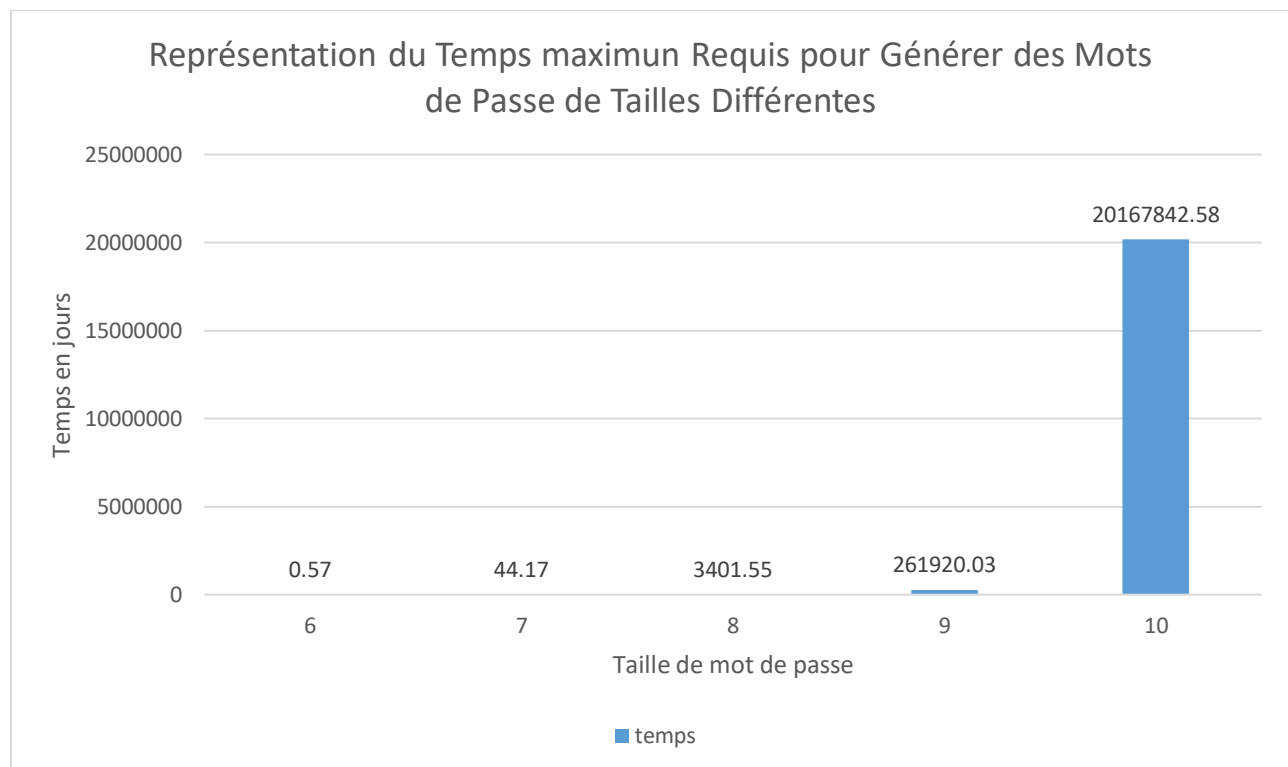


Figure 1: Temps de craquage des mots de passe par force brute.

b) Nécessité d'un PC Robuste

La méthode de génération exhaustive des combinaisons de mots de passe nécessite un ordinateur puissant pour effectuer les calculs efficacement, ce qui peut être contraignant en termes de ressources matérielles. Cette exigence en termes de matériel peut représenter un obstacle pour les utilisateurs disposant d'équipements informatiques moins performants.

2.2. Attaques par dictionnaire

L'attaque par dictionnaire est une méthode courante où les cybercriminels utilisent des listes de mots de passe prédéfinies (dictionnaire des mots de passe) pour tenter d'accéder à des comptes en ligne sécurisés.

Dans le cadre de notre projet, nous avons exploré deux types d'attaques par dictionnaire : l'attaque simple et l'attaque avec variation.

a) Dictionnaire des mots de passe

Un dictionnaire des mots de passe est un fichier textuel (.txt) qui regroupe des mots soigneusement sélectionnés. Ces mots sont choisis après des études et des analyses statistiques, par exemple, un dictionnaire des mots de passe les plus répandus en Algérie, en Afrique, ou dans une université particulière. Ainsi, ces mots de passe sont spécifiquement ciblés pour leur pertinence dans un contexte donné.

Ce fichier contient une liste de mots de passe potentiellement utilisés, basée sur des informations contextuelles telles que des données démographiques, des tendances culturelles, ou des habitudes de création de mots de passe. L'objectif est d'aider les spécialistes de la sécurité informatique à identifier et à protéger contre les vulnérabilités potentielles liées à l'utilisation de mots de passe courants ou prévisibles.

Le processus de création de ces dictionnaires implique généralement l'analyse de grands ensembles de données pour identifier les schémas et les tendances dans les choix de mots de passe. Ensuite, des algorithmes sont utilisés pour générer une liste de mots de passe potentiels qui sont ensuite triés et inclus dans le dictionnaire.

Il est essentiel de mettre à jour régulièrement ces dictionnaires pour refléter les évolutions des tendances et des habitudes de création de mots de passe. De plus, les gestionnaires de mots de passe et les systèmes de sécurité informatique peuvent utiliser ces dictionnaires pour renforcer la sécurité en détectant et en empêchant l'utilisation de mots de passe faibles ou prédictibles.

2.2.1. Attaque par dictionnaire (simple)

L'attaque par dictionnaire est une méthode utilisée pour accéder à des comptes sécurisés en exploitant des mots de passe faibles ou prévisibles. Elle repose sur l'utilisation de dictionnaires de mots de passe, où chaque mot ou combinaison de mots est testé pour trouver une correspondance avec le mot de passe ciblé. L'algorithme sous-jacent à cette attaque peut être décrit comme suit :

Tout d'abord, le mot de passe fourni par l'utilisateur est haché. Ensuite, l'algorithme ouvre le fichier du dictionnaire contenant une liste de mots de passe préétablis. Une boucle itérative est alors initiée pour hacher chaque mot du dictionnaire et stocker ces valeurs hachées dans un autre fichier.

Ensuite, une seconde boucle est mise en place pour comparer le mot de passe haché récupéré avec les mots de passe hachés stockés dans le fichier du dictionnaire. Si une correspondance est trouvée, cela signifie que le mot de passe est présent dans le dictionnaire et donc potentiellement vulnérable. Dans ce cas, une notification est affichée à l'écran pour indiquer que le mot de passe a été trouvé dans le dictionnaire.

