Architecture d'un système d'exploitation (UE S3)

TP4

Introduction à la sécurité informatique en environnement HPC

Etudiant : Romain Pereira *Encadrant :* M. F. Combeau

17/11/2018

Table des matières

1 Réponses aux questions 1 à 22 (faites en cours)

4. Pour se connecter, on a utilisé l'authentification système, à savoir, les **Pluggable Authentication Modules (PAM)**

5. Le mot de passe *root* est stocké dans le fichier '/etc/shadow' sous forme d'un hachage. Le mot de passe est haché avant d'être stocké car il est stocké localement (chaque machine connecté au réseau possède le fichier '/etc/shadow' localement).

6. Je ne peux pas déterminer facilement le mot de passe de *Bob*, car il est également stocké sous forme de hash. Cependant, à l'aide d'une méthode de brute force, on pourrait eventuellement déterminer des chaîne de caractères qui ont le même hash que celui de *Bob*. Le mot de passe de *Bob* serait donc potentiellement l'une de ces chaînes de caractères, ou pas.

On ne peut pas déterminer le mot de passe d'*Alice*, car elle n'en a pas. En effet, le champ devant correspondre à son mot de passe dans le fichier '/etc/shadow' est ***** (ou **!!**). Les fonctions de hachages utilisées par les PAM ne donneront jamais des hashs avec ces caractères ('*' et '!'). En spécifiant des caractères impossibles à obtenir par hachage, on désactive de façon indirecte l'authentification par mot de passe pour l'utilisateur : jamais un mot de passe ne donnera ce hash. L'utilisateur est en quelque sorte 'banni'.

7. Si l'on passe cette ligne de **sufficient** à **required**, alors *root* ne pourra plus se logger s'authentifier. En effet, l'identifiant utilisateur (*uid*) de *root* vaut 0. A la ligne suivante, on refuse l'authentification des utilisateurs dont l'uid est inférieur à 1000, le test ne passera donc pas. est donc inférieur à 1000 : les tests suivant ne passeront pas.

Finalement, *root* ne pourra plus se logger (à cause de la ligne *pam_ deny.so*)

8. Oui, on arrive à se connecter sur le compte de *Bob* à partir du compte *root*

9. La connexion sur le compte *bob* à partir de l'utilisateur *root* (uid=0) est bien tracée dans les logs.

```
> less /var/log/secure
[...]
Accepted password for bob form ::1 port 47404 ssh2
pam_unix(sshd:session) : session opened for user bob by (uid=0)
[...]
```

10. **requisite** et **required** rende obligatoire le succès de l'entrée PAM.

Avec **requisite**, le processus d'authentification continue quand même les instructions suivantes, même si l'authentification a échoué.

Avec **required**, le processus d'authentification s'arrête si l'instruction renvoie faux.

Danns cet exemple, rien ne change en passant de **requisite** à **required** car dans tous les cas, l'authentification s'arrête au 'deny' qui suit.

En passant de **requisite** à **sufficient**, si la ligne 'pam_succeed_if.so' renvoie vrai, alors l'authentification est vérifié, renvoie vrai.

Une entrée **sufficient** est 'suffisante' : sa validité seule permet de valider l'authentification.

La ligne avec le 'deny' ne sera donc pas appelé si le test est passé : l'utilisateur pourra s'authentifier pourvu qu'il ait un **'uid' supérieur ou égal à 1000**.

