Master Cryptologie et Sécurité Informatique
Examen de Sécurité des Systèmes d'Exploitation (MAI935EX)
Samuel Thibault
Lundi 19 Décembre 2006, durée: 1h30

Seules les notes de cours sont autorisées (pas de livre donc).
Le sujet fait 3 pages.
Les parties sont indépendantes.
N'hésitez pas à demander des éclaircissements éventuels

1 Mise en jambes

Question 1.1 *Indiquez très brièvement pourquoi les mots de passe ci-dessous sont plus ou moins mauvais.*

1. Fv2àf ;0.
2. FjwZFkav
3. 14/VII/89
4. L3) ;Fjli3j4650c.Az+4fLxvjJc9-f2k
5. ;4fZ
6. ;/,3Zw*_

Question 1.2 *Donnez un exemple de bon mot de passe. Estimez très grossièrement le nombre de mots de passe qu'un attaquant aurait à essayer avant de le trouver par attaque brute.*

Alice et Joe appartiennent à un même groupe `users` et seul Joe appartient au groupe `joe`.

```
drwxr----- 185 joe users 16K 2006-12-07 18:06 /home/joe
-rw----r-- 185 joe users 232 2006-12-07 18:06 /home/joe/tata
drwx--x--- 185 joe joe    4K 2006-12-07 18:06 /home/joe/répertoire
-rw----r-- 185 joe joe   128 2006-12-07 18:06 /home/joe/répertoire/toto
```

Question 1.3 *Quelles permissions Joe devrait-il par exemple changer pour qu'Alice puisse* lire *les fichiers* `tata` *et* `toto`*? (minimisez les changements !)*

2 Grosses matrices

Soit le programme suivant, qui ne fait qu'allouer 16 matrices 2D :

```
#define N 1024
#define T 16
typedef char matrix[N][N];
matrix *t[16];
int main(void) {
        int i;
        for (i=0; i<T; i++) {
                t[i] = malloc(sizeof(matrix));
                if (!t[i]) {
                        perror("allocation échouée");
                        exit(1);
                }
        }
        pause();
}
```

Question 2.1 *Lorsque l'on exécute ce programme, on constate que la consommation mémoire n'a quasiment pas changé (quelques Ko seulement). Comment est-ce possible ?*

On essaie maintenant de passer à des matrices 3D en changeant simplement la définition du type `matrix` :

```
typedef char matrix[N][N][N];
```

Question 2.2 *Le programme se termine maintenant avec une erreur* `allocation échouée`*. Pourtant, si l'on essaie de diminuer* `T` *à 2, l'erreur n'apparaît plus et la consommation mémoire est toujours aussi faible. Comment est-ce possible ?*

3 Rendu de TP

Supposons qu'un enseignant voudrait que ses étudiants lui rendent un TP en le déposant dans un répertoire particulier de son compte Unix de la Fac. Les conditions suivantes sont requises :

- Seuls ces étudiants peuvent déposer leur TP (d'autres étudiants qui ne suivraient pas le module ne doivent pas pouvoir en déposer).
- Un étudiant peut relire et modifier le TP qu'il a déposé, mais pas ceux des autres.
- Seul l'enseignant peut lire les TPs de tous les étudiants.

Question 3.1 *Comment réaliser cela ? (que demander à l'administrateur, quelles permissions donner au répertoire en question et comment les étudiants doivent-ils déposer leur TP ?)*

4 Gestion de clé et suspend to disk

On a vu en cours que l'on pouvait associer une *passphrase* à sa clé ssh, la clé ssh n'étant alors fonctionnelle que si l'on tape la *passphrase*.

Question 4.1 *Pourquoi fait-on cela ?*

Afin d'éviter d'avoir à retaper sans arrêt la *passphrase*, on utilise en général un agent ssh qui la demande une seule fois et s'occupe de conserver la clé ssh fonctionnelle en mémoire.

Le système Linux dispose d'une fonction *suspend to disk* : le contenu de la mémoire est stocké dans le *swap* sur le disque, et l'ordinateur peut alors être éteint. Lorsqu'on le rallume, le contenu de la mémoire est restauré depuis le disque, et le système peut ainsi continuer à fonctionner comme si de rien n'était.

Question 4.2 *Y a-t-il vraiment besoin de sauver* tout *le contenu de la mémoire ? Quelles zones pourraient ne pas être sauvegardées ?*

Question 4.3 *En quoi utiliser le* suspend to disk *est-il risqué ? Imaginez au moins 2 solutions à cela.*

5 Serveur X et authentification

Sous Unix, le fonctionnement de l'interface graphique X-Window est basée sur le modèle client-serveur :
- Le serveur X contrôle carte vidéo, clavier et souris,
- les clients X (les applications) se connectent au serveur X et lui demandent d'ouvrir des fenêtres, tracer des traits, récupérer les pressions de touches etc...

Le modèle client-serveur permet aux clients de ne pas forcément s'exécuter sur la machine locale : ils peuvent s'exécuter sur une autre machine et utiliser une connexion TCP pour se connecter au serveur X.

Question 5.1 *Lors de la connexion, le serveur X exige que le client s'authentifie, pourquoi ?*

À l'origine, cette authentification était relativement simple : à l'aide de la commande `xhost` on spécifiait une liste d'utilisateurs autorisés à se connecter en local, et on ajoutait éventuellement une liste des machines depuis lesquelles des clients étaient autorisés à se connecter via TCP.

Question 5.2 *Pour quelles raisons cette méthode n'est pas raisonnable ?*

De nos jours, on utilise un *cookie* : une clé de 128 bits connue par le serveur et stockée dans un fichier `.Xauthority` dans le *home* de l'utilisateur du serveur X. Lorsque le client se connecte (localement ou via TCP), il envoie simplement la clé au serveur pour s'authentifier.

Question 5.3 *Quelles permissions donner au fichier* `.Xauthority`*? Imaginez une situation où il pourrait être intéressant de donner des permissions à un certain groupe.*

Question 5.4 *Pourquoi cette méthode n'est pas beaucoup plus raisonnable ? Quelle solution proposeriez-vous ?*