2.2.2. Attaque par dictionnaire avec des variations

Cette méthode est similaire à l'attaque par dictionnaire simple, mais elle intègre des variations pour chaque mot de passe du dictionnaire en utilisant des règles spéciales. Ces règles définissent des substitutions de caractères pour rendre les mots de passe plus complexes et moins prévisibles pour les attaquants. Voici une explication détaillée des règles :

Tableau 3: tableau de variations

Caractère	Remplacé par
a	@ , 4 , A
e	3 , E
o	0 , O
i	1 , I
s	\$, 5 , S
t	7 , T

Par exemple le mot "password" pourrait devenir "p@ssw0rd" en remplaçant 'a' par '@', 's' par '\$', et 'o' par '0'. Cette technique augmente la complexité des mots de passe en utilisant des substitutions de caractères qui ressemblent visuellement aux caractères d'origine mais sont moins prévisibles pour les attaquants. Elle permet ainsi de renforcer la sécurité des comptes en ligne contre les attaques par dictionnaire en introduisant des variations qui échappent aux mots de passe couramment utilisés.

Dans cette méthode, les variations de chaque mot de passe du dictionnaire sont stockées dans un tableau. Ce tableau est ensuite parcouru et chaque mot est haché. En parallèle, le mot de passe récupéré est également haché. Ensuite, chaque mot haché du tableau est comparé avec le mot de passe haché récupéré. Si une correspondance est trouvée, cela signifie que le mot de passe est présent dans le dictionnaire, et donc potentiellement vulnérable. Dans ce cas, une notification est générée pour indiquer que le mot de passe a été trouvé dans le dictionnaire.

2.2.3. Avantages

- a) **Efficacité** : Cette méthode peut être rapide et efficace pour deviner les mots de passe faibles ou couramment utilisés.
- b) **Accessibilité** : Les dictionnaires de mots de passe sont facilement disponibles en ligne, ce qui facilite la mise en œuvre de cette attaque.

2.2.4. Inconvénients

- a) **Limitations des dictionnaires** : Les mots de passe complexes ou uniques ne seront pas présents dans les dictionnaires, réduisant ainsi l'efficacité de l'attaque.
- b) **Temps nécessaire** : Pour des mots de passe complexes, l'attaque par dictionnaire peut prendre beaucoup de temps, surtout si des variations sont nécessaires.

2.2.5. La complexité

-La complexité exacte du programme dépendra de plusieurs facteurs et peut varier en fonction de l'implémentation spécifique de certaines opérations. Cependant, nous pouvons donner une estimation de la complexité en fonction des opérations principales du programme :

1-Chargement du dictionnaire : La complexité de cette opération est généralement proportionnelle à la taille du fichier. Si le fichier du dictionnaire fait T o, cela signifie qu'il contient environ. La complexité peut donc être approximée à $O(T)$, où T est la taille du fichier en octets.

2-Génération des variations de mots de passe : Pour chaque mot de passe dans le dictionnaire, le génère plusieurs variations. La complexité de cette opération dépend donc de la longueur moyenne des mots de passe dans le dictionnaire, ainsi que du nombre moyen de variations générées pour chaque mot de passe. Si l'on suppose que la longueur moyenne des mots de passe est m et qu'il y a en moyenne v variations par mot de passe, la complexité peut être approximée à $O(n * m * v)$, où n est le nombre de mots de passe dans le dictionnaire.

3-Recherche du hachage MD5 dans le dictionnaire : Cette opération a une complexité moyenne de $O(1)$ pour chaque recherche, car le dictionnaire est stocké sous forme d'ensemble, ce qui permet une recherche rapide.

4-Recherche du hachage MD5 dans les variations de mots de passe : La complexité de cette opération dépend du nombre total de variations de mots de passe générées. Si le nombre total de variations est d'environ x , alors la complexité peut être approximée à $O(x)$.

En résumé, la complexité totale du programme dépend de la combinaison de ces opérations. La complexité exacte peut être difficile à déterminer sans plus de détails sur la taille du dictionnaire, la longueur moyenne des mots de passe, le nombre de variations générées, etc.

2.2.6. Conclusion

L'efficacité de cette attaque dépend de la qualité du dictionnaire utilisé, ainsi que de la complexité des mots de passe choisis par les utilisateurs ciblés. Plus le dictionnaire est large et diversifié, plus

il est susceptible de réussir à deviner les mots de passe. Par conséquent, les gestionnaires de sécurité informatique doivent prendre des mesures pour contrer cette attaque, telles que l'imposition de politiques de mot de passe robustes et l'utilisation de techniques de cryptage avancées.

Cette méthode permet aux attaquants de tester rapidement une grande quantité de mots de passe potentiels, en exploitant la faiblesse des mots de passe courants ou prévisibles.

2.3. Attaque qui combine Brute force et dictionnaire

2.3.1. Principe

Phase 1 : Préparation

- **Génération de fichier de combinaisons** : Un fichier texte exhaustif est créé, contenant toutes les combinaisons possibles de caractères, y compris des lettres minuscules et majuscules ainsi que des chiffres.
- **Hachage des mots de passe** : Pour chaque combinaison de caractères générée, son hachage correspondant est calculé et stocké dans le fichier, en utilisant deux algorithmes de hachage différents : MD5 et SHA1. Ceci permet de créer deux fichiers distincts, un pour chaque algorithme de hachage.

Phase 2 : Attaque par comparaison

- **Saisie du hachage cible** : L'utilisateur fournit le hachage du mot de passe qu'il souhaite déchiffrer.
- **Comparaison des hachages** : Le programme compare le hachage cible avec l'ensemble des hachages stockés dans les fichiers MD5 et SHA1.
- **Récupération du mot de passe** : Si une correspondance est trouvée dans l'un des fichiers, le mot de passe associé à ce hachage correspondant est révélé.

2.3.2. Avantages

- **Efficacité accrue** : En combinant les approches par force brute et par dictionnaire, l'attaque couvre un large spectre de mots de passe potentiels, augmentant ainsi les chances de succès.
- **Rapidité optimisée** : L'utilisation de fichiers de hachage précalculés permet d'accélérer considérablement le processus de comparaison, par rapport à une attaque par force brute classique.
- **Flexibilité** : La méthode peut être adaptée pour cibler des mots de passe de différentes longueurs et complexités.

2.3.3. Inconvénients

- **Dépendance à la taille du dictionnaire** : L'efficacité de l'attaque dépend de la taille et de la qualité du dictionnaire de mots de passe utilisé. Un dictionnaire incomplet peut limiter les chances de réussite.

- **Temps de préparation conséquent** : La génération initiale des fichiers de hachage peut être longue et gourmande en ressources, en fonction de la taille du dictionnaire et de la puissance de calcul disponible.
- **Nécessité d'un espace de stockage conséquent** : Les hachages pré calculés nécessitent une quantité importante d'espace de stockage, en fonction de la longueur du mot de passe et de la complexité de la fonction de hachage. Par exemple, en pratique pour un mot de passe de 6 caractères, l'espace de stockage peut dépasser 600 Go.

