Rapport SYSG5 - Acces Control List

AZDAD Yassin
BOUAYAD Amer

22 November 2019

Contents

1 Introduction

En Linux, chaque fichier et dossier possède des permissions. Il existe un schéma traditionnel UNIX (et Linux) qui les décrit. Pour de nombreuses applications, ce schéma est suffisant. Cependant, certaines applications nécessitent un contrôle plus précis des autorisations accordées à des utilisateurs et à des groupes spécifiques. C'est dans ce contexte qu'interviennent les ACL, autrement dit, les Access Control List.

La mise en place des ACL permet une gestion fine des accès des utilisateurs, des groupes, aux répertoires et aux fichiers d'une partition qui dispose d'un "file system" qui accepte les ACL.

Afin de démontrer l'utilisation et l'intérêt des ACL, nous avons créé plusieurs scripts à ce propos. Nous avons travaillé sous la distribution Linux-Opensuse, sur une partition formatée en ext4.

1.1 Prérequis

Afin de pouvoir utiliser et attribuer des ACL, il faut, préalablement,vérifier que le paquet des ACL est installé, et dans le cas échéant où il ne le serait pas, l'installer. Pour ce faire on peut faire usage de la commande :

```
if grep "CONFIG_FS_POSIX_ACL=y" /boot/config-*; then echo "OK"; else
echo apt-get install acl; fi
```

2 Qu'est ce qu'une ACL ?

2.1 Définition

Une ACL est une Access Control List. C'est-à-dire que c'est une liste de permissions accordées sur un fichier ou un dossier à certains utilisateurs. Il existe 3 types de permissions :

- Le droit de lecture - R
- Le droit d'écriture - W
- Le droit d'exécution - X

Il existe également 3 types d'utilisateurs auxquels on peut accorder ou révoquer des permissions.

- On peut accorder une/des permission(s) à un utilisateur grâce à son nom.
- On peut accorder une/des permission(s) à un groupe incluant tous ses utilisateurs grâce à son nom.
- On peut accorder une/des permission(s) à tous les autres utilisateurs qui ne correspondent à aucune ACL.

2.2 Illustrations

2.2.1 Visualisation ACL

Afin de visualiser les ACL attribués à un fichier ou un dossier, il suffit d'utiliser la commande :

```
getfacl nom_de_fichier
```

Cette commande va nous permettre de visualiser les ACL attribuées à nom_de_fichier comme on peut le voir dans l'image ci-dessous.

```
user0@linux-ily4:~/Bureau> getfacl nom_de_fichier
# file: nom_de_fichier
# owner: user0
# group: users
user::rw-
user:yassin:rwx
group::r--
group:etudiants:rw-
mask::rwx
other::r--
```

On peut y voir que l'utilisateur yassin dispose de tous les droits (RWX) sur ce fichier, et que les utilisateurs du groupe étudiants ont seulement le droit de lecture et écriture(RW-) et, pour finir, que les autres utilisateurs (ceux qui n'ont pas déjà été nominés par une ACL) n'ont aucun droit dessus.

2.2.2 Suppression d'ACL

Il est aussi possible de supprimer des ACL attribuées à un fichier ou un dossier avec l'option -b. Par exemple via la commande :

```
setfacl -b nom_de_fichier
```

Il est également possible de supprimer une seule ou plusieurs parties des ACL attribuées à un fichier. Pour ce faire, il faut utiliser l'option -x. Par exemple via la commande :

```
setfacl -x u:michel,g:esi fichier2
```

Cette commande supprimera les permissions de l'utilisateur michel et du groupe esi attribuées au fichier fichier2.

2.2.3 Autres options possibles

Il est également possible de copier les ACL d'un fichier file1 à un autre fichier file 2 via la commande :

```
getfacl file1 | setfacl -set-file=- file2
```

Il est également possible de supprimer les ACL par défaut d'un fichier file1 avec l'option -k. Par exemple, via la commande :

```
setfacl -k file1
```

2.2.4 Attribution et vérification d'ACL

Afin d'illuster et de démontrer l'attribution et la vérification d'ACL, nous avons mis au point un script automatique qui permet de poser des ACL sur des fichiers et de prouver que seuls les utilisateurs autorisés ont la ou les permissions qui leurs ont été attribuées.