11. Alice n'a pas de mot de passe (cf '!!' et '*')

```
> ssh alice@localhost
Enter passphrase for key '/root/.ssh/id_rsa'
```

Donc ssh utilise son propre système d'authentification, et non pas les PAM systèmes. La clef privé ssh (RSA) a été chiffré à l'aide d'un mot de passe (pour éviter le vol de clef)

Une fois connecté sous Alice, on a la clef dans '.ssh/authorized_keys'

Logs :

```
sshd : Accepted publickey for alice from ::1 port 47410 ssh2: RSA SHA256:/JdhKRDYT.
sshd : pam_unix(sshd:session): session opened for user alice by (uid=0)
```

12. Protection '775' ('002' est le complément à 8 du masque octal)

```
> drwxrwxr-x 2 bob 4096 19 nov. 15:22
> umask
'0002'
```

13. Oui on a le droit de créer, car on est sous root (root à le droit d'écrire partout)

```
> -rw-r--r--. 1 root    0 19 nov. 15:52 toto
```

Ce ficher appartient à root Linux applique le masque octal (umask), mais par défaut, il est non executable par mesure de sécurité.

14. Le système notifie qu'on a pas les droits d'écriture sur le fichier (entant que 'bob') Mais le fichier a été supprimé, car 'bob' à le droit d'écriture dans le dossier

```
> rm toto
'rm : supprimer fichier vide (prot g  en  criture ) "␣toto␣" ? y
```

15. Il suffit d'utiliser l'utilitaire 'chmod'

```
> chmod 1775 test
> drwsr-sr-t. 1 bob    0 19 nov. 15:52 test
```

16. Oui, on a toujours le droit, et le fichier appartient à 'root'

```
> -rw-r--r--. 1 root    0 19 nov. 15:52 toto
```

17. Oui on a toujours le droit.

18. Il suffit d'utilliser l'utilitaire 'chown'

```
 > chown root test
 > drwsr-sr-t. 1 root    0 19 nov. 15:52 test
```

19. Non **bob** ne peut pas l'effacer, car il y a le sticky bit (le 't') sur le repertoire. Il faut être propriétaire du dossier pour supprimer un fichier. Dans ce cas, **root** est le propriétaire du dossier, donc **bob** ne peut pas effacer de fichiers.

20. */bin/passwd* : modifie des fichiers auxquelles seul root à accès (ex : */etc/shadow*)

21. **Bob** peut partager son fichier en utilisant les ACL.

Pour pouvoir parcourir le dossier

```
> setfacl -m u:alice:rx dossier
```

Pour pouvoir ecrire dans le fichier du dossier

```
> setfacl -m u:alice:rw pour_alice
```

22. On remarque que l'utilisateur 'sftp' est ch-rooté dans '/home/sftp'

Lors d'une connection ssh :

```
This service allows sftp connections only.
Connection to localhost closed.
```

(le service n'autorise que des connections sftp)

Lors d'une connection sftp :

```
'Connected to localhost'
```

Avec l'utilisateur **bob**, le chroot est '/', alors qu'avec **sftp**, le chroot avec '/home/stfp'

2 Réponses aux questions 23 à 32 (bonus)

23. Oui, l'utilisateur root put prendre l'identité de n'importe quel utilisateur, **sans même** avoir à entrer un mot de passe (pas d'authentification requise).

```
> su bob
```

"su bob" change l'utilisateur courant, en gardant le même shell (et la plupart des variables d'environnement)

"su - bob" change l'utilisateur courant et crée un nouveau shell, en redefinissant la plupart des variables d'environnement.

24. Oui, **bob** peut prendre l'identité de **root**, mais seulement s'il connait le mot de passe de **root** : une authentification est demandée.

```
[bob@localhost ~] > su root
Mot de passe:
[root@localhost bob] > _
```

25. La commande 'su' peut être desactivé pour tous les utilisateurs qui ne font pas parti du groupe 'wheel'. Il suffit de configurer les PAM, en éditant la configuration dans **/etc/pam.d/su**, et en décommentant la ligne 15 suivante :

```
 8 # Uncomment this to force users to be a member of group root
 9 # before they can use ‘su’. You can also add "group=foo"
10 # to the end of this line if you want to use a group other
11 # than the default "root" (but this may have side effect of
12 # denying "root" user, unless she’s a member of "foo" or explicitly
13 # permitted earlier by e.g. "sufficient pam_rootok.so").
14 # (Replaces the ‘SU_WHEEL_ONLY’ option from login.defs)
15 auth       required   pam_wheel.so
```

