Script Shell TP

Dans le cadre de mon travail de surveillance système, j'ai créé trois scripts shell pour surveiller les fichiers, les processus et les services. Les scripts ont été conçus pour être simples et faciles à utiliser pour les administrateurs système, afin qu'ils puissent rapidement et facilement surveiller les différents aspects du système.

Description des scripts

Le premier script, `script1.sh`, surveille les fichiers d'un répertoire spécifié et affiche les changements dans la console en temps réel. Il est utile pour surveiller les modifications de fichiers critiques tels que les fichiers de configuration ou les journaux système.

Le deuxième script, `script2.sh`, surveille les processus en cours d'exécution et affiche leur utilisation de CPU, de mémoire et d'E/S. Cela permet aux administrateurs système de surveiller les processus qui utilisent trop de ressources système ou qui pourraient être des signes de problèmes plus graves.

Le troisième script, `script3.sh`, surveille les services système en cours d'exécution et affiche leur état (activé/désactivé) ainsi que leur temps de fonctionnement. Cela permet aux administrateurs système de surveiller les services qui sont critiques pour le système et de s'assurer qu'ils sont toujours en cours d'exécution.

Difficultés rencontrées et solutions

J'ai rencontré plusieurs difficultés lors de la création de ces scripts. L'une des principales difficultés était de m'assurer que les scripts étaient sécurisés et n'offraient pas d'accès indésirable aux informations système sensibles. J'ai résolu ce problème en créant un utilisateur spécifique pour les scripts et en utilisant des autorisations appropriées pour les fichiers, les processus et les services.

Une autre difficulté était de s'assurer que les scripts étaient assez simples pour être utilisés facilement par les administrateurs système, tout en offrant suffisamment de fonctionnalités pour être utiles. J'ai résolu ce problème en ajoutant des commentaires clairs dans les scripts pour expliquer leur fonctionnement et en testant les scripts pour m'assurer qu'ils étaient faciles à utiliser.

Limitations et extensions éventuelles

Bien que ces scripts soient utiles pour surveiller les fichiers, les processus et les services, ils ont certaines limitations. Par exemple, ils ne peuvent pas surveiller les activités du réseau ou les vulnérabilités de sécurité. Pour améliorer ces scripts, je pourrais ajouter des

fonctionnalités de surveillance réseau ou de sécurité, ou les intégrer à un outil de surveillance plus complet.

Conclusion

Dans l'ensemble, ces scripts shell sont utiles pour surveiller les fichiers, les processus et les services d'un système. J'ai surmonté les difficultés rencontrées en utilisant des autorisations appropriées et en testant les scripts pour leur simplicité d'utilisation. Bien qu'ils aient certaines limitations, ces scripts pourraient être étendus pour offrir davantage de fonctionnalités de surveillance système.

Partie 1

a) pwd
b) date -d "now" +"%d/%m/%Y %H:%M:%S"
c) ls -l | grep '^-' | awk '{sum += $5} END {printf "%d fichiers, taille totale : %.2f Mo\n", NR, sum/1024/1024}'
d) ls -l | grep '^d' | wc -l
e) tree -L "$profondeur"
f) find . -type d -size +"$taille"k -exec du -sh {} + | sort -h
g) cd documents/
h) find . -type f $recherche avec recherche+=" -newermt $date
i) find . -type f $recherche avec recherche="-newerct $date
j) find . -type f $recherche avec recherche+=" -oldermt $date
k) find . -type f $recherche avec recherche=" -olderct $date
l) find . -type f -size +"$taille"c
m) find . -type f -size +"$taille"M
n) find . -type f -size -"$taille"c
o) find . -type f -size -"$taille"M
p) find . -type f -name ".$extension"
q) find . -maxdepth 1 -type f -name ".$extension"
r) grep -r "$chaine"

Partie 2

a) ps -ef | awk '{print $0 " - propriétaire: " $1}'
b) ps -ef | awk '{if ($8 ~ /R|S/) print $0 " - propriétaire: " $1}'
c) ps -ef | grep $1 | grep -v grep | awk '{if ($1 == ""$1"") print $0}'
d) ps -eo pid,ppid,cmd,%mem,%cpu,user --sort=-%mem | head | awk '{print $0 " - propriétaire: " $6}'

e) ps -eo pid,ppid,cmd,user --sort=user | awk '{if ($3 ~ /"$1"/) print $0 " - propriétaire: " $6}'
f) ps -eo pid,%mem,command --sort=-%mem | head

Partie 3

a) systemctl list-unit-files | grep service | awk '{print $1}' | sed 's/\.service//'
b) systemctl list-units --type=service | grep running | awk '{print $1}' | sed 's/\.service//'
c) systemctl status $service_name.service
d) systemctl show -p Description $service_name.service | awk -F'=' '{print $2}'

Partie 4

Voir Interface.sh

Partie 5

1. Pour permettre à un simple utilisateur de consulter toutes les informations (fichiers, processus, services) sans pour autant avoir la possibilité d'interagir avec eux comme le ferait un administrateur, vous pouvez créer un utilisateur spécifique et lui donner les autorisations appropriées.
   Voici les étapes pour créer cet utilisateur et configurer les autorisations nécessaires :
   1. Créez un nouvel utilisateur en utilisant la commande `adduser` :

```
sudo adduser consulter-info
```

2.
3. Ajoutez l'utilisateur au groupe `sudo` pour lui donner les autorisations nécessaires pour consulter les informations :

```
sudo usermod -aG sudo consulter-info
```

4.
5. Créez un nouveau groupe pour les autorisations supplémentaires, par exemple `info-group` :
   csharp

```
sudo groupadd info-group
```

6.
7. Ajoutez l'utilisateur au nouveau groupe en utilisant la commande `usermod` :
   csharp

```
sudo usermod -aG info-group consulter-info
```

8.
9. Modifiez les autorisations des fichiers et répertoires pertinents pour que le groupe `info-group` puisse y accéder :
   bash

```
sudo chgrp info-group /path/to/file/or/directory
sudo chmod g+r /path/to/file/or/directory
```

Cela permettra à l'utilisateur `consulter-info` d'accéder aux fichiers et répertoires pertinents, mais sans pouvoir les modifier.