Remarque : Dans le cadre présent, nous avons opté pour la création d'un fichier composé de mots allant de 1 à 4 caractères.

En effet, l'ajout de caractères supplémentaires engendrerait une taille de fichier susceptible d'atteindre plusieurs téraoctets, ce qui s'avère irréalisable dans nos conditions actuelles

2.3.4. Complexité

a) En terme de temps :

Dans cette méthode d'attaque, le mot de passe est découvert en un laps de temps relativement court. Nous avons tenté de déchiffrer des mots de passe hachés en MD5 et SHA-1, avec une longueur variant de 1 à 4 caractères. Voici les résultats :

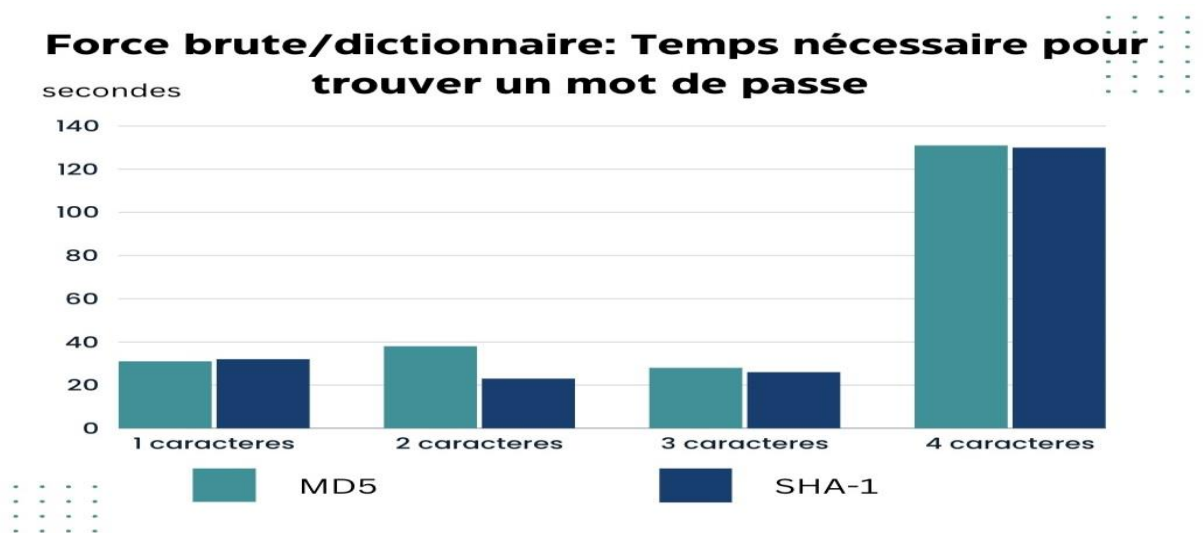


Figure 2: Temps de Génération des Mots de Passe par Force brute/dictionnaire

L'algorithme de l'attaque force brute/dictionnaire a une complexité temporelle de $O(n^2)$

b) En terme de mémoire

Cette méthode d'attaque nécessite une quantité considérable d'espace de stockage. Nous avons généré un fichier avec toutes les combinaisons possibles de mots de quatre caractères, ce qui a abouti à un fichier de 670 Mo. Lorsque nous avons essayé d'inclure des mots plus longs, la taille du fichier a augmenté pour atteindre plusieurs téraoctets.

Par conséquent, on peut conclure que cette technique d'attaque de mots de passe est particulièrement exigeante en termes d'espace de stockage.

2.3.5. Contre-mesures contre l'attaque brute force / dictionnaire

- **Créer des mots de passe longs** : Il est recommandé d'utiliser des mots de passe qui mélangent plusieurs lettres (majuscules et minuscules), chiffres et caractères spéciaux. Cela augmente la complexité du mot de passe et le rend plus difficile à deviner ou à craquer.
- **Ajouter des "salts"** : Un "salt" est une donnée aléatoire qui est utilisée comme une valeur supplémentaire dans le processus de hachage du mot de passe. L'ajout d'un "salt" aux mots de passe avant leur hachage peut aider à protéger les mots de passe.

2.4. Arc en ciel

2.4.1. Définition d'une table arc en ciel

Une table arc-en-ciel est une structure de données contenant une multitude de mots de passe associés à leur valeur de hachage, elle est sauvegardée sous forme d'un fichier. Les attaquants s'en servent pour cracker des mots de passe.

Les tables arc-en-ciel permettent généralement de réduire le temps et la mémoire nécessaires à l'attaque, contrairement aux attaques par force brute qui requièrent beaucoup de temps et aux attaques par dictionnaires qui nécessitent beaucoup de mémoire.

Lors de la génération d'une table arc-en-ciel, un premier mot de passe est haché. À partir de l'empreinte obtenue, une fonction de réduction calcule un nouveau mot de passe, qui va lui aussi être haché puis réduit en un nouveau mot de passe. Cette opération est répétée un nombre de fois défini pour former une chaîne, jusqu'à obtenir une valeur finale.

Les tables arc en ciel peuvent être générées automatiquement dans certaines distributions de linux, ou même en utilisant des outils comme Rainbow Crack.

À noter que les tables arc-en-ciel peuvent également être utilisées par des experts en cyber sécurité pour identifier des failles ou effectuer des tests de sécurité.

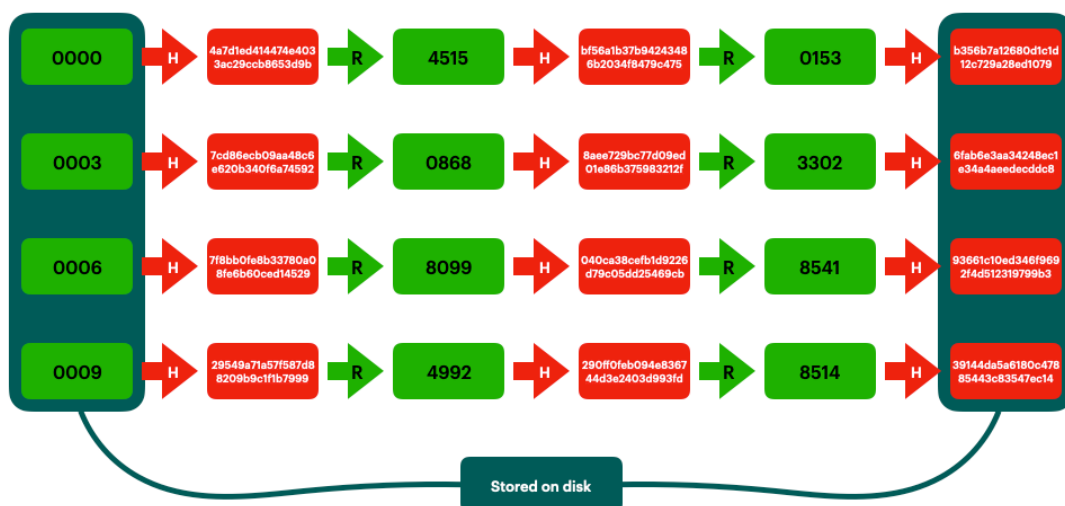


Figure 3: Mécanisme de Stockage et de Récupération de Données Hachées

2.4.2. Principe de l'attaque par table arc en ciel

a) Phase de préparation

- La création d'une table arc-en-ciel s'effectue en sélectionnant des mots aléatoires d'une taille définie. La taille des mots est généralement comprise entre 6 et 12 caractères.
- Chaque mot est ensuite haché, puis soumis à une fonction de réduction qui permet de réduire la taille des hachages tout en conservant leur unicité.
- Ce processus de hachage et de réduction est répété autant de fois que nécessaire. Dans le cas présent, il est répété 1000 fois.
- Le mot de passe initial et le 1000ème hachage réduit sont stockés dans un fichier

