**TD Pentest 3 – Force Brute**

Le but de ce TD est d’approfondir les techniques d’attaques par force brute. Cette technique consiste à essayer toutes les combinaisons possibles des caractères disponibles, jusqu’à trouver la bonne ! Ça peut être envisageable sur un cadenas à 3 voire 4 chiffres, mais ça peut être très long pour trouver un mot de passe …

Nous allons utiliser différents outils qui existent et étudier les différentes optimisations possibles pour accélérer les recherches

## **Génération de dictionnaires**

# Crunch

Une première optimisation possible consiste à limiter les essais à une liste de mots limitée. On parlera alors d’attaque par dictionnaire au lieu de force brute.

Sachant qu’il existe des profils connus d’utilisateurs (utilisation de prénoms de leur fils, de leur fille, de leur chien, de leur chat … avec souvent des majuscules au début et des chiffres à la fin quand ils y sont obligés), cela peut réduit considérablement le nombre de combinaisons possibles.

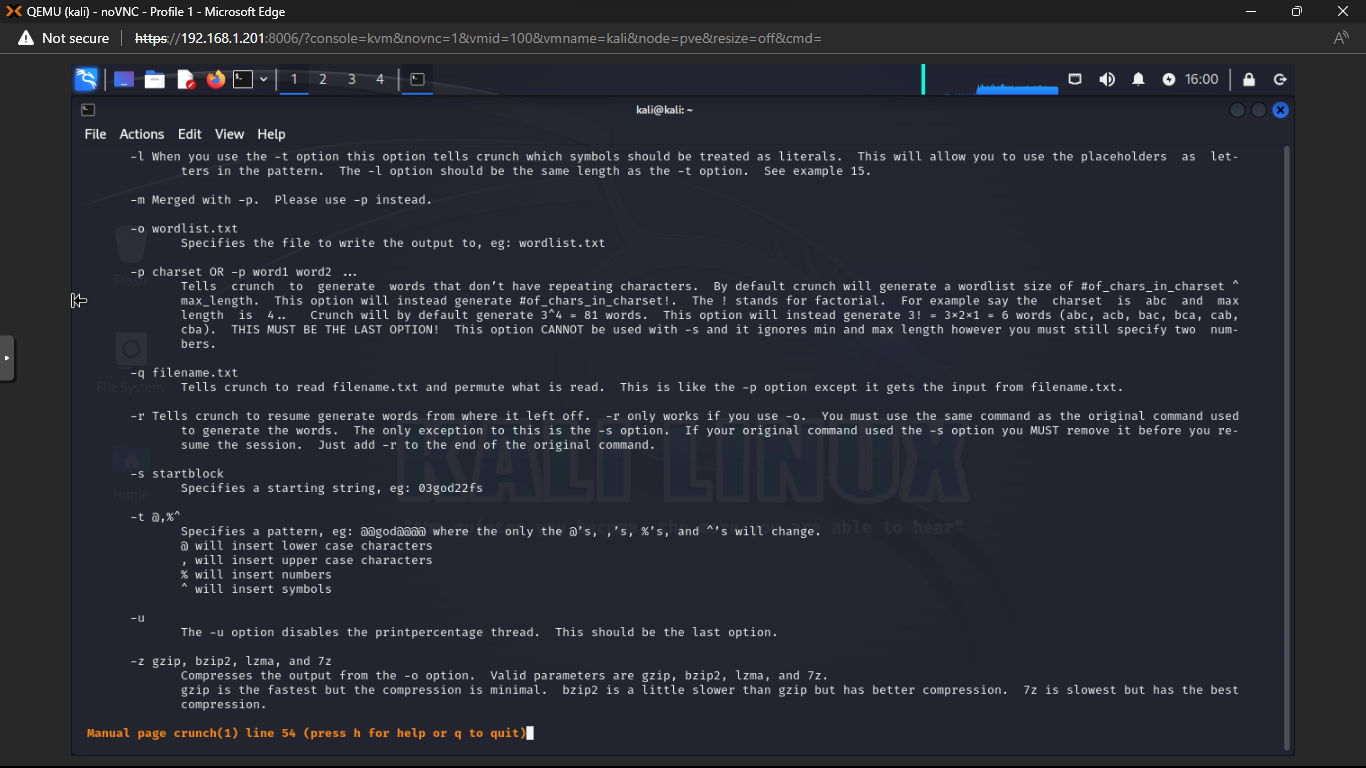
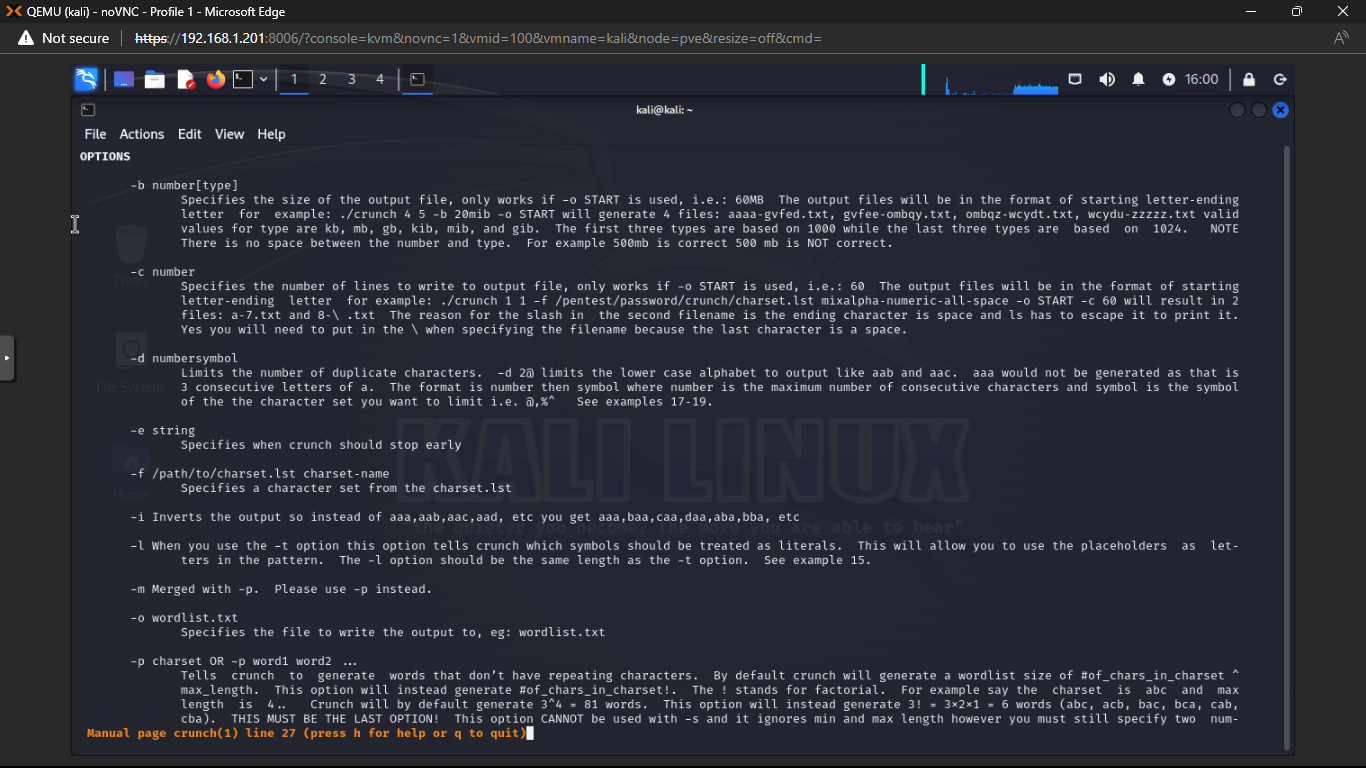
Il est aussi très facile d’avoir des indications sur le mot de passe de quelqu’un qu’on voit, ou entend taper…( les premiers et/ou derniers caractères, le nb de caractères, un caractère en particulier, …)

Le logiciel **crunch** permet de créer des listes de mots, qui peuvent être utilisées comme dictionnaires dans une attaque de ce type.

1. Affichez la page du manuel :

man crunch

Quels sont les options de **crunch** ?



