Examen de Sécurité des Systèmes d'Exploitation

Mercredi 19 Janvier 2005
Seules les notes de cours sont autorisées (pas de livre donc)
Les parties sont indépendantes

1 Mise en jambes

Indiquez très brièvement pourquoi les mots de passe ci-dessous sont mauvais.

a) 08121975
b) iZ4-
c) Vls=2.s$jv,FvZ9 !Bs9(f.Xfo_3 :Of02
d) toto
e) zosw
f) W|§ &°a=
g) kaSqwUiV
h) Ljé1F-.v

Donnez un exemple de bon mot de passe.

2 Un bug ?

Récemment, quelqu'un a rapporté aux développeurs de la distribution Linux Debian qu'«il y a un bug critique dans librairie C» [1]. En voici une traduction :

```
Sujet: libc6: Les utilisateurs normaux peuvent supprimer des fichiers
       appartenant à d'autres utilisateurs, y compris ceux de root.
$ cd ~                    # répertoire personnel de l'utilisateur normal
$ su                      # devenir root
# touch XXX               # créer un fichier appartenant à root
# chmod 700 XXX
# exit                    # redevenir utilisateur normal
$ rm -f XXX               # le fichier est supprimé
```

Est-ce un bug ? La commande `chmod` y change-t-elle quelque chose ?

La personne a imputé le «bug» à la librairie C. Que cela soit un bug ou non, avait-elle raison d'incriminer la librarie C ? Sinon, quoi d'autre faudrait-il incriminer ?

3 Agents SSH et passphrases

Pour se connecter par ssh depuis une machine A vers une machine B, on peut utiliser une simple authentification par mot de passe. Quel problème se pose si un malfaiteur prend le contrôle de la machine B ?

Pour entre autres éviter un tel problème, on a vu qu'on pouvait utiliser un mécanisme de clé publique / clé privée pour s'authentifier auprès de B sans utiliser de mot de passe. Quel problème se pose si un malfaiteur parvient à lire le disque dur de la machine A ?

[1] Vous pourrez aller voir `http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=280632`

Pour éviter ce désagrément, on peut indiquer une *passphrase* au moment de la génération de la clé privée. Seule la clé privée est stockée sur le disque dur, et elle ne «fonctionne» pas seule : il faut la combiner avec la bonne passphrase pour obtenir une *identité* qui, elle, permet de répondre au challenge de B. Si le disque dur de la machine A a été espionné, en quoi est-ce alors différent de l'authentification par mot de passe ?

Pour éviter d'avoir à taper sans arrêt la passphrase, on utilise temporairement un *agent ssh* qui conserve en mémoire l'identité créée à partir de la clé privé et la passphrase : c'est un simple programme tournant en arrière-plan auquel le programme `ssh` demande de répondre pour lui au challenge de B. Cette demande est faite via une socket locale située par exemple dans `/tmp/ssh-RYvuuT6097/agent.6097` :

```
$ ls /tmp/ssh-RYvuuT6097 -la
drwx------  2 utilisateur groupe   4,0K 2005-01-18 14:05 .
drwxrwxrwt  6 root        root     4,0K 2005-01-18 14:05 ..
srwxr-xr-x  1 utilisateur groupe      0 2005-01-18 14:05 agent.6097
```

Pourquoi un autre utilisateur ne peut pas l'utiliser ? Qui pourrait tout de même l'utiliser et comment ? Qui pourrait lire l'identité et comment ?

4 Une faille

Sur le système TENEX, on pouvait protéger les fichiers par des mots de passe de 8 lettres codées sur 7 bits. Ces mots de passe étaient stockés en clair sur le disque dur de manière inaccessible à l'utilisateur.

Combien de possibilités de mots de passe cela représente-t-il ?

L'appel-système pour ouvrir un fichier ressemblait à ceci :

```
fd = open("fichier",O_RDONLY,"motdepasse");
```

L'implémentation dans le noyau de cet appel système ressemblait à ceci :

```
int sys_open(const char *path, int how, const char *passwd) {
  char real_passwd[8];

  /* first of all, check password */
  get_real_passwd(real_passwd);
  if (strncmp(real_passwd,passwd,8) != 0)
    /* bad password, permission denied */
    return -EPERM;

  ... /* good password, open file */
}
```

Où `get_real_passwd()` est une fonction qui récupère sur le disque dur le mot de passe en clair et le stocke dans la chaîne de caractères donnée, que l'on compare simplement au mot de passe fourni par le programme en utilisant la fonction `strncmp()` [2].

D'autre part, pour faciliter le débugage des programmes allouant de la mémoire, les *gros*[3] blocs mémoire alloués par la fonction `malloc()` sont «gardés» : toute lecture ou écriture (légèrement) en-dehors du bloc alloué (que ce soit par le programme lui-même ou par le noyau) génère un signal SIGSEGV, que le programme peut détourner pour réagir en conséquence.

Sachant tout cela, décrivez (sans l'écrire précisément !) un programme qui n'aurait besoin de tester que très peu de mots de passe. Combien en aurait-il à tester au pire ? Comment corriger la faille dans le noyau ?

[2] `strncmp(s1,s2,n)` compare deux par deux les caractères des deux chaînes `s1` et `s2`, et renvoie une valeur non nulle dès qu'elle remarque une différence (économisant ainsi le parcours entier des chaînes). Si elle arrive en même temps à la fin des deux chaînes (ou qu'elle a atteint la limite de taille `n`) sans constater de différence, `0` est renvoyé

[3] de taille au moins 4096 octets