b) Phase d'exécution

- L'utilisateur saisit le hachage du mot de passe qu'il souhaite retrouver.
- L'algorithme commence par réduire le hachage saisi et le compare avec les hachages présents dans le fichier.
- Si une correspondance est trouvée, le mot de passe initial dans le fichier est haché et réduit 999 fois jusqu'à obtenir l'avant-dernier hachage réduit, qui est ensuite renvoyé comme le mot de passe recherché.
- En l'absence de correspondance dans le fichier, le programme commence à hacher et à réduire les mots de passe aléatoires stockés dans le fichier, 1000 fois pour chaque mot.
- À chaque étape, le hachage réduit est comparé au hachage saisi. Si les deux hachages sont identiques, le mot de passe dont le hachage résultant est le hachage réduit est renvoyé.
- Si aucune correspondance n'est trouvée après avoir parcouru toutes les lignes du fichier, le programme passe au fichier suivant qui contient des mots plus longs.

2.4.3. Remarques

- Chaque fichier contient 10000 mots.
- Chaque fichier contient une table arc-en-ciel pour une longueur de mot spécifique (6 caractères, 7 caractères, ..., 12 caractères)

2.4.4. Avantages

- **Rapidité** : L'attaque par table arc-en-ciel est beaucoup plus rapide que les attaques par force brute ou par dictionnaire, car son approche est particulièrement efficace et permet de tester un large nombre de mots de passes dans un temps réduit.
- **Efficacité** : L'attaque par table arc-en-ciel est plus efficace que les attaques par dictionnaire, car elle a plus de chances de trouver le mot de passe cible, même si celui-ci n'est pas un mot du dictionnaire.

2.4.5. Inconvénients

- **Fichier de table arc en ciel volumineux** : Lors de la génération d'une table arc en ciel contenant plusieurs milliers ou millions de mot de passes possible, la taille du fichier peut augmenter considérablement.
- **Risque de non-correspondance** : Il existe une possibilité que le mot de passe cible ne soit pas présent dans la table arc-en-ciel, ce qui rend l'attaque inefficace. Comme par exemple dans le cas d'un mot de passe salé, car le hachage contenant un « salting » n'est pas présent dans la table.

2.4.6. Complexité

L'attaque par table arc-en-ciel est basée sur un compromis entre l'espace et le temps, ce qui signifie qu'elle utilise moins de temps de traitement mais plus d'espace de stockage par rapport à une attaque par force brute.

a) En terme de mémoire

L'attaque par table arc-en-ciel nécessite un espace de stockage convenable comparé aux autres méthodes d'attaque de mots de passe que nous avons expérimentées tels que l'attaque qui combine force brute et dictionnaire.

La taille des fichiers augmente chaque fois qu'on accroît la longueur des mots de passes et/ou le nombre de mots par fichier.

Voici un graphe qui représente l'espace de stockage requis pour notre algorithme :

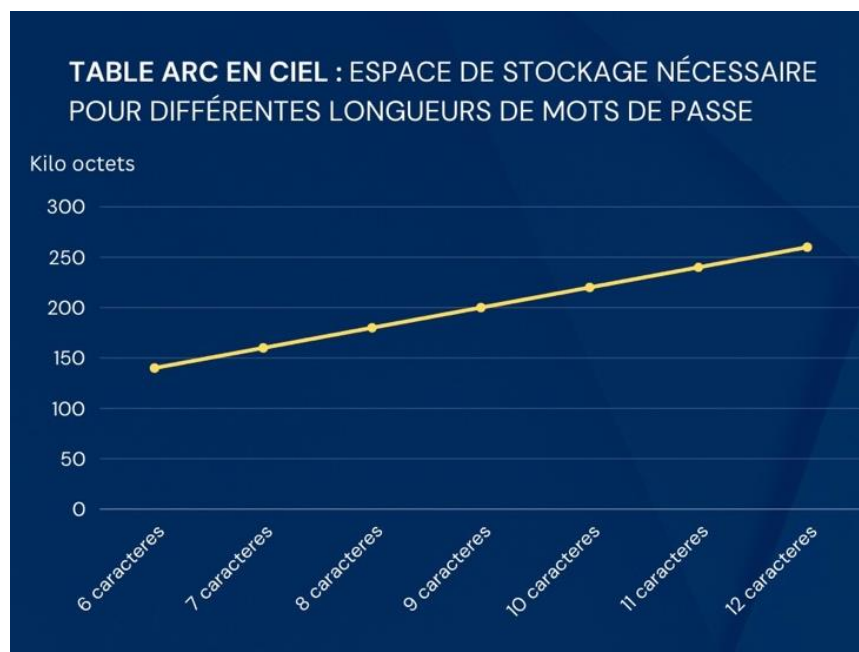


Figure 4: Espace de Stockage Requis pour 'Algorithme

Dans une notre cas, nous avons généré 10000 mots par fichier, cependant il est toujours possible d'augmenter le nombre de mots si nécessaire pour une méthode d'attaque plus efficace, ce qui augmentera l'espace de stockage nécessaire pour ces fichiers.

b) En terme de temps

Dans le **meilleur des cas**, la complexité de cet algorithme est de $O(n)$, ce qui signifie que le temps nécessaire pour trouver le mot de passe augmente linéairement.

Cependant, dans le **pire des cas**, la complexité peut atteindre $O(n^2)$, le temps nécessaire augmente donc de manière quadratique.

Par conséquent, nous pouvons conclure que cette attaque produit généralement des résultats plus rapidement qu'une attaque par dictionnaire ou par force brute, souvent en quelques secondes ou minutes là où d'autres méthodes peuvent prendre beaucoup plus de temps.

