**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'état 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

**Table des matières:**

**Table des figures**

# 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. Cependant, malgré leur ubiquité, les mots de passe restent vulnérables à une multitude de techniques d'attaque.

# Problématique :

# Objectif :

# **Chapitre 1: Cadre Théorique et Analyse**

dans ce chapitre1…..

## **Mot de passe : utilite des mots de passe**

Les mots de passe sont des éléments de sécurité cruciaux et des données hautement sensibles. C'est pourquoi la méthode de stockage des mots de passe diffère de celle des données normales. Les mots de passe sont hachés à l'aide de fonctions de hachage (md5,sha-1,sha-256)spécifiques, qui effectuent des traitements sur les mots de passe pour produire une chaîne de caractères unique et ambiguë.

Les fonctions de hachage sont des fonctions non réversibles, ce qui signifie que l'on ne peut pas retrouver les mots de passe à partir du hachage, mais on peut calculer le hachage de chaque mot de passe. Cette propriété garantit 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 donc les mots de passe des utilisateurs restent protégés.

## **2. 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 et mesurer, efficace, universaliser, applicable et déterministe.

1. 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, comme la cryptographie post-quantique, 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 :**

Cependant il y a les collisions qui fait référence à une situation 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 sont conçues pour minimiser les risques de collisions, mais elles ne peuvent pas les éliminer complètement. Les algorithmes plus faibles ou obsolètes comme MD5 et SHA-1 sont particulièrement vulnérables aux collisions, tandis que les 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.

# **Chapitre 2:** **Conception et Implémentation**

Dans ce chapitre, nous examinerons en détail chaque type d'attaque utilisé dans notre logiciel, en mettant en lumière les méthodes et les principes sous-jacents.

## **Force brute**

Une attaque par force brute utilise la méthode de l’essai-erreur pour deviner les identifiants d'un utilisateur. Les pirates informatiques examinent toutes les combinaisons possibles de mots de passe ou de clés jusqu'à trouver la bonne.

* 1. **Méthodologie**

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

**Automatisation :** Les pirates utilisent des scripts ou des logiciels automatisés qui peuvent tenter des 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.

* 1. **Types d’attaques par force brute**

Chaque attaque par force brute peut utiliser différentes méthodes pour découvrir vos données confidentielles. Vous pourriez être exposé à l’une des méthodes par force brute suivantes :

* + 1. **Attaques par force brute simples**

L'attaque par force brute simple est une méthode utilisée en cryptanalyse pour trouver un mot de passe ou une clé. Elle consiste à tester, une à une, toutes les combinaisons possibles. Dans le cadre de cette méthode, nous avons développé un code qui permet à l'utilisateur de choisir la méthode de hachage qu'il souhaite utiliser (MD5, SHA1, SHA256, SHA512).

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 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")

**Avantage :**

1. **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.

1. **Simplicité :**

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

1. **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.

**Inconvénients :**

1. **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 a 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 :

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

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

|  |  |  |
| --- | --- | --- |
| Taille de mot de passe | combinaisons possibles | Les valeurs |
| 6 | 776 | 2.084 1011 |
| 7 | 777 | 1.604 1013 |
| 8 | 778 | 1.235 1015 |
| 9 | 779 | 9.515 1016 |
| 10 | 7710 | 7.326 1018 |
| 11 | 7711 | 5.641 1020 |
| 12 | 7712 | 4.3431022 |

**- Estimation du Temps maximum Requis pour Générer 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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 générer 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.

1. **Nécessité d'un PC Robuste :**

La méthode de génération exhaustive des combinaisons de mots de passe nécessite un ordinateur puissant pour effectuer efficacement les calculs, 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.

* + 1. **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.

**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 compile une liste de mots de passe potentiellement utilisés par les utilisateurs, 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.

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.

1. **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 :

- 'a': ['@', '4', 'A'] : Le caractère 'a' peut être remplacé par '@', '4', ou 'A'.

- 'e': ['3', 'E'] : Le caractère 'e' peut être remplacé par '3' ou 'E'.

- 'i': ['1', 'I'] : Le caractère 'i' peut être remplacé par '1' ou 'I'.

- 'o': ['0', 'O'] : Le caractère 'o' peut être remplacé par '0' ou 'O'.

- 's': ['$', '5', 'S'] : Le caractère 's' peut être remplacé par '$', '5', ou 'S'.

- 't': ['7', 'T'] : Le caractère 't' peut être remplacé par '7' ou 'T'.