scriptACL.sh

```
user0@linux-ily4:~/Bureau/AclAzdadBouayad/Script> ./scriptACL.sh
 Creation d'un fichier test nommé file1

# file: file1
# owner: user0
# group: users
user::rw-
group::r--
other::r--

Permet de voir les ACL par défaut du fichier file1

Creation d'un user Yassin

Creation d'un user Amer

Creation d'un group esiGroup

On attribut l'utilisateur Amer au groupe esiGroup

Appuyez sur enter pour lancer <setfacl -m u:yassin:rwx,g:esiGroup:r--,o::--- file1>


Droit attribue à file1

# file: file1
# owner: user0
# group: users
user::rw-
user:yassin:rwx
group::r--
group:esigroup:rw-
mask::rwx
other::---

Appuyez pour executer file1 en tant que Yassin


coucou moi je peux exécuter et pas toi
Appuyez pour essayez d'éxécuter file1 en tant que Amer


bash: ligne 1: ./file1: Permission non accordée
```

Via ce script et ses sorties, on peut voir et démontrer plusieurs choses :

- Qu'on peut attribuer des ACL à un fichier.

```
Creation d'un fichier test nommé file1

# file: file1
# owner: user0
# group: users
user::rw-
group::r--
other::r--
```

```
Droit attribue à file1

# file: file1
# owner: user0
# group: users
user::rw-
user:yassin:rwx
group::r--
group:esigroup:rw-
mask::rwx
other::---
```

- Que l'utilisateur Yassin peut exécuter le fichier file1.

```
Appuyez pour executer file1 en tant que Yassin
coucou moi je peux exécuter et pas toi
```

- Que l'utilisateur Amer ne peut pas exécuter le fichier file1.

```
Appuyez pour essayez d'éxécuter file1 en tant que Amer
bash: ligne 1: ./file1: Permission non accordée
```

Voici la sortie totale de l'exécution du script scriptACL.sh

sortie du script scriptACL.sh

```
user0@linux-ily4:~/Bureau/AclAzdadBouayad/Script> ./scriptACL.sh
 Creation d'un fichier test nommé file1

# file: file1
# owner: user0
# group: users
user::rw-
group::r--
other::r--

Permet de voir les ACL par défaut du fichier file1

Creation d'un user Yassin

Creation d'un user Amer

Creation d'un group esiGroup

On attribut l'utilisateur Amer au groupe esiGroup

Appuyez sur enter pour lancer <setfacl -m u:yassin:rwx,g:esiGroup:r--,o::--- file1>


Droit attribue à file1

# file: file1
# owner: user0
# group: users
user::rw-
user:yassin:rwx
group::r--
group:esigroup:rw-
mask::rwx
other::---

Appuyez pour executer file1 en tant que Yassin


coucou moi je peux exécuter et pas toi
Appuyez pour essayez d'éxécuter file1 en tant que Amer


bash: ligne 1: ./file1: Permission non accordée
```

3 Héritage d'ACL

Les fichiers incluent dans un dossier n'héritent pas automatiquement des ACL attribués à ce dossier. Pour que ces fichiers puissent en hériter, il faut attribuer des droits par défaut à ce dossier codé *-d*

Par conséquent, et suite à cela, les fichiers héritent de cet attribut d'ACL par défaut.

Afin de prouver ce concept, nous avons élaboré un script heritage.sh, qui a permis de démontrer que les fichiers créés dans un dossier à qui des ACL par défaut ont été attribuées, héritent eux-mêmes de ces ACL par défaut.

heritageACL.sh

```
user0@linux-ily4:~/Bureau/AclAzdadBouayad/Script> ./heritageACL.sh

 Héritage d'ACL par défaut démo


 Droit du dossier à la creation

# file: dossierACLDefault
# owner: user0
# group: users
user::rwx
group::r-x
other::r-x


 Voici les ACL par défaut du dossier créé


 Appuyez enter pour éxécuter setfacl -dm u::rw-,g::---,o::--- dossierACLDefault



 Voici les ACL du dossier dorénavant

# file: dossierACLDefault
# owner: user0
# group: users
user::rwx
group::r-x
other::r-x
default:user::rw-
default:group::---
default:other::---


 Création du fichier file dans le dossier dossierACLDefault


 Voici les ACL de ce fichier
# file: file
# owner: user0
# group: users
user::rw-
group::---
other::---
```

Ce script a pour but de créer un dossier dossierACLDefault et de lui attribuer des ACL par défaut pour tous les fichiers qui seront créés dedans. De plus, ce script permet de vérifier que cette opération s'est passé sans encombre.