2.4.7. Contre-mesures contre l'attaque par table arc-en-ciel

Plusieurs contre-mesures peuvent être mises en place pour protéger les mots de passe contre l'attaque par table arc-en-ciel :

- **Utilisation de mots de passe longs et complexes :** Les mots de passe longs et complexes sont plus difficiles à craquer par les attaques par table arc-en-ciel, car ils ne sont pas présents dans les tables pré calculées.
- **Salage des mots de passe :** Le salage des mots de passe consiste à ajouter une valeur aléatoire au mot de passe avant de le hacher. Cela permet de générer des hachages uniques pour des mots de passe identiques, ce qui rend l'attaque par table arc-en-ciel inefficace.

2.5.Comparaison des méthodes d'attaques

Caractéristique	Attaque par force brute	Attaque par dictionnaire	Attaque arc en ciel
Méthode	Essai de toutes les combinaisons possibles	Utilise un ensemble de mots de passe courants	Utilise des tables précalculées de hachages de mots de passe
Efficacité contre des mots de passes longs et complexes	Lente mais peut réussir	Inefficace	Peut réussir, mais nécessite des tables arc-en-ciel plus grandes
Temps d'attaque	Long	Rapide pour les mots de passes faibles	Plus rapide que force brute et dictionnaire
Mémoire nécessaire	Faible	Grande	Moyenne
Ressources	Puissance de calcul importante	Accès a un dictionnaire de mots de passes	Acces a une table arc en ciel et puissance de calcul

Figure 5: Comparaison des méthodes d'attaques

Chapitre III : Test et Exploitation

Les méthodes d'attaque informatique constituent un domaine crucial de la sécurité des systèmes d'information. Dans ce chapitre, nous explorerons les différentes méthodes d'attaques et leur mise en œuvre dans des environnements populaires tels que Linux et Microsoft Office.

3.1.Office

L'attaque de mots de passe sur des fichiers Office est une méthode utilisée pour contourner la protection des documents Microsoft Word, Excel ou PowerPoint en trouvant le mot de passe associé. Des outils spécialisés comme John the Ripper ou Hashcat sont souvent utilisés pour automatiser ce processus.

3.1.1. En utilisant l'outil "John the Ripper"

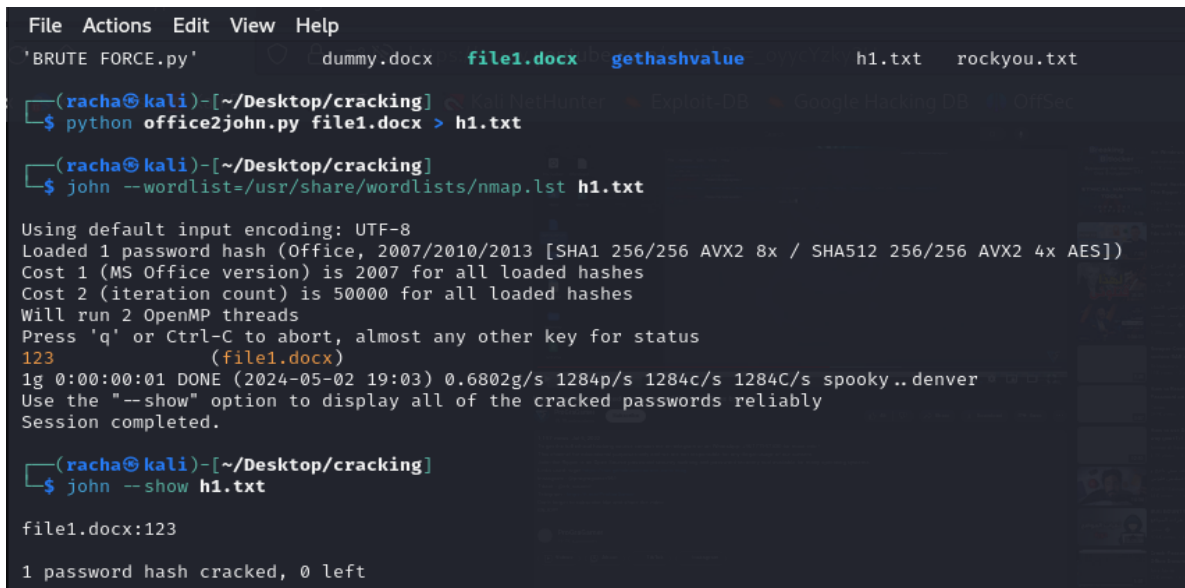
John the Ripper est un outil de craquage de mots de passe largement utilisé dans le domaine de la sécurité informatique. Il est conçu pour tester la robustesse des mots de passe en essayant différentes méthodes d'attaque, notamment la force brute et les attaques par dictionnaire. L'outil prend en charge une variété de formats de hachage de mots de passe et peut être utilisé pour analyser la sécurité des fichiers protégés par mot de passe.

Lorsqu'un fichier Word 2007, tel que "file1.docx", que j'ai créé et protégé par un mot de passe "123", il est souvent nécessaire d'extraire le hachage du mot de passe afin de tenter de le craquer. Sachant que, le processus de stockage des mots de passe des fichiers Microsoft Office implique généralement plusieurs étapes. Tout d'abord, le mot de passe est transformé en une forme de hachage cryptographique à l'aide d'un algorithme de hachage, tel que SHA-1. Ensuite, ce hachage est souvent combiné avec d'autres données, telles que des clés de chiffrement, pour former une représentation finale du mot de passe, qui est ensuite stockée dans le fichier. C'est là qu'intervient l'outil "office2john.py". Ce script est spécialement conçu pour extraire le hachage du mot de passe à partir de fichiers Microsoft Office, tels que les fichiers Word 2007 protégés. Lorsqu'il est exécuté, le script extrait le hachage du mot de passe du fichier spécifié et le stocke dans un fichier texte, par exemple "h1.txt".

Une fois que le hachage du mot de passe est extrait avec succès, on peut utiliser John the Ripper pour tenter de craquer le mot de passe. La commande "john --wordlist=/usr/share/wordlists/nmap.lst h1.txt" est un exemple de la manière dont John the Ripper peut être utilisé. Dans cette commande, "--wordlist=/usr/share/wordlists/nmap.lst" spécifie le fichier de liste de mots que John the Ripper utilisera pour essayer de deviner le mot de passe.

Lorsque la commande est exécutée, John the Ripper commence à essayer différentes combinaisons de mots de la liste fournie pour trouver le mot de passe correspondant au hachage extrait. Dans le cas où le mot de passe est trouvé avec succès, John the Ripper affiche le mot de passe craqué, comme illustré dans la sortie de la commande "john --show h1.txt". Dans cet exemple, le mot de passe "123" a été trouvé avec succès.