26. Oui, **bob** peut passer **root** via ssh s'il connait le mot de passe.

Pour l'en empêcher, on peut configurer **sshd** en éditant le fichier **/etc/ssh/sshd_config**

Il suffit de modifier la ligne 38 :

```
[...]
 37 #LoginGraceTime 2m
 38 #PermitRootLogin yes
 39 #StrictModes yes
[...]
```

par :

```
[...]
 37 #LoginGraceTime 2m
 38 PermitRootLogin no
 39 #StrictModes yes
[...]
```

Puis il faut redémarrer le serveur ssh pour que la configuration soit bien prise en compte

```
[root@localhost ~]# /etc/init.d/sshd restart
Stopping sshd: [ OK ]
Starting sshd: [ OK ]
```

27. Non, **bob** n'est pas autorisé à utiliser la commande **sudo**

```
[bob@localhost ~]# sudo ls
Mot de passe:
bob n’appara t pas dans la liste des sudoers. Cet    vnement    sera signal .
```

Le fichier de configuration de la liste des 'sudoers' se situe dans '/etc/sudoers'

On peut ajouter **bob** à la liste en ajoutant cette ligne au fichier de configuration :

```
bob ALL=(ALL) ALL
```

Par défaut, la configuration '/etc/sudoers' autorise tous les membres du groupe 'sudo' à utiliser la commande **sudo**.

On peut donc autoriser **bob** à utiliser 'sudo' en l'ajoutant au groupe 'sudo' :

```
[root@localhost ~]# sudo adduser bob sudo
```

28. Oui, on peut autoriser **bob** à utiliser certaines commandes uniquement en tant que root.

Il suffit d'éditer le fichier '/etc/sudoers' et d'y ajouter/modifier la ligne concernant **bob**.

On peut aussi l'autoriser à éxecuter des commandes en tant que **root** sans qu'il n'ait à rentrer son mot de passe. Exemple :

```
bob ALL=(ALL) NOPASSWD: cat /var/log/secure
```

29. Pour récuperer le shell d'Alice, une façon de faire est de récupérer les informations d'Alice via **getent**. La 7ème entrée correspond au shell.

```
[root@localhost ~]# getent passwd | grep alice | cut -d : -f 7
/bin/bash
```

30. Si l'on essaye de modifier le shell d'**alice** par '/bin/false', en tant qu'utilisateur (**alice** par exemple), cela nous est refusé.

```
[alice@localhost ~]# chsh -s /bin/false alice
chsh: "/bin/false" n'apparait pas dans /etc/shells
Utiliser chsh -l pour afficher la liste.
```

Par contre, si on essaye de changer le shell d'**alice** entant que **root** :

```
[root@localhost ~]# chsh -s /bin/false alice
chsh: Avertissement, "/bin/false" n'apparait pas dans /etc/shells
L'interpr teur a ete modifie.
```

Autrement dit, le shell a bien été changé, mais nous avons eu un message d'avertissement car le binaire '/bin/false' n'est pas enregistré en tant que shell valide sur le système.

L'executable '/sbin/nologin' est un shell enregistré, on peut donc le définir en tant que shell sans avertissement.

Le comportement final est le même : l'utilisateur n'a pas de shell.

31. Si l'on change le shell d'**alice** par '/bin/rbash', le shell par défaut d'**alice** sera un bash restreint. Ce bash à l'utilisation restreinte ajoute une couche de sécurité.

32. '/bin/rbash' est un lien symbolique vers le binaire 'bash'.

```
[root@localhost ~]# ls -la /bin/rbash
lrwxrwxrwx. 1 root root 9 5 nov. 2017 /bin/rbash -> /bin/bash
```

Autrement dit, le comportement du programme 'bash' varie en fonction de son nom (argv[0]).