Sous quelle forme doit être donnée la liste des caractères pour la génération des mots ?

crunch <minimum\_size> <maximum\_size> <w\_characters> -o <output\_file>

1. Testez la commande :

crunch 2 3 -o toto.lst

Que contient le fichier **toto.lst** ? Combien de lignes ? (utilisez **wc**)

Tous les mots de 2 & 3 caractères en lettres en minuscules. 18252 (wc -l)

1. Testez la commande :

crunch 2 3 abcABC123 -o toto.lst

Que contient le fichier **toto.lst** ?

Toutes les combinaisons de 2 & 3 caractères avec a, b, c, A, B, C, 1, 2 & 3

Combien de lignes ? (utilisez wc) 810

Peut-on prévoir ce nombre de lignes ? oui par le calcul

Par quelle formule ?

On a 9 caractères 🡪 d’une longueur de 2 éléments 🡪 9^2

On a 9 caractères 🡪 d’une longueur de 3 éléments 🡪 9^3

9^2 + 9^3 = 810

Testez la commande :

crunch 2 3 -o toto.lst -p abc ABC 123

Que contient le fichier **toto.lst** ?

Toutes les concaténations des 3 paramètres

Combien de lignes ? (utilisez **wc**)

6

1. Testez la commande :

crunch 4 4 -t ,@%^ -o toto.lst

Que contient le fichier **toto.lst** ? Combien de lignes ?

223080

Que signifie l’option **-t** ?

Donne un paterne à respecter (lettres, puis chiffres…)

1. On veut générer des mots de 4 caractères avec 2 lettres minuscules suivi de 2 chiffres. Quelle commande **crunch** faut-il faire ?

crunch 4 4 -t @@%% -o toto.lst

1. On veut générer des mots de 4 caractères avec A puis une lettre minuscule suivi de un chiffre et d’un point d’exclamation.

Quelle commande **crunch** faut-il faire ?

crunch 4 4 -t A@%! -o toto.lst

1. Testez la commande :

crunch 2 2 abc + 123 -o toto.lst -t @%

Que contient le fichier **toto.lst** ? Que signifie le **+** ?

On spécifie ce qu’on définit dans -t (quelles lettres, quels chiffres)

1. Testez la commande :

crunch 2 2 abc + + \!$ -o toto.lst -t @^

Que contient le fichier **toto.lst** ?

Mots de ­2 de longueur avec une lettre (a, b ou c) au début puis \, ! ou $ à la fin