Sortie script heritageACL.sh

```
user0@linux-ily4:~/Bureau/AclAzdadBouayad/Script> ./heritageACL.sh

 Héritage d'ACL par défaut démo


 Droit du dossier à la creation

# file: dossierACLDefault
# owner: user0
# group: users
user::rwx
group::r-x
other::r-x


 Voici les ACL par défaut du dossier créé


 Appuyez enter pour éxécuter setfacl -dm u::rw-,g::---,o::--- dossierACLDefault



 Voici les ACL du dossier dorénavant

# file: dossierACLDefault
# owner: user0
# group: users
user::rwx
group::r-x
other::r-x
default:user::rw-
default:group::---
default:other::---


 Création du fichier file dans le dossier dossierACLDefault


 Voici les ACL de ce fichier
# file: file
# owner: user0
# group: users
user::rw-
group::---
other::---
```

On peut facilement se rendre compte que le fichier file créé dans le dossier dossierACLDefault a hérité des ACL par défaut du dossier dossierACLDefault créé préalablement.

En effet, par défaut, les fichiers créés dans ce dossier étaient censés se voir attribuer le droit de lecture et d'écriture aux users et c'est bien ce qui s'est passé comme on peut le voir après la commande getfacl file.

4 Masque

Le masque est une des notions propres aux ACL étendues. Avant cela, il faut, bien entendu, faire une différence entre les ACL minimales et les ACL étendues.

4.1 ACL minimales

Les ACL minimales sont des ACL qui contiennent sémantiquement et exclusivement les 3 champs d'attribution :

- `ACL_USER_OBJ`: Attribution à un utilisateur.
- `ACL_GROUP_OBJ`: : Attribution à un groupe.
- `ACL_OTHER`: Attribution aux autres (ni utilisateur, ni dans le groupe)

Pour illuster une ACL minimale voici un exemple :

```
setfacl m u:yassin:rwx,g:amer:r-,o:- file1
```

On peut voir que le champ `ACL_USER_OBJ`= yassin, que le champ `ACL_GROUP_OBJ`=amer et que le champ `ACL_OTHER` n'a pas, et ne devrait, logiquement, jamais avoir d'assignation d'utilisateur étant donné qu'il s'agit de tous les utilisateurs qui ne se sont pas vu attribué des ACL et que par conséquent il n'y a pas de nom précis pour ces utilisateurs.

4.2 ACL étendues

Il existe des ACL étendues qui contiennent un autre champ qui est le masque. Le masque existe lorsqu'une ACL est attribuée à un dossier ou un fichier. Le masque est une synthèse des valeurs les plus permissives que possède un fichier doté d'une ACL L'intérêt du masque est de pouvoir préalablement assigner les permissions qui peuvent être accordées ou non, indépendamment de l'utilisateur. Autrement dit, seuls les droits du masque peuvent être accordés de manière effective.

4.3 Illustration

Afin de démontrer et d'illustrer ce concept, nous avons élaboré un script scriptMasque.sh que voici :

scriptMasque.sh

```
user0@linux-ily4:~/Bureau/AclAzdadBouayad/Script> ./scriptMasque.sh
Appuyez sur Enter pour éxecuter cat testMask


/home/user0/Bureau/AclAzdadBouayad/Script
Appuyez sur Enter pour éxecuter getfacl


# file: testMask
# owner: user0
# group: users
user::rw-
user:test1:rwx
group::r--
mask::rwx
other::r--

Appuyez sur Enter pour éxecuter getfacl


# file: testMask
# owner: user0
# group: users
user::rw-
user:test1:rwx                  #effective:--x
group::r--                      #effective:---
mask::--x
other::r--

Appuyez sur Enter pour éxecuter cat après masque


cat: testMask: Permission non accordée
```

Dans ce script, on attribue une ACL, de façon "normale" au fichier testMask, ensuite on visualise ses ACL afin de se rendre compte du masque par défaut attribuer à ce fichier. Puis on modifie ce masque de façon à ce que les droits les plus permissifs soient ceux d'exécution uniquement. Ensuite on essaie de lire le fichier afin de voir si notre masque à réussi à contrer les droits accordés à l'user test1

sortie scriptMasque.sh

```
user0@linux-ily4:~/Bureau/AclAzdadBouayad/Script> ./scriptMasque.sh
Appuyez sur Enter pour éxecuter cat testMask


/home/user0/Bureau/AclAzdadBouayad/Script
Appuyez sur Enter pour éxecuter getfacl


# file: testMask
# owner: user0
# group: users
user::rw-
user:test1:rwx
group::r--
mask::rwx
other::r--

Appuyez sur Enter pour éxecuter getfacl


# file: testMask
# owner: user0
# group: users
user::rw-
user:test1:rwx                  #effective:--x
group::r--                      #effective:---
mask::--x
other::r--

Appuyez sur Enter pour éxecuter cat après masque


cat: testMask: Permission non accordée
```

On peut se rendre compte grâce à la sortie générée à l'automatisation du script scriptMasque.sh qu'effectivement, malgré le fait que l'user test1 avait tous les droit (RWX), il n'a pas pu lire le fichier testMask car le masque a fait en sorte que les droits permissifs soient seulement ceux d'exécution et que donc seul ce droit pourra être effectué.