Ainsi, en utilisant une combinaison d'outils tels que office2john.py et John the Ripper, il est possible de tester la sécurité des mots de passe utilisés pour protéger les fichiers et les documents.



```
File Actions Edit View Help
'BRUTE FORCE.py' dummy.docx file1.docx gethashvalue h1.txt rockyou.txt

(racha@kali)-[~/Desktop/cracking]
$ python office2john.py file1.docx > h1.txt

(racha@kali)-[~/Desktop/cracking]
$ john --wordlist=/usr/share/wordlists/nmap.lst h1.txt

Using default input encoding: UTF-8
Loaded 1 password hash (Office, 2007/2010/2013 [SHA1 256/256 AVX2 8x / SHA512 256/256 AVX2 4x AES])
Cost 1 (MS Office version) is 2007 for all loaded hashes
Cost 2 (iteration count) is 50000 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
123 (file1.docx)
lg 0:00:00:01 DONE (2024-05-02 19:03) 0.6802g/s 1284p/s 1284c/s 1284C/s spooky..denver
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

(racha@kali)-[~/Desktop/cracking]
$ john --show h1.txt

file1.docx:123

1 password hash cracked, 0 left
```

Figure 6: Utilisation de John the Ripper pour Craquer des Mots de Passe Office.

3.1.2. En utilisant l'outil "hashcat"

Hashcat est un outil de récupération de mots de passe puissant et polyvalent, largement utilisé par les professionnels de la sécurité informatique et les chercheurs en sécurité. Il est conçu pour exécuter divers types d'attaques de récupération de mots de passe, y compris les attaques par force brute, par dictionnaire.

Dans l'exemple, Hashcat est utilisé pour tenter de retrouver un mot de passe associé à un hachage spécifique qui est stockée dans le fichier h1 hash, utilisé pour les fichiers Office, comme ceux utilisés dans Word 2007. Voici comment cela fonctionne en détail :

a) Choix d'attaque et de mode d'hachage

Une attaque par force brute est choisie avec l'option "-a 0", ce qui signifie que Hashcat va essayer toutes les combinaisons possibles de caractères pour trouver le mot de passe.

Le mode de hachage spécifique utilisé pour les fichiers Office 2007 est spécifié avec l'option "-m 9400". Chaque mode de hachage a un identifiant unique dans Hashcat, et "9400" est l'identifiant correspondant aux fichiers Office.

b) Fourniture de fichier de dictionnaire

Un fichier de dictionnaire contenant une liste de mots courants, appelé "sqlmap.txt", est fourni à Hashcat. Ce fichier contient une multitude de mots qui pourraient potentiellement être utilisés comme mots de passe.

c) Exécution de Hashcat

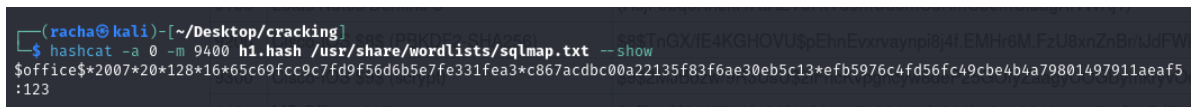
Hashcat commence alors son exécution, essayant chaque mot du dictionnaire et générant le hachage correspondant pour chaque mot. Il compare ensuite ce hachage généré avec le hachage fourni pour le mot de passe que vous essayez de retrouver.

d) Trouver le mot de passe

Lorsque Hashcat trouve un mot de passe dont le hachage correspond à celui fourni, il affiche ce mot de passe.

Dans votre exemple, Hashcat parvient à trouver le mot de passe "123" correspondant au hachage donné.

En résumé, Hashcat est un outil puissant et flexible qui peut être utilisé pour récupérer des mots de passe en utilisant diverses techniques d'attaque et en exploitant des dictionnaires de mots courants.



```
(racha@kali) - [~/Desktop/cracking]
$ hashcat -a 0 -m 9400 h1.hash /usr/share/wordlists/sqlmap.txt --show
$office$*2007*20*128*16*65c69fcc9c7fd9f56d6b5e7fe331fea3*c867acdbc00a22135f83f6ae30eb5c13*efb5976c4fd56fc49cbe4b4a79801497911aef5
:123
```

Figure 7: Méthodologie d'Utilisation de Hashcat pour Fichiers Office.

3.2. Attaque sur les mots de passe sous Linux

Pour comprendre les défis posés par les attaques sur les mots de passe sous Linux, il est essentiel de se pencher sur les mécanismes de hachage et de salage utilisés.

En examinant les fichiers "shadow" sur plusieurs versions d'Ubuntu et Kali Linux, on observe que le mot de passe haché est dans le format « \$id\$sel\$motdepassehaché ».

"\$id" est l'algorithme utilisé : sur Linux, "\$1\$" signifie MD5, "\$5\$" signifie SHA-256 et "\$6\$" signifie SHA-512...etc

On observe l'utilisation du type de hachage "\$6\$", associé à SHA-512, sur Ubuntu version 10.10. Bien que SHA-512 soit un algorithme de hachage capable de produire des hachages de 512 bits, on constate que les hachages obtenus sont inférieurs à cette longueur, car ils sont exprimés en base 64.

Cette observation nous conduit à explorer le concept de salage, une pratique courante dans la sécurisation des mots de passe.

Le sel, généré de manière aléatoire, est combiné au mot de passe avant d'être haché, ce qui rend la récupération du mot de passe d'origine plus difficile.

De plus, en réalisant des tests sur les hachages de mots de passe sous Linux, on constate qu'une simple conversion des hachages de leur représentation en base 64 vers la base 16 ne permet pas de retrouver les mots de passe d'origine.

Cette difficulté supplémentaire provient du fait que le hachage SHA-512 est exécuté 5000 fois, voire davantage dans certaines versions, rendant ainsi les attaques de récupération de mot de passe

extrêmement ardues. En effet, des versions peuvent même exécuter SHA-512 jusqu'à 250000 fois pour renforcer la sécurité et décourager toute tentative d'attaque.

Cette complexité supplémentaire vise à protéger les mots de passe des utilisateurs contre les attaques par force brute et par dictionnaire, renforçant ainsi la sécurité des systèmes Linux.

En somme, l'étude approfondie des mécanismes de hachage et de salage des mots de passe sous Linux met en lumière une approche robuste et sophistiquée en matière de sécurité. L'utilisation de techniques telles que le salage aléatoire et le hachage répété, combinée à des algorithmes de chiffrement puissants comme SHA-512, témoigne de l'engagement constant envers la protection des données des utilisateurs. Ces pratiques visent à décourager efficacement les attaques par force brute et par dictionnaire, renforçant ainsi la sécurité globale des systèmes Linux.

3.3.Test de logiciel

3.3.1. Attaque combiné brute force-dictionnaire

Cette figure représente le test d'une attaque qui combine force brute et dictionnaire.