# Cewl

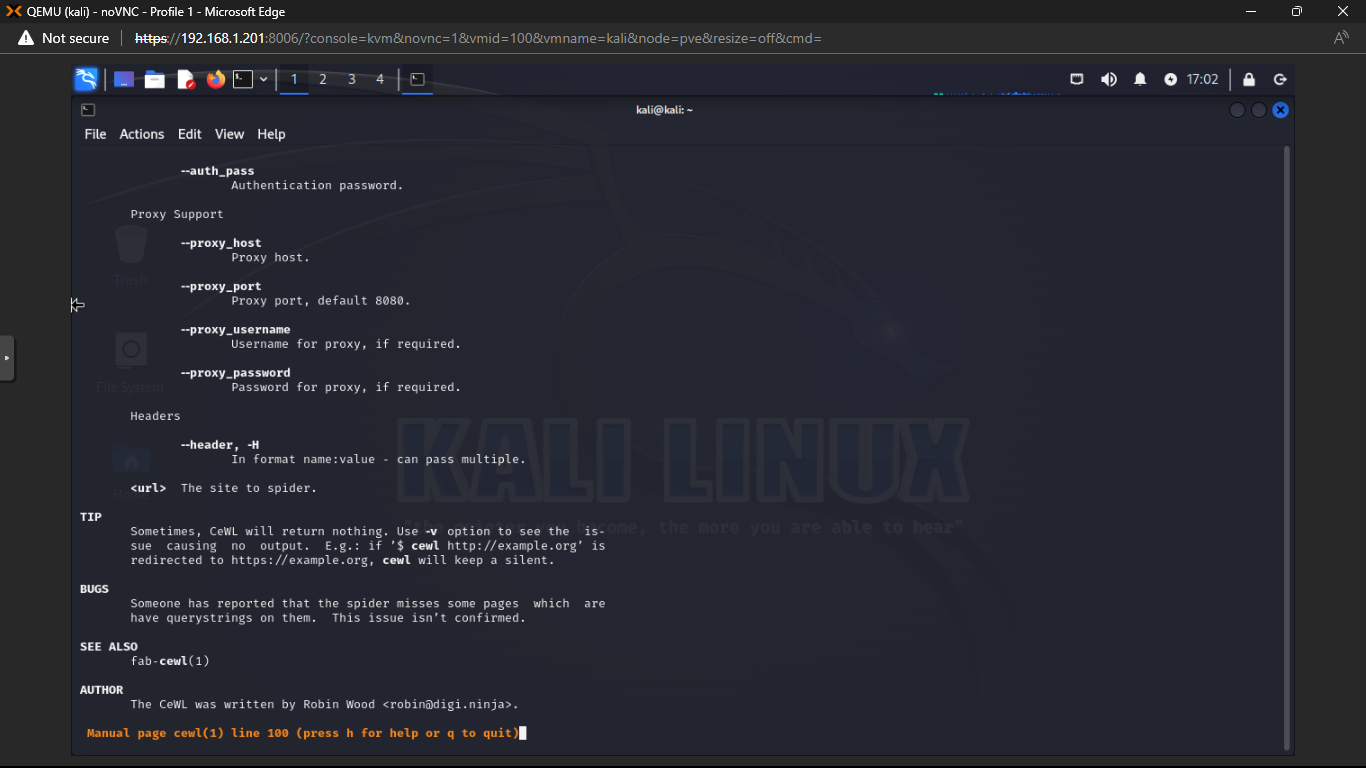
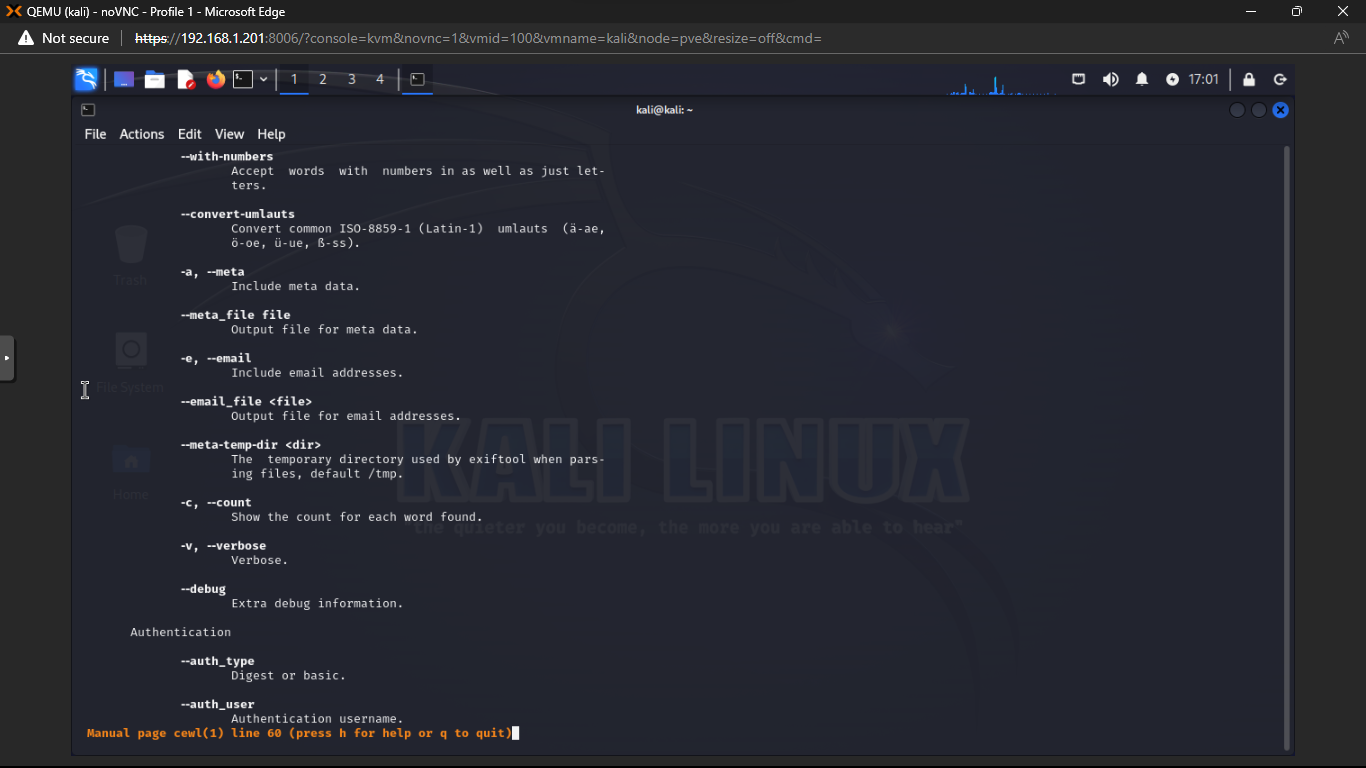
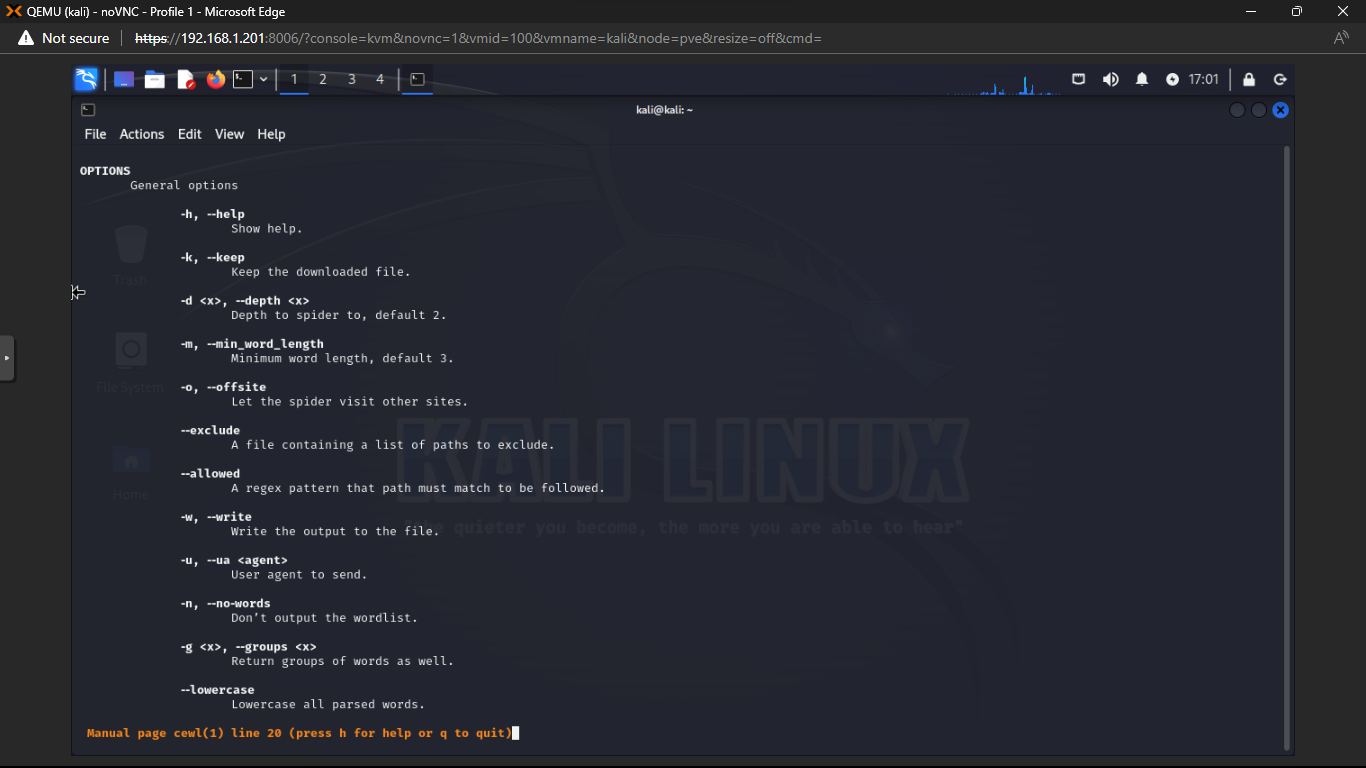
Il se peut que le choix d’un mot de passe soit influencé par un contexte professionnel, passionnel, ou autre … Cela peut parfois être intéressant de tester !

