Défi n°2 : force brute d'un mot de passe haché en MD5

Le deuxième défi est une attaque en force brute sur un mot de passe haché en MD5.

1 Outils et documents

Documents à utiliser :

- document 3 : Chiffrement symétrique et asymétrique ;

- document 4 : Fonctions de hachage d'après la CNIL.

Le programme hashcat :

Hashcat est un outil qui permet de retrouver des mots de passe à partir de plusieurs exemples de hachés.

1. Hashcat prend comme entrée un ensemble de mots à partir d'un dictionnaire ou par combinaisons ;
2. Hashcat calcule ensuite le haché pour chacun de ces mots puis compare le résultat avec le ou les hachés contenus dans un autre fichier qui stocke des hachés ciblés ;
3. Les coïncidences sont des mots de passe récupérés.

Cas d'usage :

- disponibilité de la base SAM[4] sous Windows ou du fichier */etc/shadow[5] sous Linux ;*

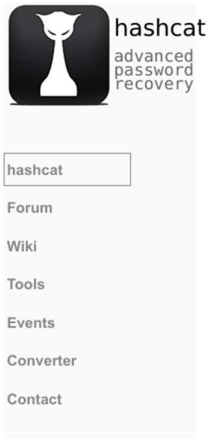
- récupération de hachés dans le code d'un programme ou dans une base de données.

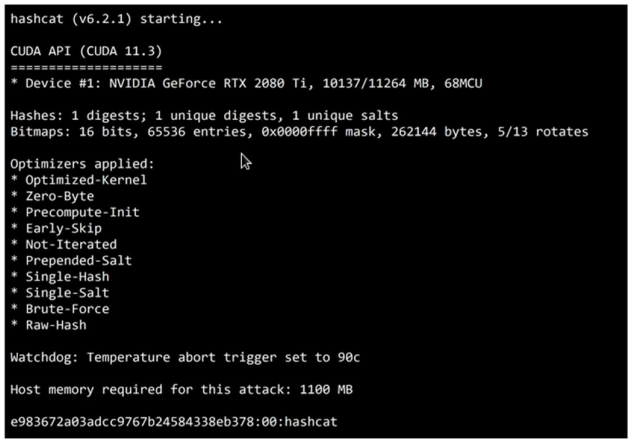
Installation :

```
# sudo apt install hascat
```

Le lien suivant donne le mode d'emploi de l'outil: *https://www.kali-linux.fr/hacking/comment-cracker-des-mots-de-passe-avec-hashcat*

Autres outils en ligne :



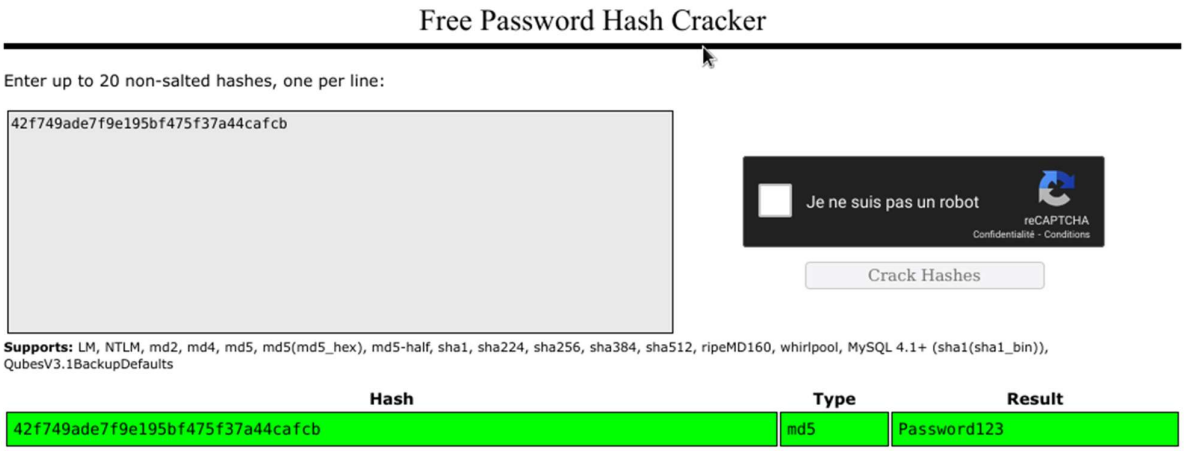


https://crackstation.net/

[4]*Le gestionnaire de compte de sécurité (SAM) est une base de données présente sur les ordinateurs exécutant des systèmes d'exploitation Windows qui stocke les comptes d'utilisateur et les descripteurs de sécurité pour les utilisateurs sur l'ordinateur local.*

[5]*/etc/shadow est un fichier texte qui contient les informations sur les mots de passe des utilisateurs du système.*

https://hashes.com/en/decrypt/hash



Le système détermine que le hachage MD5 de la chaîne Password123 a produit le haché fourni.

A vous de jouer

Travail à faire 2

Q1. Rappeler la différence entre chiffrement symétrique et chiffrement asymétrique et proposer un schéma pour chacune de ces deux méthodes.

Q2. Recenser et décrire les principaux algorithmes de chiffrement symétrique et asymétrique.

Q3. Qu'appelle t-on une fonction de hachage, quel peut-être son rôle lors du développement d'une application. Lister les principales fonctions de hachage et leurs principales caractéristiques.