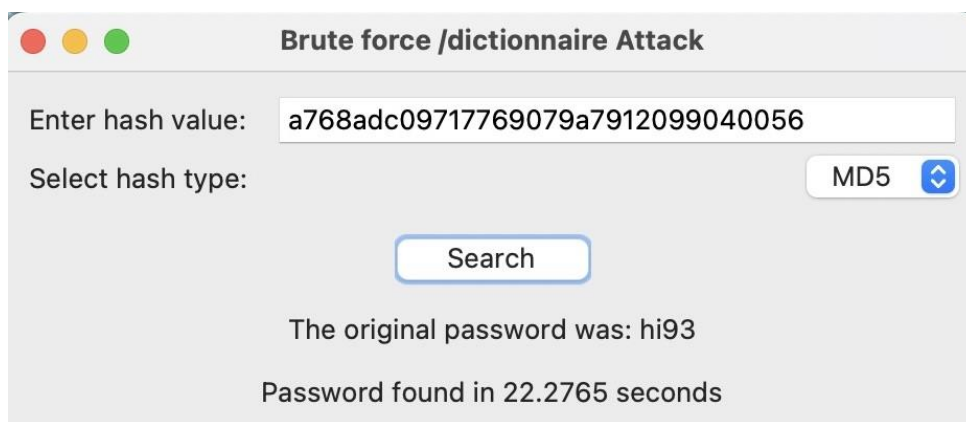


Figure 8 :Attaque combiné brute force/dictionnaire.

3.3.2. Attaque force brute

Cette figure représente le test d'une attaque par force brute.

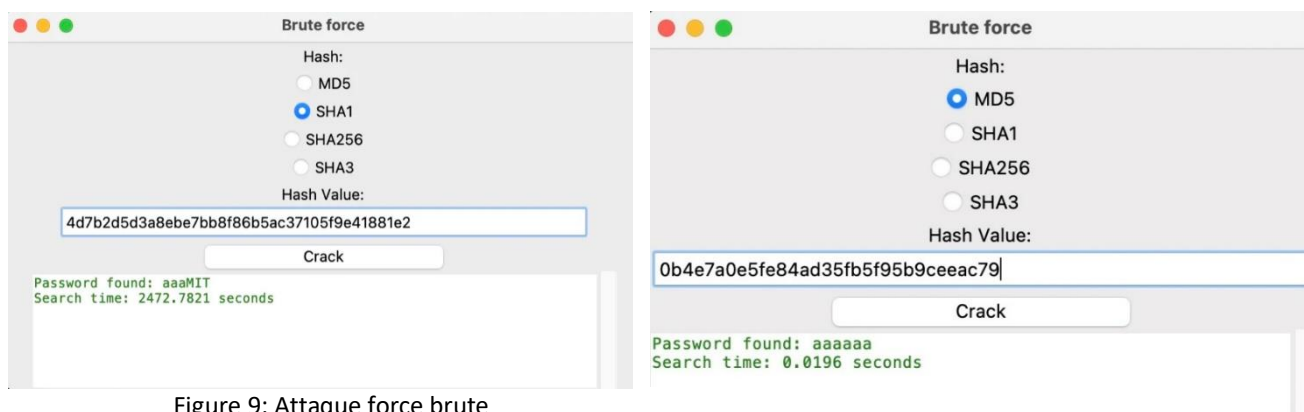


Figure 9: Attaque force brute

3.3.3. Attaque arc en

Cette figure représente le test d'une attaque par arc en ciel.

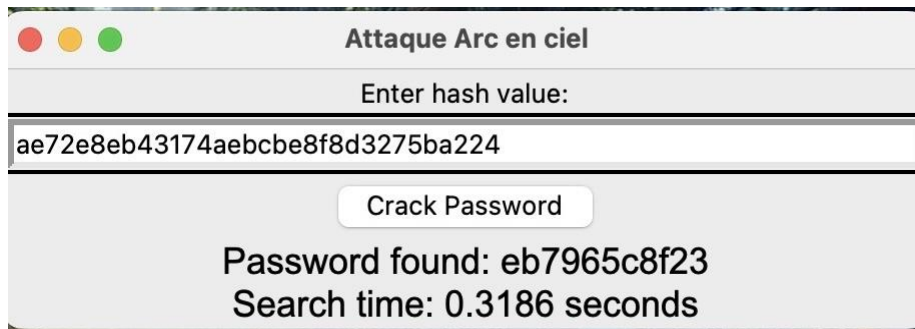


Figure 10: Attaque Arce en ciel

3.3.4. Attaque par dictionnaire

Cette figure représente le test d'une attaque par dictionnaire.

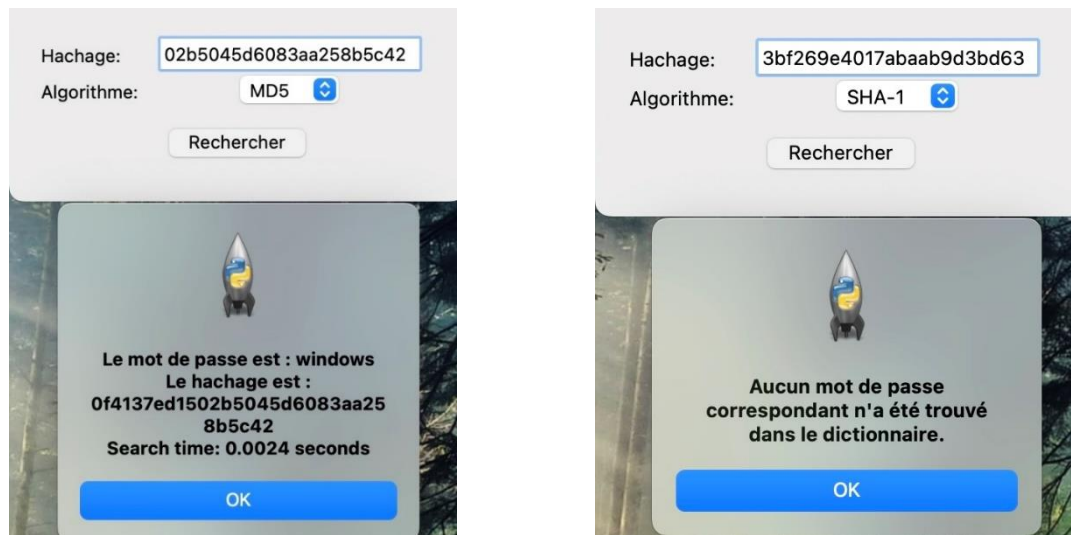


Figure 11: Attaque par dictionnaire

3.4. Attaque par dictionnaire avec des variations

Cette figure représente le test d'une attaque par dictionnaire avec des variations.