Le logiciel **cewl** permet de créer un fichier de mots à partir d’un site web.

1. Affichez la page du manuel de cewl

man cewl

Quelles sont les options de la commande ?



1. Testez la commande :

cewl -w titi.lst -d 2 https://www.univ-pau.fr

Expliquez les options.

-d deph to spider to, default 2 (profondeur de recherche)

-w écrire sur un fichier de sortie

Combien de lignes contient **titi.lst** ?

2249

1. Quelle commande faut-il faire pour récupérer les adresses email du site www.montdemarsan.fr dans le fichier mail.txt ? (faites une profondeur de 2)

cewl -e <http://www.montdemarsan.fr> -w titi.lst

A quoi cette liste de mails peut-elle bien être utile ?

phishing

# Hydra

Le logiciel **hydra** permet de faire, entre autres, une attaque par dictionnaire.

1. Générez un dictionnaire avec des mots commençant par une majuscule, puis un @ suivi d’une minuscule et de 2 chiffres.

crunch 5 5 + + + @ -t ,^@%% -o tata.lst

1. Ouvrez **hydra-graphical** dans le menu **05-Password Attacks/Online Attacks**.
2. Configurez la cible, le port et le protocole dans le menu **Target**.
3. Configurez l’identifiant **bad** et la liste de mots de passe dans le menu **Password**.
4. Dans le Menu **Tuning**, baissez le **timeout**, qui correspond au délai maximum accordé au processus, à partir duquel on considère ici la connexion comme échouée.

➔ Si le **timeout** est trop petit (au moins **1**), le mot de passe ne sera pas trouvé.

➔ Si le **timeout** est grand, le temps pour trouver le mot de passe sera plus long.

1. Dans le menu **Start**, lancez l’attaque

Quel est le **timeout** optimal ? (faites plusieurs essais)

Quel mot de passe avez-vous trouvé ?

A combien de mots de passe par seconde (environ) pouvez-vous estimer la vitesse de recherche optimale de cet outil ?

Cette technique est extrêmement longue et ne peut être utilisée que dans le cadre d’un mot de passe pressenti, pour lequel le nombre de tentatives est extrêmement limité …

Il offre néanmoins un avantage majeur : il n’est pas utile de disposer du fichier de mot de passe du système cible pour pouvoir tenter l’expérience !

# John the ripper

**John the ripper** est un logiciel qui permet de perpétrer le même type d’attaques que hydra. En revanche, il a besoin d’un fichier comportant des identifiants et des empreintes de mot de passe. Ce fichier peut être créé à partir du fichier d’identifiants ***/etc/passwd*** et du fichier des empreintes de mot de passe ***/etc/shadow*** du système Linux. Chacun de ces deux fichiers contient une information.

1. Récupérez le fichier ***/etc/passwd*** par **scp** et le compte trouvé précédemment.

Pour des raisons de sécurité qui ne vont pas tarder à vous paraître évidentes, le fichier /etc/shadow est seulement accessible au root sur un équipement Unix.

Il faudra donc utiliser un stratagème pour récupérer ce fichier dans un contexte réel … la mise en place d’une attaque XSS peut par exemple faire ce travail …

Ici nous allons tricher et supposer que le travail est déjà fait. Ce fichier est disponible via ftpd sur la machine cible.

1. Récupérez le par la commande :

tftp <@ IP de la machine cible>

get shadow

scp … :… …/.

1. Créez le fichier formaté pour **john** :

unshadow passwd shadow > unshadow.txt

1. Lancez l’attaque :

john –format=crypt unshadow.txt

Qu’avez-vous trouvé comme mots de passe ?

Bonzo

Que pensez-vous de cette méthode par rapport à la méthode précédente ?

Rapide + discret

La commande précédente a utilisé le fichier de mots de passe par défaut de john. La méthode suivante utilisera une méthode avec des règles.

1. Voir <https://openwall.com/john/doc/RULES.shtml>pour la syntaxe.
2. Effacez le répertoire **.john** qui a gardé en mémoire les mots de passe trouvés :

rm -rf .john

1. Créez le fichier de configuration ***john.conf*** :

[List.Rules:Example]

cAz"[0-9]"

1. Exécutez la commande :

john --config=./john.conf --wordlist=/usr/share/john/password.lst -- rules=Example --stdout

Quel est l’effet de la règle ?

