

Contrôle d'accès e le POSIX Access Control Lists(ACL)

CS435 - Administration de Système

Dan Pham et Fabrício Nascimento

Octobre 2009

Introduction

Quand l'objectif c'est contrôler l'accès sur les données dans un système de fichiers, il y a plusieurs formes de règlement. Par défaut, les systèmes POSIX (Portable Operating System Interface)[2, 3] ont un mécanisme qui permet de associer chaque entité avec un ensemble de règles, lequel est composé par une séquence d'octet qui exprime le droit du propriétaire, de son groupe et des autres utilisateurs.

Ce mode traditionnel est assez simple et capable d'adresser les problèmes plus fréquents. Par contre, il pose des limitations aux administrateurs de système, lesquels fréquemment doivent employer quelques configurations non évidentes afin d'être capable d'exprimer ces besoins. Par exemple les applications comme le serveur FTP Proftpd[4] ont ces exclusives façons de résoudre ces problèmes de droits pour accéder aux objets du système de fichiers.

À cause de remédier ces limitations présentes les UNIX permettent d'employer les ACL.

Cette article présente une exposition sur les ACL POSIX, ces modes de fonctionnement, ces clés de succès et désavantages. Le texte est fortement basé en l'article de Andreas Gruenbacher[1] dont a été dans l'équipe qui a ajouté le support aux ACL dans le noyau Linux pour les systèmes de fichiers ext2 et ext3, lequel est le système de fichiers plus utilisé dans le monde UNIX.

Le POSIX 1003.1

Traditionnellement les systèmes qui implémentent les patrons POSIX avaient un système simple et puissant de permission, quand même, certains problèmes ont arrivé, et éventuellement les problèmes ont apparu.

Après savoir la nécessité de régler sur le domaine de sécurité et non seulement les ACL, un groupe a été formé pendant la définition de la famille de patrons POSIX 1003.1. Les premiers documents POSIX qui ont été considérés ces questions étaient les documents 1003.1e (*System Application Programming Interface*) et 1003.2c (*Shell and Utilities*), cependant, la première approximation de ce sujet était trop ambitieuse. Le groupe responsable pour la patronisation avait centré ces efforts dans une tâche assez grande de choses, lesquelles comprenant *Access Control Lists* (ACL), *Audit*, *Capability*, *Mandatory Access Control* (MAC), et *Information Labeling*[1].

En Janvier de 1998[1] le financement était fini, par contre, le travail n'était pas près. De toute façon le dix-septième brouillon a été publié quand même[5].

Donc après cette année, les systèmes UNIX appelés "*trusted*" (Trusted Solaris, Trusted Irix, Trusted AIX) ont été développés avec quelques parts de la documentation 17. Ces systèmes ne sont pas complètement compatibles entre eux.

Aujourd'hui la plupart des systèmes UNIX et UNIX-like supportent ACL. Ces implémentations sont usuellement compatibles avec le document 17. Le projet TrustedBSD aussi avait ajouté les ACL sur les systèmes BSD. Les ACL et les MAC FreeBSD-RELEASE ont apparu en 2003.

Les base des ACL sont lancé sur le système traditionnel présent usuellement dans presque tous les système UNIX, alors, avant de préciser sur les ACL on parlerais du modelé traditionnel.

Système de permission Traditionnel

Le modele traditionnel POSIX offre trois group de utilisateur qui sont le propriétaire, le group e les autres. Chaque group a une octet que indique les permission de lecture (**r**ead), écrire (**w**rite) et execution (**x**ecute). La premiere classe fournit les permission pour le utilisateur que rempli le role de propriétaire, ensuite, vien les droits pour le groupe principal du propriétaire enfin les droites pour tous les autres utilisateurs.

Après les trois octets peut venir le *Set User Id*, *Set Group Id* et *Sticky bit*, lesquelles sont utilisé dans certain cases. Il faut faire attention avec le *Sticky Bit*, il permet les utilisateur normale d'exécuter les utilitaire comment le administrateur(*root*), par contre, quelque manque de sécurité peut compromettre le système entiere.

Seulement le *root* peut créer les groupes e changer les association de groupes. Celui-là que aussi peut changer les propriétaire.

Les ACL

En Janvier de 1998[1]

Dans une modèle de de sécurise ACL, si quelque agent faire une requête pour acceder aux données, il faut consulte les ACL pour une entrée que permetre l'operation demandé.

Linux

ACL Implementation in Linux

Conclusion

Références

- [1] Andreas Gruenbacher, *POSIX Access Control Lists on Linux*. [http ://www.suse.de/ agruen/acl/linux-acls/online/](http://www.suse.de/~agruen/acl/linux-acls/online/), 2003.
- [2] IEEE Std 1003.1-2001 (Open Group Technical Standard, Issue 6), Standard for Information Technology–Portable Operating System Interface (POSIX) 2001. ISBN 0-7381-3010-9. [http ://www.ieee.org/](http://www.ieee.org/)
- [3] IEEE 1003.1e and 1003.2c : Draft Standard for Information Technology–Portable Operating System Interface (POSIX)–Part 1 : System Application Program Interface (API) and Part 2 : Shell and Utilities, draft 17 (withdrawn). October 1997. [http ://wt.xpilot.org/publications/posix.1e/](http://wt.xpilot.org/publications/posix.1e/)

Algorithm 1 Verifie se une utilisateur peut ou ne peut pas acceder une objet du système de fichier

```
if the user ID of the process is the owner then
    the owner entry determines access
else if the user ID of the process matches the qualifier in one of the named
user entries then
    this entry determines access
else if one of the group IDs of the process matches the owning group and the
owning group entry contains the requested permissions then
    this entry determines access
else if one of the group IDs of the process matches the qualifier of one of the
named group entries and this entry contains the requested permissions then
    this entry determines access
else if one of the group IDs of the process matches the owning group or any
of the named group entries, but neither the owning group entry nor any of
the matching named group entries contains the requested permissions then
    this determines that access is denied
else
    the other entry determines access.
end if
if the matching entry resulting from this selection is the owner or other entry
and it contains the requested permissions then
    access is granted
else if the matching entry is a named user, owning group, or named group
entry and this entry contains the requested permissions and the mask entry
also contains the requested permissions (or there is no mask entry) then
    access is granted
else
    access is denied.
end if
```

- [4] Mark Lowes : Proftpd : A User's Guide March 31, 2003.
<http://proftpd.linux.co.uk/>
- [5] Winfried Trümper : Summary about Posix.1e. Publicly available copies of POSIX 1003.1e/1003.2c. February 28, 1999.
<http://wt.xpilot.org/publications/posix.1e/>