Par exemple, le mot "password" pourrait être modifié en "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.

1. **Avantages :**

**- Efficacité :** Elle peut être rapide et efficace pour deviner les mots de passe faibles ou couramment utilisés.

**-Accessibilité :** Les dictionnaires de mots de passe sont facilement disponibles en ligne, ce qui facilite la mise en œuvre de cette attaque.

1. **Inconvénients :**

**- Limitations des dictionnaires :** Les mots de passe complexes ou uniques ne seront pas présents dans les dictionnaires, réduisant ainsi l'efficacité de l'attaque.

**- Temps nécessaire :** Pour des mots de passe complexes, l'attaque par dictionnaire peut prendre beaucoup de temps, surtout si des variations sont nécessaires.

1. **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.

* + 1. **Attaque qui combine Brute force et dictionnaire**

**1)-Principe :**

**Phase 1 : Préparation**

* **Génération de fichier de combinaisons :** Un fichier texte exhaustif est créé, contenant toutes les combinaisons possibles de caractères, incluant 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)-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

**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 s'avérer 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 précalculés 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.

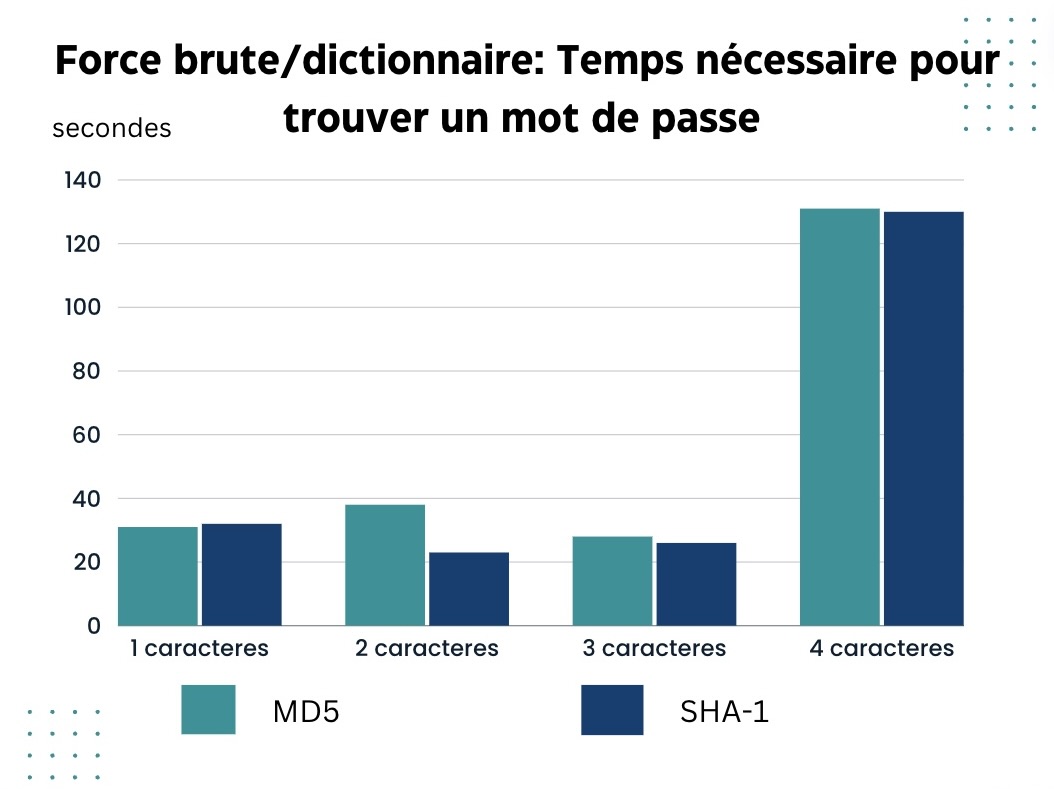
**Remarque :** Dans le cadre présent, nous avons opté pour la création d'un fichier composé de mots allant de 1 à 4 caractères.