Figure 12: Attaque par dictionnaire avec des variations

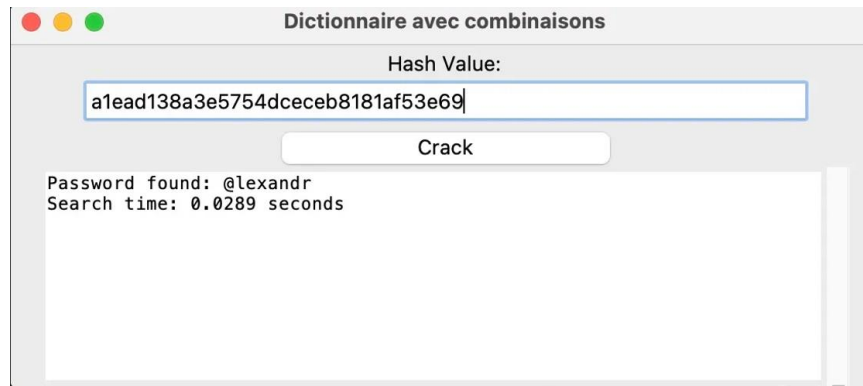


Figure 13: Attaque par dictionnaire avec des variations

3.5. Avantages de ce projet

- Développement de compétences de travail en équipe et renforcement des compétences collaboratives.
- Amélioration de la communication entre les membres et avec l'encadreur.
- Capacité à exprimer et à partager notre vision.
- Opportunité de découvrir et d'approfondir des connaissances dans un domaine nouveau pour nous : la cybersécurité.
- Apprentissage et utilisation de nouveau langage Python.
- Familiarisation avec Kali Linux.
- Apprentissage de nouvelles compétences techniques.

3.6. Division des tâches

Répartition équitable des tâches entre les membres de l'équipe.

- Attaque par force brute
- Attaque par dictionnaire simple
- Attaque par dictionnaire avec des variations
- Attaque combine force brute force / dictionnaire
- Attaque par table de hachage arc-en-ciel
- Partie des tests « Office »
- Partie des tests « Attaque sur les mots de passe sous Linux »
- Design de logiciel
- Préparation des slides de la présentation

Remarque : La répartition des tâches a renforcé notre cohésion d'équipe. En cas de difficulté, nous aidions le membre concerné ou demandions des conseils et des orientations supplémentaires à notre encadrant, M. Zaidi. Nous avons travaillé comme une seule entité, unissant nos efforts pour atteindre des objectifs communs

- Collaboration et coordination régulières.
- Assurer la cohérence du projet via des réunions (Meet) ou des échanges de messages.

3.4.Outils de communication utilisés

- Utilisation de GitHub pour le partage et la gestion du code.
- Utilisation de Discord pour la communication instantanée et la coordination.
- Utilisation de Google Docs pour la rédaction et le partage de documents.

3.5.Difficultés Rencontrées

- Un manque de connaissances dans le domaine de la sécurité des mots de passe.
- La gestion de données très volumineuses, sachant que la RAM maximale de nos ordinateurs est de 8 Go, ce qui peut entraîner une saturation de la mémoire.
- Durée de temps limitée : Gestion du projet en parallèle avec la charge des cours et des TDs.
- Nous avons été confrontés à une difficulté majeure lors la tentative de craquage des mots de passe sous Linux. Pour évaluer notre logiciel, nous avons déployé plusieurs distributions Linux telles que LMDE 6, Ubuntu 10.10 et Ubuntu 04.10, dans le but de trouver une version utilisant des fonctions de hachage plus faibles ou obsolètes comme MD5 et SHA-1.

Cependant, malgré nos nombreux tests et l'aide de notre encadrant, nous avons identifié un problème : les fonctions de hachage étaient appliquées plusieurs fois. Cette situation a rendu notre tâche complexe et a pris beaucoup de temps pour l'installation et les tests des différentes versions.

Conclusion générale

Ce projet pluridisciplinaire a permis de mettre en lumière l'importance de la robustesse des mots de passe dans le domaine de la sécurité informatique. Les attaques par mot de passe constituent une menace majeure pour la sécurité individuelle et collective car ces derniers protègent l'accès à nos données les plus sensibles.

C'est pour cela qu'on a développé notre logiciel, « Arc-Hash », qui est conçu pour évaluer la robustesse des mots de passe contre diverses formes de cyberattaques.

Il permet d'identifier les failles potentielles pour aider l'utilisateur à opter pour des mots de passe plus sûrs.

Cela est réalisé en soumettant le mot de passe à cinq types d'attaques différents :

- Attaque par force brute
- Attaque par dictionnaire
- Attaque par table arc-en-ciel
- Attaque par force brute amélioré
- Attaque par dictionnaire avec des variations

Notre logiciel démontre que même les mots de passe les plus complexes peuvent être vulnérables à des attaques sophistiquées.

D'un autre côté, « Arc-Hash » nous donne la possibilité de craquer un mot de passe en lui fournissant un hachage md5, sha-1, sha-256 ou sha-3, et cela dans un contexte légal.

Grâce à nos recherches approfondies sur le sujet et les résultats obtenus par nos nombreux tests, nous avons pu identifier les faiblesses potentielles. On a constaté que l'adoption de certaines pratiques peut considérablement améliorer la résistance des mots de passe.

Pour un mot de passe plus puissant, nous recommandons les pratiques suivantes :

- Utilisation des mots de passe longs et complexes : Les mots de passe doivent avoir au moins 12 caractères et inclure une combinaison de lettres majuscules, minuscules, chiffres et caractères spéciaux.
- Éviter l'utilisation de mots de passe personnels : Ne pas utiliser le nom, date de naissance, adresse ou d'autres informations personnelles comme mot de passe.
- Ne pas utiliser le même mot de passe pour plusieurs comptes : Si un pirate informatique obtient votre mot de passe pour un compte, il peut l'utiliser pour accéder à vos autres comptes.
- Changement régulier de mots de passe : Il est recommandé de changer les mots de passe tous les 6 mois.
- Utilisation d'un gestionnaire de mots de passe : Ces outils peuvent générer et stocker en toute sécurité des mots de passe complexes.

En conclusion, la sécurité des mots de passe est un pilier fondamental de la cybersécurité. Face aux menaces croissantes, il est crucial de mettre en place des pratiques robustes pour protéger nos données sensibles.

La sécurité informatique est une responsabilité partagée. Chacun de nous a un rôle à jouer pour assurer la sécurité de nos informations personnelles et professionnelles. En continuant à sensibiliser et à éduquer les utilisateurs sur l'importance des mots de passe robustes, nous pouvons tous contribuer à rendre le cyberspace plus sûr.

Bibliographie

- [1] Schneier, Bruce. Secrets and Lies: Digital Security in a Networked World. John Wiley & Sons, 2000.
- [2] Goodrich, Michael T., and Roberto Tamassia. Introduction to Computer Security. Addison-Wesley, 2011.
- [3] Beggs, Justin, and Ajay, Nixon. Mastering Kali Linux for Advanced Penetration Testing. Packt Publishing, 2014.

Webographie

<https://blogs.oracle.com/oracle-france/post/quest-ce-quune-rainbow-table>

<https://medium.com/@jsquared7/password-cracking-what-is-a-rainbow-table-attack-and-how-to-prevent-it-7904000ffcff>

<https://www.kali.org/>

<https://hashcat.net/hashcat/>

<https://fr.wikipedia.org>

<https://github.com/>