Projet UNIX

O. Delgrange

Bloc complémentaire au Master d'Informatique
Bloc complémentaire au grade de Master Ingénieur Civil IG
Systèmes d'Exploitation
2020-2021

Généralités

- Les scripts doivent être rendus au plus tard le **vendredi 14 mai 2021** sur la plateforme d'E-learning (Moodle).
- Les scripts UNIX demandés peuvent être dévéloppés sur n'importe quel UNIX, mais ils doivent **impérativement** être réalisés dans le Bash. De plus il doivent fonctionner sur la machine utilisée lors des travaux pratiques (`tpos.umons.ac.be`), c'est sur cette machine qu'ils seront testés.
  Vous pouvez y transférer des fichiers en utilisant le protocole SFTP. Pour cela des clients gratuits existent pour toutes les plate-formes (Filezilla, gftp, ...).
- Si des commandes non vues au cours et non précisées dans ce document sont utilisées, préciser leur rôle et leur syntaxe.
- Les solutions doivent absolument être accompagnées de **textes explicatifs** décrivant vos méthodes de résolution.
- D'autres scripts peuvent être créés si nécessaire.
  Il est impératif que les scripts soient fournis sous la forme de fichiers textes standards et non sous la forme de fichiers en format propriétaire tels que `.doc`, `.rtf`...
  En ce qui concerne le texte explicatif, le fournir soit en format texte, soit le convertir en .pdf.
- Le travail est **strictement personnel**, il compte pour $\frac{1}{2}$ des points du cours « Systèmes d'Exploitation ». Veillez donc à **ne pas divulguer trop d'informations sur votre travail à vos camarades, vous pourriez être sanctionné !**

Script de surveillance d'activités terroristes

Réaliser une application client/serveur, tournant continuellement en tâche de fond, capable d'identifier des activités suspectes dans le cadre de la surveillance contre le terrorisme.

La recherche d'activités suspectes se base sur des mots qui peuvent attirer l'attention dans le cadre d'une enquête. Ces mots sont décrits par un fichier dictionnaire `dict.txt` qui contient un motif (une expression régulière) par ligne. Une chaîne de caractères qui concorde avec un de ces motifs est jugée *suspecte* et doit être signalée.

Exemple de fichier `dict.txt`

```
daesh
kamikaze
bombe
attaque
ceinture.*explosi.*
mitraillette
kalachnikov
détonateur
```

Le serveur `surveille` doit inspecter, toutes les x secondes (x est un paramètre du serveur), la liste des activités actuelles sur le système et écrire dans un fichier journal (fichier texte) le résultat de son observation en commençant par la date et le moment de l'analyse.

Les conditions pour qu'une activité soit suspecte sont :

1. le nom de la commande exécutée concorde avec le dictionnaire ;
2. un argument passé à la commande concorde avec le dictionnaire ;
3. l'activité possède un fichier régulier ouvert pour lequel :
   — le nom concorde avec le dictionnaire ;
   — le fichier ne vous est pas accessible en lecture (en pratique, le script devrait être exécuté par `root`) ;
   — le fichier contient un/des motifs du dictionnaire

Pour chaque activité suspecte, le script doit écrire dans le fichier journal :

— le numéro de processus, son propriétaire (ainsi que son nom complet) et groupe propriétaire, la commande exécutée et son état (actif, suspendu, . . .).
— toutes les raisons faisant penser que cette activité est suspecte ;
— si des fichiers contiennent des mots suspects, ils doivent être signalés et pour chaque fichier suspect, tous ses synonymes (liens physiques dans la même partition) doivent être recherchés et signalés.

La syntaxe d'utilisation du serveur est :

— `surveille start [-p` *pid*`] [-u` *user*`] [-d` x`]`
  → démarre le serveur (et crée les processus, fichiers, tubes éventuels,. . . utiles)
  Par défaut, toutes les activités du système sont concernées, on peut restreindre l'inspection à un processus (option `-p`) ou aux processus d'un utilisateur (option `-u`).
  L'option `-d` x permet de fixer le nombre de secondes entre chaque ré-inspection (par défaut, 10 secondes séparent deux inspections).
— `surveille stop`
  → stoppe le serveur.
  Tous les processus, fichiers et tubes impliqués doivent être correctement détruits.
— `surveille restart`
  → stoppe le serveur et le redémarre avec les mêmes paramètres initiaux.

Le script client `surevents`, lancé à la demande, affiche les événements qui se sont déroulés depuis la dernière fois où il a été lancé.

Exemple d'exécution

```
$ surveille start -d 5
Serveur lance
$ surevents
Date : Mon Mar 29 15:39:01 CET 2021
EVENT 1
  proc 1784, user lamin (Lamin Sahle), group guess, command : gedit, state : activable
  Fichier ouvert : /home/lamin/plan151113.txt, motifs rencontrés :
      détonateur attaque
    Ce fichier ne contient pas d'autre synonymes
EVENT 2
  proc 1899, user olivier (Olivier Dupont), group info, command : vi menu.txt, state : suspended
  Fichier ouvert : /home/olivier/commandes/menu.txt, motifs rencontrés :
      mitraillette
    Synonymes :
      /home/olivier/fc/m.txt
$ surevents
No EVENT
$ surveille stop
Serveur de surveillance arrêté
```

Puisque le fichier journal est un fichier texte, il pourrait être visualisé tel quel par l'utilisateur qui a les droits de lecture (dans un environnement en production, ce devrait être `root`). L'avantage du script `surevents` est cependant de ne plus réafficher les événements qu'il a déjà affichés lors d'un précédent appel (Il faut faire en sorte que ce soit le cas).

Indices

1. Certaines utilisations de la commande `ps` permettent de connaître la commande lancée ainsi que ses arguments.
2. La recherche de synonymes devant se dérouler dans la même partition que le fichier, il faut déterminer à quelle partition appartient le fichier : connaître le répertoire racine de la partition où se trouve le fichier.

   Pour cela, la commande `df` *ref* donne le répertoire racine de la partition dans laquelle se trouve le fichier de référence *ref*.
3. La commande `lsof` [1] (qui signifie « list open files ») permet de visualiser les fichiers ouverts par tous les processus qui tournent sur le système. Elle possède différentes options qui permettent (entre autres), de fixer le délai de rafraîchissement (option `-r`), de spécifier les colonnes que l'on désire visualiser (option `-F`),...

   Les fichiers utilisateurs ouverts par un processus peuvent être rapidement localisés grâce à la colonne « FD » : ce sont des descripteurs numérotés (0 pour l'entrée standard, 1 pour la sortie standard, 2 pour la sortie erreur standard, 3 pour le premier fichier ouvert, ...) suivi d'une lettre qui spécifie le mode d'ouverture du fichier.

   La colonne « TYPE » permet quant à elle de se restreindre uniquement aux fichiers réguliers (`REG`), ce qui évite de facto les entrées/sorties standards.
4. Une attente passive de n secondes est effectuée dans le Bash grâce à la commande `sleep` n.

Bon travail !

1. `lsof` n'est pas installée par défaut sur toutes les distributions Linux, il convient alors de l'installer