TP cyber sécurité 3

Exercice 1: John

John The Ripper (JTR) est un logiciel libre de cassage de mot de passe. Il inclut l'auto-détection des tables de hachage utilisées par les mots de passe, l'implémentation d'un grand nombre d'algorithme de cassage, et autorise facilement l'implémentation de nouveaux algorithmes.

John utilise 3 modes d'actions successivement : mode simple, attaque par dictionnaire, mode incrémental.

Mode simple : des tests sont effectués à partir du nom de l'utilisateur auquel on fait subir des variantes

Attaque par dictionnaire : des tests sont effectués à partir d'un fichier dictionnaire auquel on fait subir des variantes

Mode incrémental: toutes les combinaisons sont testées -> très long...

Sur la page web https://perso.isima.fr/~palauren, dans la partie Aide pour les TP de cybersécurité se trouve 3 fichiers intéressants : passwd.txt, dico_animaux.txt et jtr_cheat.

Le premier contient 10 mots de passe, chiffré en MD5, l'autre un dictionnaire des noms d'animaux et le dernier une petite aide sur JTR. Autre aide possible, le site de john the ripper *http://www.openwall.com/john/doc/RULES.shtml*

1. Pour lancer john en mode single, il suffit de taper :
   ***john --single fichier_passwd* --format="raw-md5"**

John va prendre le fichier passwd et tester différents algorithmes. Ces algorithmes se trouvent dans le fichier **john.conf** dans la partie List.Rules : single (fichier par défaut)
Il est nécessaire de spécifier le codage utilisé, ici : **--format="raw-md5"**

2. Pour lancer en mode dictionnaire, il suffit de taper :
   john --wordlist=nom_dico fichier_passwd ***--rules***

   Les algorithmes testés seront dans la section : List.Rules: wordlist du fichier john.conf.

   Surtout ne pas oublier –rules, car cela permet de tester toutes les règles

   Pour voir les mots de passe trouvés : john --show fichier_passwd

Votre objectif : trouvez les mots de passe de ces 12 personnes.

Pour cela, il faudra modifier le fichier john.conf.

1) Créer un répertoire test.
2) Chercher le fichier john.conf dans l'arborescence et copier le dans votre répertoire test.
3) Ajouter des règles dans ce fichier john.conf pour découvrir les autres mots de passe
   Il faut ajouter dans la ligne de commande : --config= john.conf

   Ex : john –config=john.conf –single password.txt –format="raw-md5"

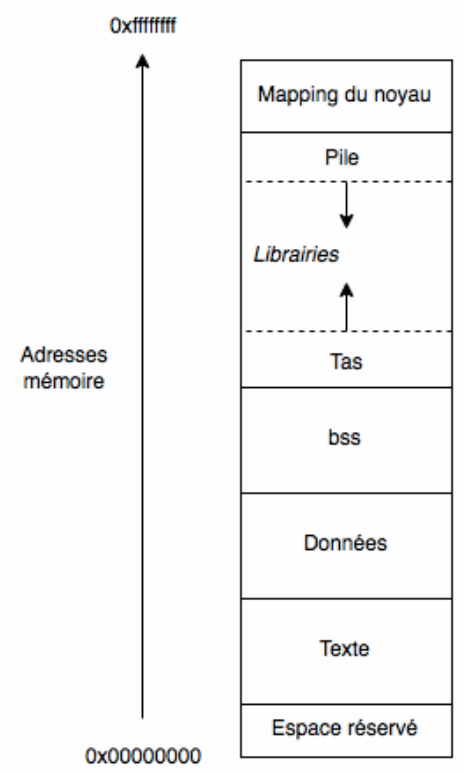
Pour décoder les mots de passe, un peu d'aide quand même. Trois d'entre eux utilisent des règles simples basées sur leur nom, avec optionnellement des nombres rajoutés (maximum 4). Un a mis une lettre devant son nom, avant de rajouter à la fin son année de naissance. Un autre à inverser la casse de sa troisième et quatrième lettre avant de rajouter un nombre inférieur à 9999 à la fin. Le dernier à fait une rotation d'une lettre avant de doubler le mot.

Les autres aiment les animaux et ont dérivés leurs mots de passe de ceux-ci. L'un est classique, d'autres ont rajoutés des nombres devant ou derrière (max 3), et mis la première lettre ou pas en majuscule. Le suivant a pris le nom d'un animal, l'a mis à l'envers, et l'a dédoublé. Un autre est plus complexe : il a changé toutes les voyelles par un nombre et mis le tout en majuscule. Le dernier a seulement concaténé 2 animaux...

Exercice 2: Introduction au Buffer overflow

Le but de cet exercice consiste à montrer le problème du buffer overflow qui comme il est bien connu, commence à être assez facilement contré par les systèmes d'exploitation.

1. Créez un petit programme en C utilisant 2 buffers de 8 caractères. Faites afficher l'emplacement des 2 buffers et leur contenu. Copiez dans le second buffer 20 caractères. Faites de nouveau afficher les mêmes informations. Que constatez-vous ?
2. Compilez en utilisation l'option -g : pour gdb
   a. Relancer l'exécution avec gdb : gdb prog.
   b. Avancez dans votre programme fonction par fonction. Regardez ce qu'il se passe au niveau de la mémoire :
      commande sous gdb : x /nfu adresse_variable
      avec n : nb d'occurrences
      f : format (x -> hexa, i -> instruction machine)
      u : taille (b: octet, h : 2 octets, w : 4 octets, g : 8 octets)



Rappel :

Dans la pile, on retrouve les différentes variables locales des fonctions.

3. Créer un programme en C ayant 2 variables : char passwd[16]; int ok=0;
   On passe en argument un nom, qui est recopié dans passwd. Puis on compare passwd par rapport à "motdepasse". Si identique, ok=1, sinon il reste à 0. On affiche: mot de passe OK, ou non.
   En utilisant gdb, regardez ce qu'il se passe. Peut-on toujours avoir mot de passe OK même si on ne connait pas le "motdepasse" ?