5 Localisation d'ACL

Les ACL sont stockés en tant qu'attributs étendus. Elles sont directement stockés dans les inodes en tant que méta-données. Et dans le cas échéant où la taille de la liste de ces ACL est trop conséquente, un seul bloc supplémentaire peut être utilisé pour le stockage de ces ACL.

Si le stockage des ACL nécessite un bloc de disque supplémentaire, le champ i_file_acl désigne le n de ce bloc supplémentaire.

Afin de prouver et démonter tout cela, nous avons élaboré un script scriptLocalisation.sh qui permet d'utiliser le debugger pour voir les stats d'un fichier créé avant et après une attribution d'ACL pour voir le stockage de cette ACL en tant que méta donnée dans l'inode.

scriptLocalisation.sh

```
#!/bin/bash

echo -e "\n\E[32mCréation d'un fichier file2...\E[0m"
touch file2

echo -e "\n\E[32mInsertion de texte dans file2...\E[0m"
pwd > file2

echo -e "\n\E[32mRécuperation des ACL par défaut de file2...\E[0m"
getfacl file2

echo -e "\n\E[32mCréation du group testGroup...\E[0m"
sudo groupadd testGroup

echo -e "\n\E[32mStockage du n° inode dans une variable i...\E[0m"
i=$(ls -li | grep file2 | cut -f1 -d ' ')

echo -e "\n\E[32mLe numéro d'inode de file2 est : \E[0m"
echo $i

echo -e "\n\E[32mStockage du n° de partition contenant /home dans une variable s...\E[0m"
s=$(lsblk | grep /home | cut -c3-7)

echo -e "\n\E[32mLe numéro d'inode de file2 est : \E[0m"
echo $s

echo -e "\n\E[32mPassage en mode debugger\E[0m"
sudo debugfs /dev/$s <<END
stat <$i>
q
END

echo -e "\n\E[32mAppuyez enter pour éxécuter setfacl -m g:testGroup:rwx, file2\E[0m"
read -p " "
setfacl -m g:testGroup:rwx, file2

echo -e "\n\E[32mRécuperation des nouveau ACL par de file2...\n\E[0m"
getfacl file2

echo -e "\n\E[32mPassage en mode debugger\E[0m"
sudo debugfs /dev/$s << END
stat <$i>
q
END

sudo groupdel testGroup
rm file2
```

Voici le résultat de la commande stat <n inode du fichier> avant de poser une ACL dessus. On peut se rendre compte qu'il n'y a pas d'attribut étendus.

```
user::rw-
group::r--
other::r--


Création du group testGroup...

Stockage du n° inode dans une variable i...

Le numéro d'inode de file2 est :
265019

Stockage du n° de partition contenant /home dans une variable s...

Le numéro d'inode de file2 est :
sda7

Passage en mode debugger
debugfs 1.42.11 (09-Jul-2014)
debugfs:  stat <265019>
Inode: 265019   Type: regular    Mode:  0644   Flags: 0x80000
Generation: 2113284098    Version: 0x00000000:00000001
User:  1000   Group:   100   Size: 44
File ACL: 0    Directory ACL: 0
Links: 1   Blockcount: 8
Fragment:  Address: 0    Number: 0    Size: 0
 ctime: 0x5dd7ecd1:b54efd60 -- Fri Nov 22 15:12:33 2019
 atime: 0x5dd7ecd1:b54efd60 -- Fri Nov 22 15:12:33 2019
 mtime: 0x5dd7ecd1:b54efd60 -- Fri Nov 22 15:12:33 2019
crtime: 0x5dd7ecd1:b54efd60 -- Fri Nov 22 15:12:33 2019
Size of extra inode fields: 32
EXTENTS:
(0):1147659
debugfs:  q

Appuyez enter pour éxécuter setfacl -m g:testGroup:rwx, file2


Récuperation des nouveau ACL par de file2...

# file: file2
# owner: user0
# group: users
user::rw-
group::r--
```