1. Testez la règle **sa@Az’’[0-9][0-9]’’** à la place de l’autre

Quel est l’effet de cette règle ?

1. Avec **crunch**, créez une liste de mots de 3 caractères dans ***pwd3.lst***.

Comment modifier la règle précédente pour créer une liste contenant le mot de passe de bad ?

1. Avec la liste précédente, appliquez la bonne règle pour trouver le mot de passe de **bad**.

john –config=./john.conf –-wordlist=pwd3.lst –-rules=Example –-format=crypt unshadow.txt

Quelle règle avez-vous utilisée ?

Pouvez-vous donner une estimation des performances de john-the-ripper en nombre de tests par seconde ?

1. **Hashcat**

Vous avez bien compris que la puissance de calcul est le nerf de la guerre en ce qui concerne les approches par force brute ou par dictionnaire. Plus on peut faire d’essai à la seconde, plus vite on se rapproche de la solution. Le logiciel **hashcat** a été conçu pour tirer profit du processeur de la carte graphique de votre pc. Mais pour cela … il faut disposer d’une bonne carte graphique ! Comme ce n’est pas le lot des ordinateurs de l’IUT, nous utiliserons ici le processeur classique.

1. Testez la commande :

hashcat -I

Quel type de processeur est utilisé ?

1. Prenez la ligne de l’utilisateur *george* de **shadow** et gardez l’empreinte du mot de passe :

grep george shadow | cut -d “:” -f 2 > hash.txt

1. Éditez le fichier *hash.txt* et supprimez **$y$** en début de ligne.
2. La méthode utilisée pour calculer l’empreinte est SHA512, on peut le vérifier sur la machine GNU/Linux par la commande :

grep ENCRYOPT\_METHOD /etc/login.defs

1. Affichez les méthodes d’empreintes sha512 utilisées par **hashcat** :

hashcat -h | grep sha512

Quel code correspond à SHA512 (Unix) ?

La chaîne de caractères qui correspond à SHA512 Unix doit commencer par **$6$.**

Est-ce le cas ?

La chaîne de caractère de l’empreinte du mot de passe commence par **$y$**, ce qui correspond à un chiffrement **yescrypt**, lequel n’est pas encore supporté par **hashcat**. Nous allons donc encore être obligés de tricher !

Pour pouvoir tester **hashcat,** on va créer un mot de passe pour **apache2**.

1. Créez le mot de passe **tula** dans le fichier ***hsh.txt*** avec la commande :

htpasswd -c hsh.txt george

1. Extrayez l’empreinte du mot de passe de *george* :

grep george hsh.txt | cut -d ":" -f 2 > hsh-pass.txt

1. Cherchez dans **hashcat** les mots de passes qui correspondent à **HTTP :**

hashcat -h | grep HTTP

Quel code correspond au mot de passe dans hsh-pass.txt (voir 1er caractères) ?

1. Testez la commande :

hashcat -m 1600 -a 0 hsh-pass.txt /usr/share/john/password.lst

À quoi correspond l’option **-a 0** ?

Notez le temps mis pour trouver le mot de passe.

1. Effacez l’historique de **hashcat** :

rm .local/share/hashcat/hashcat.profile

1. Lancez l’attaque :

hashcat -m 1600 -a hsh-pass.txt

Notez le temps mis pour trouver le mot de passe, comparez-le à la valeur précédente.

1. Refaites la manipulation précédente avec le mot de passe **newcourt** pour l’utilisateur **badboy**.
2. Utilisez la commande suivante pour lancer l’attaque :

hashcat -m 1600 -a hsh-pass.txt ?l?l?l?l?l?l?l?l

Que signifient les « **?l »** ?

Que constatez-vous au niveau des temps ? (vous pouvez appuyer sur **s** pour voir l’avancement de l’attaque).

Quel est l’estimation du temps restant ? Qu’en pensez-vous ?

1. Essayez avec un mot de passe de 6 lettres : **azerty** (n’oubliez pas de modifier le **mask**)
2. Utilisation d’une carte graphique : si vous disposez d’une carte graphique **NVidia** par exemple, vous pouvez télécharger la version binaire sur <https://hashcat.net/>.

Sous Windows, vous pouvez vérifier la prise en compte de la carte graphique par :

hashcat.exe -I