Q4. Expliquer ce que fait le code java suivant et quel est le problème de sécurité posé par ce code ?

```
@Test
public void givenFile_generatingChecksum_thenVerifying()
  throws NoSuchAlgorithmException, IOException {
    String filename = "src/test/resources/test_md5.txt";
    String checksum = "5EB63BBBE01EEED093CB22BB8F5ACDC3";

    MessageDigest md = MessageDigest.getInstance("MD5");
    md.update(Files.readAllBytes(Paths.get(filename)));
    byte[] digest = md.digest();
    String myChecksum = DatatypeConverter
      .printHexBinary(digest).toUpperCase();

    assertThat(myChecksum.equals(checksum)).isTrue();
}
```

Q5. Dans le code Symfony suivant, expliquer le rôle de 'auto'. Conclure sur l'intérêt d'utiliser un framework lors du développement d'une application.

Source du code : https://symfony.com/doc/current/security/passwords.html

 # config/packages/security.yaml

```
security:
    # ...

    password_hashers:
        # auto hasher with default options for the User class (and children)
        App\Entity\User: 'auto'

        # auto hasher with custom options for all PasswordAuthenticatedUserInterface instances
        Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface:
            algorithm: 'auto'
            cost:      15
```

Q6. Un attaquant récupère le haché suivant dans une base de données.

4138dd8a9c4ac256fcf4a42b633f9f88

Trouver le mot de passe correspondant avec hashcat ou avec un autre outil en ligne.

IX Bonnes pratiques et contre-mesures

Bonnes pratiques :

D'après le groupe OWASP :

Déterminer d'abord quelles données doivent bénéficier d'une protection chiffrée (mots de passe, données patients, numéros de cartes, données personnelles, etc.), lors de leur transfert ou leur stockage. Pour chacune de ces données :

- Les données circulent-elles en clair ? Ceci concerne les protocoles tels que HTTP, SMTP et FTP. Le trafic externe sur internet est particulièrement dangereux.
- Des algorithmes faibles ou désuets sont-ils utilisés, soit par défaut, soit dans le code source existant ?
- Est-ce que des clés de chiffrement par défaut sont utilisées ? Des clés de chiffrement faibles sont-elles générées ou réutilisées ? Leur gestion et rotation sont-elles prises en charge ?
- Les réponses transmises au navigateur incluent-elles les directives et en-têtes de sécurité adéquats ?
- Est-ce que le certificat serveur reçu est valide ?
- Est-ce que des fonctions de hachage dépréciées telles que MD5 ou SHA1 sont utilisées ou est-ce que des fonctions de hachage non cryptographiques sont utilisées là où des fonctions de hachage cryptographiques sont nécessaires ?
- Est-ce que des méthodes cryptographiques de remplissage dépréciées, comme PKCS 1 v1.5 sont utilisées ?

Contre-mesures : d'après OWASP.

On veillera au minimum à suivre les recommandations suivantes :

- Classifier les données traitées, stockées ou transmises par l'application. Identifier quelles données sont sensibles selon les lois concernant la protection de la vie privée, les exigences réglementaires ou les besoins métier.
- Ne pas stocker de données sensibles sans que cela ne soit nécessaire. Les rejeter ou utiliser une tokenisation[6] conforme à la norme de sécurité de l'industrie des cartes de paiement (PCI DSS). Les données que l'on ne possède pas ne peuvent être volées.

[6]Procédé permettant de remplacer une donnée critique par un élément équivalent qui n'aura aucune valeur intrinsèque ou signification exploitable une fois sortie du système. Lorsqu'un jeton de session est utilisé, l'utilisateur bénéficie, pendant toute la durée de vie du jeton, d'un accès à l'application ou au site web pour lequel le jeton lui est accordé.

- S'assurer de chiffrer toutes les données sensibles au repos.
- Choisir des algorithmes éprouvés et générer des clés robustes. S'assurer qu'une gestion des clés est en place.
- Chiffrer toutes les données transmises avec des protocoles sécurisés tels que TLS. Forcer le chiffrement en utilisant des directives comme HTTP Strict Transport Security (HSTS).
- Désactiver le cache pour les réponses contenant des données sensibles.
- Appliquer les contrôles de sécurité requis selon la classification de la donnée.
- Ne pas utiliser des protocoles obsolètes tels que FTP et SMTP pour échanger des données sensibles.
- Stocker les mots de passe en utilisant des fonctions de hachage avec salage et facteur de délai, telles que Argon2, scrypt, bcrypt ou PBKDF2.
- Les vecteurs d'initialisation doivent être choisis de façon appropriée au mode d'opération. Pour la plupart des modes, cela signifie utiliser un générateur de nombres pseudo-aléatoires cryptographiquement sécurisé (CSPRNG[7]). Dans tous les cas, un vecteur d'initialisation ne devrait pas être utilisé deux fois pour une clé fixe.
- S'assurer qu'une génération cryptographiquement aléatoire est utilisée là où c'est approprié, et qu'elle n'a pas une graine aléatoire prévisible ou avec une faible entropie.

[7]CSPRNG : Un générateur de nombres pseudo-aléatoires cryptographiquement sécurisé ou un générateur de nombres pseudo-aléatoires cryptographiques est un générateur de nombres pseudo-aléatoires avec des propriétés qui le rendent adapté à une utilisation en cryptographie.