Et voici le résultat de la commande stat <n inode du fichier> après avoir attribuer une ACL au fichier. On peut se rendre compte que désormais il y'a des attributs étendus du fichier. On peut en conclure que les ACL sont bel et bien stockés dans les inodes.

```
Passage en mode debugger
debugfs 1.42.11 (09-Jul-2014)
debugfs:  stat <265019>
Inode: 265019   Type: regular    Mode:  0674   Flags: 0x80000
Generation: 2113284098    Version: 0x00000000:00000001
User:  1000   Group:   100   Size: 44
File ACL: 0    Directory ACL: 0
Links: 1   Blockcount: 8
Fragment:  Address: 0    Number: 0    Size: 0
 ctime: 0x5dd7ecd3:5c9de9f8 -- Fri Nov 22 15:12:35 2019
 atime: 0x5dd7ecd2:7751d9a4 -- Fri Nov 22 15:12:34 2019
 mtime: 0x5dd7ecd1:b54efd60 -- Fri Nov 22 15:12:33 2019
crtime: 0x5dd7ecd1:b54efd60 -- Fri Nov 22 15:12:33 2019
Size of extra inode fields: 32
Extended attributes stored in inode body:
   = "01 00 00 00 01 00 06 00 04 00 04 00 08 00 07 00 e9 03 00 00 10 00 07 00 20 00 04 00 " (
28)
EXTENTS:
(0):1147659
debugfs:  q
```

6 Limite d'attribution d'ACL

Comme précedemment expliqué, les ACL peuvent se voir attribuer une bloc supplémentaire pour leur stockage. Afin de démontrer cela, nous avons élaboré un script qui va attribuer, en boucle, une quantité conséquente d'ACL (508). Voyons comment le système va réagir.

scriptLimite.sh

```
#!/bin/bash

echo -e "\E[32m\n Création du fichier fileLim...\n\E[0m"

touch fileLim

echo -e "\E[32m\n Taille occupé par le fichier vide fileLim...\n\E[0m"

du -h fileLim

echo -e "\E[32m\n Attribution d'ACL en boucle sur le fichier fileLim...\n\E[0m"

for i in {1..508} ; do
    group="group$i"
    if grep -q $group /etc/group ; then
    echo "$group exist"
    else
        sudo groupadd $group
        setfacl -m g:$group:rwx fileLim
    fi
done

echo -e "\E[32m Appuyez enter pour voir le nombre d'ACL attribué au fichier fileLim\E[0m"
read -p " "

getfacl fileLim | grep -v \# | tr -s \\n  | wc -l

echo -e "\E[32m\n Taille occupé par le fichier fileLim après AttributionACL..\n\E[0m"

du -h fileLim

echo -e "\E[32m\n On peut voir que les ACL occupe un bloc physique entier.\n\E[0m"
```

sortie scriptLimite.sh

```
user0@linux-v6fe:~/Bureau/SYSG5 02-12/Script> ./scriptLimiteAcl.sh

 Crγation du fichier fileLim...


 Taille occupγ par le fichier vide fileLim...

0       fileLim

 Attribution d'ACL en boucle sur le fichier fileLim...

setfacl: fileLim: Aucun espace disponible sur le pγriphγrique
setfacl: fileLim: Aucun espace disponible sur le pγriphγrique
setfacl: fileLim: Aucun espace disponible sur le pγriphγrique
setfacl: fileLim: Aucun espace disponible sur le pγriphγrique
setfacl: fileLim: Aucun espace disponible sur le pγriphγrique
 Appuyez enter pour voir le nombre d'ACL attribuγ au fichier fileLim

507

 Taille occupγ par le fichier fileLim aprλs AttributionACL..

4,0K    fileLim

 On peut voir que les ACL occupe un bloc physique entier.
```

Comme nous pouvons le constater, le système ne permet d'attribuer des ACL au fichier fileLim, ce qui démontre le fait qu'il y'a une limite et que justement, seul un bloc est attribué pour le stockage de ces ACL.

7 Conclusion

Nous connaissons dans le cadre de nos études, le système d'autorisations par défaut fournit par les systèmes UNIX et Linux. Celui-ci est pleinement suffisant dans la plupart des utilisations, cependant il ne supervise que 3 catégories d'utilisateur: le propriétaire, le groupe auquel appartient le propriétaire et les "autres membres". Alors que ACL permet de superviser plusieurs utilisateurs et groupes spécifiques. Les ACL permettent donc d'autoriser un utilisateur tiers sans autoriser tout un groupe ou les "autres membres". De plus, les ACL sont accès simple d'utilisation à condition que votre noyau le supporte, dans le cas contraire les droits accordés par les ACL ne seront pas pris en compte.

8 Bibliographie

- https://doc.ubuntu-fr.org/acl
- https://www.geeksforgeeks.org/access-control-listsacl-linux/
- http://sdz.tdct.org/sdz/les-acl-access-control-lists-sous-linux.html
- https://lea-linux.org/documentations/Gestion_des_ACL
- https://access.redhat.com/documentation/fr-fr/red_hat_enterprise_linux/6/html/storage_administrat
setting