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

**4)- Complexité :**

1. **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, de longueur variant de 1 à 4 caractères. Voici les résultats :



L’algorithme de l’attaque force brute/dictionnaire a une complexité temporelle de O (

**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.

**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), des chiffres et des 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. Arc en ciel**

**1.Definition d’une table arc en ciel**

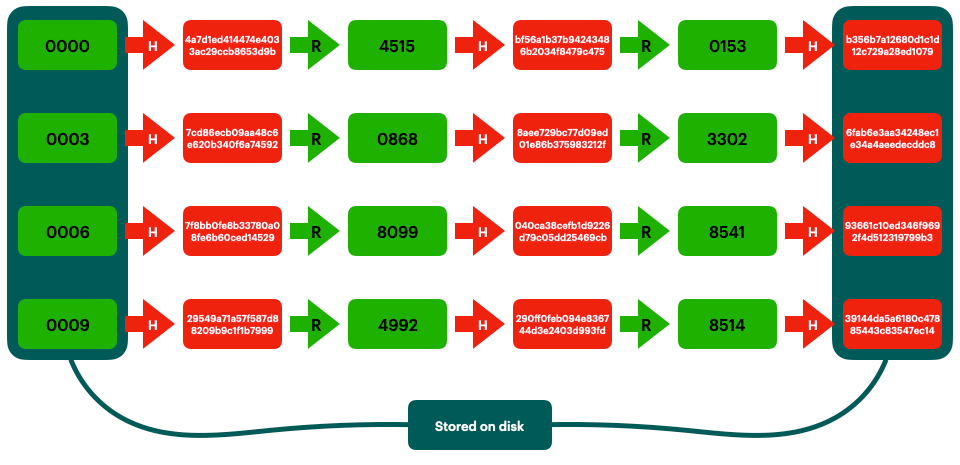
Une Rainbow Table est un fichier contenant une multitude de mots de passe reliés à leur valeur de hachage. Les cybercriminels s’en servent pour cracker des mots de passe.

Les Rainbow Tables 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 généré automatiquement dans certaines distributions de linux, ou même en utilisant des outils comme Rainbow Crack.

À noter que les Rainbow Table peuvent également être utilisées par des experts en cybersécurité pour identifier des failles ou effectuer des tests de sécurité.



**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.

**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)

**3.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 nombres 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.

**4.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.

**5.Complexité**

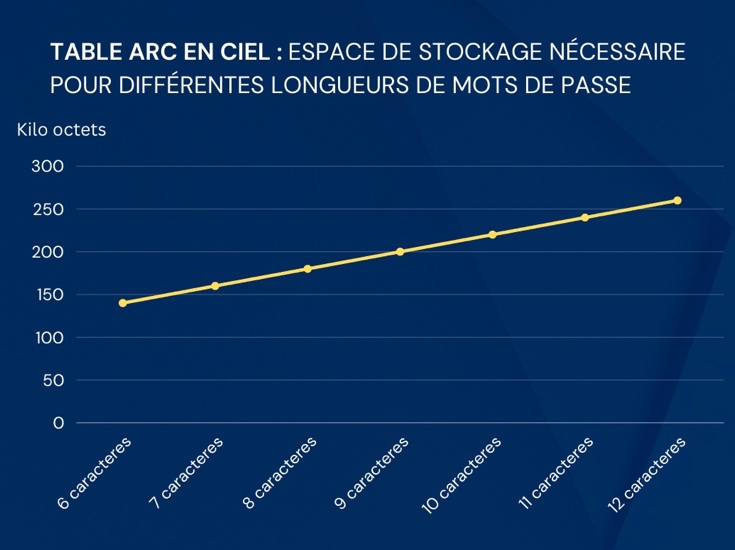
L’attaque par table arc-en-ciel est basée sur un compromis entre l’espace et le temps, ce qui signifie qu’elle utilise moins de temps de traitement mais plus d’espace de stockage par rapport à une attaque par force brute.

**5.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 augmentent chaque fois qu’on accroit 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 :



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.

**5.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() , le temps nécessaire augmente donc de manière quadratique.

Par conséquent, nous pouvons conclure que cette attaque produis 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.

**6.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 précalculées.
* **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.

# **Chapitre 3 :** **Test et Exploitation**

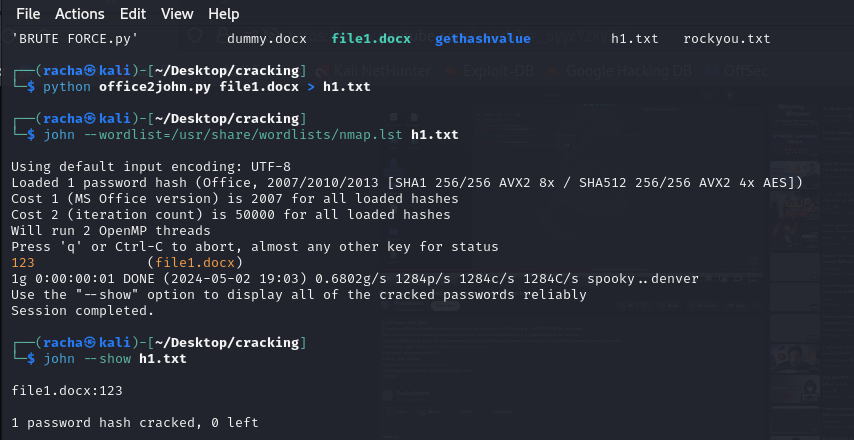
Les méthodes d'attaque informatique constituent un domaine crucial de la sécurité des systèmes d'information. Dans cet chapitre. nous explorerons les différentes méthodes d'attaques et leur mise en œuvre dans des environnements populaires tels que Linux et Microsoft Office.

**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.

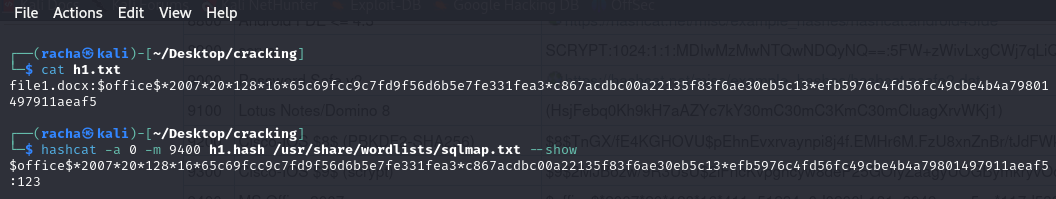
**En utilisant l’outil "John the Ripper" :**

L’outil John the Ripper est utilise pour le craquage de mots de passe qui teste la sécurité des mots de passe en essayant de les deviner. Il prend en charge diverses méthodes d'attaque, telles que la force brute et les attaques par dictionnaire. Quant à "john", c'est la commande utilisée pour exécuter John the Ripper dans un terminal. Maintenant, en ce qui concerne la séquence d'actions que vous avez fournie : Tout d'abord, le script "office2john.py" est utilisé pour extraire le hachage du mot de passe du fichier Word "file1.docx" et le stocker dans un fichier nommé "h1.txt". Ensuite, la commande "john --wordlist=/usr/share/wordlists/nmap.lst h1.txt" est exécutée, ce qui indique à John the Ripper de tenter de craquer le mot de passe à partir du fichier "h1.txt". Dans ce cas particulier, John the Ripper a réussi à trouver le mot de passe, qui est "123", comme indiqué dans la sortie de la commande "john --show h1.txt ". Cela signifie qu'un seul hachage de mot de passe a été craqué avec succès à partir du fichier "h1.txt", et aucun hachage supplémentaire n'est resté à craquer., qui est "123". Le résultat de la tentative de craquage est affiché, montrant que le mot de passe a été trouvé avec succès.

****

**En utilisant l’outil "hashcat" :**

l'outil Hashcat pour tenter de retrouver le mot de passe correspondant à ce hachage. Hashcat est un utilitaire de récupération de mots de passe spécialisé qui peut exécuter différents types d'attaques, y compris les attaques par force brute et par dictionnaire. Dans ce cas, on a spécifie une attaque par force brute avec l'option "-a 0" et fournit le mode de hachage spécifique utilisé pour les fichiers Office avec l'option "-m 9400". En plus de cela, un fichier de dictionnaire contenant une liste de mots courants sqlmap.txt est fourni à Hashcat pour qu'il puisse essayer de deviner le mot de passe. Lorsque Hashcat termine son exécution, il parvient à trouver un mot de passe correspondant au hachage fourni. Dans cet exemple, le mot de passe découvert est "123", qui est alors affiché pour l'utilisateur avec le hachage correspondant. Cette séquence démontre l'utilisation efficace de Hashcat pour craquer les mots de passe des fichiers Office en utilisant des attaques spécialisées et des dictionnaires de mots.

****

**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 sont 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

Sur Ubuntu version 10.10, on observe l'utilisation du type de hachage "$6$", associé à SHA-512. 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 avec le 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.

# 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 de hachage arc-en-ciel
* Attaque par force brute amélioré
* Attaque par dictionnaire amélioré

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

D ‘un autre coté, « 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 légale.

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 resistance 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 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.

En fin de compte, 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 cyberespace plus sûr.

# Bibliographie

[1 ]Sécurité informatiqueEthical Hacking : Apprendre l'attaque pour mieux se défendre (6e édition)

# Webographie