Network Working Group
Internet Draft
Document: draft-ietf-nfsv4-acl-mapping-00.txt

Marius Aamodt Eriksen
August 2003

Mapping Between NFSv4 and Posix Draft ACLs

Status of this Memo

Abstract

The NFS (Network File System) version 4[rfc3530] (NFSv4) specifies a flavor of Access Control Lists (ACLs) that apply to file system objects. ACLs specify an access level for a number of entities. The NFSv4 ACLs model resembles that of Windows NT. A POSIX draft[posixacl] proposes another, more restrictive ACL model. Many systems implement this proposed standard. Differing in syntax, semantics and extensiveness, it is only feasible to create a correct representation for POSIX ACLs with NFSv4 ACLs. This does not hold for an attempt to represent arbitrary NFSv4 ACLs with POSIX ACLs.

Table of Contents

1. Introduction

The NFS (Network File System) version 4 [rfc3530] (NFSv4) specifies a flavor of Access Control Lists (ACLs) that resembles that of Windows NT's. ACLs are used to specify fine grained control of access to file system objects. An ACL consists of a number of Access Control Entries (ACEs), each specify some level of access for an entity; an entity can be a a user, group or a special entity. The access level is described using an access mask, which is a bitmask where each bit describes a level of access, for example read, write and execute permissions on the file system object. The POSIX Draft Standard 17[posixacl] proposes a simpler, more limited ACL model.

Due to the difference in syntax, semantics and extensiveness, it is only feasible to correctly represent POSIX ACLs using NFSv4 ACLs, and not the other way around. Thus, we provide such a mapping for use in systems that implement NFSv4, already have POSIX Draft Standard ACL support and wish to continue to use this interface with NFSv4 and interoperate with other such systems. A client may also use the mapping for storing, retrieving and interpreting ACLs on an NFSv4 server that supports the storage, retrieval and interpretation of arbitrary NFSv4 ACLs.

2. NFSv4 ACLs

NFSv4 ACLs are rich in nature and expands upon the traditional idea of ACLs. An NFSv4 ACE can be of type ALLOW, DENY, LOG or ALARM; each specifies a different action to take should the ACE match a current request. NFSv4 ACLs also have a rich set of access types that complement the traditional types. These include appending data to the file system object, deleting children of the file system object and deleting the file system object, etc [rfc3530].

NFSv4 ACLs are interpreted in a straightforward manner.

1) Walk through the list of ACEs from the ACL in order

2) If the "who" (entity) field in the ACE does not match that of the requester, the particular ACE is not processed.

3) Process all ACEs until all the bits in the requested access mask have been ALLOWed; that is, the bits have entries in matching ALLOW ACEs that are not flagged ACE4_INHERIT_ONLY_ACE. Once a particular bit has been ALLOWed by an ACE, it is no longer considered in further processing.

4) If a particular access is DENYed (while that bit is still under

consideration), the request is denied.

5) If all bits have been ALLOWed, the access is granted, otherwise the access is denied

NFSv4 ACLs also specify a number of special entities such as OWNER, GROUP and EVERYONE. These refer to the traditional UNIX permissions. Others include DIALUP, BATCH and AUTHENTICATED, which have specialized uses.

Additionally the NFSv4 ACLs specify a number of flags that can be applied to individual ACEs. These include a specification of how an ACE on a directory may be propagated to newly created files or directories inside of said directory.

The granularity of access control provided by NFSv4 ACLs is well beyond that provided by standard UNIX file system permissions.

3. POSIX ACLs

"POSIX ACLs" refer to POSIX 1003.1e/1003.2c Draft Standard 17 [posixacl]. It was meant to specify a POSIX standard for ACLs, but unfortunately never materialized. However, many systems still use it, in forms of its latest and earlier drafts.

POSIX ACLs are simpler than its NFSv4 equivalent. Each ACE an has an entity and the traditional UNIX mode bits that are assigned to the particular entity. The entity may be an arbitrary UID or GID or one

of a few special entities, the most notable of which is the ACL_MASK entity. POSIX ACLs are also interepreted differently than their NFSv4 equivalents.

POSIX ACLs are interpreted as follows:

1) Process the ACL_USER_OBJ (equivalent to UNIX file owner) ACE first; if the UID of the requester does not match that of the ACL_USER_OBJ, then the ACE is ignored. Otherwise, the request is granted if and only if the request access mask is allowed by the access mask of the ACE.

2) Process all of the ACL_USER ACEs; the entity of these ACEs specifies a user on the system. If the UID of the requester does not match that of the ACE, then the ACE is ignored. Otherwise, the request is granted if and only if the request access mask is allowed by the access mask of the ACE.

3) Process the ACL_GROUP_OBJ ACE and all of the ACL_GROUP ACEs; the entity of each ACE specify a group on the system. If none of the GIDs of the requester match the entity of the current ACE, the particular ACE is ignored. For any matching ACE, if the the requester's access mask is allowed by the ACEs access mask, then access is permitted. If there are matching ACEs, but none allow access, then access is denied.

4) If the requester's access mask is allowed by the ACL_OTHER ACE, then grant access.

5) Deny access.

Steps (2) and (3) have an additional restriction; in addition to checking whether the requested access mask is allowed by the access mask in the ACE, the requested bits also have to be in the access mask of the special ACE with the ACL_MASK entity. This allows file owners to specify a maximum level of access allowed by any other user or group that has any access to the file system object.

In addition to a regular POSIX ACL, a directory in the file system may also have associated with it a default ACL. A default ACL governs the ACL a file system object will be assigned initially when it is created as a child of the particular directory.

4. Mapping Posix ACLs to NFSv4 ACLs

Given the difference in both extensiveness and interpretation of POSIX and NFSv4 ACLs, conversion of arbitrary NFSv4 ACLs to POSIX ACLs is infeasible. However, POSIX ACLs are a subset of NFSv4 ACLs. Any POSIX ACL can be emulated with an NFSv4 ACL.

The difference in the format of POSIX ACEs and NFSv4 ACEs can be compensated for by a direct mapping.

The ACE entity is translated as follows. The non-special entity in form of UIDs and GIDs is translated to equivalent strings (a system dependent process, typically done by lookups to /etc/passwd in UNIX). The POSIX ACL_USER_OBJ entity is translated to the "OWNER" NFSv4 entity. Similary, the POSIX ACL_GROUP_OBJ is translated to the "GROUP" NFSv4 entity. The ACL_OTHER entity is translated to the "EVERYONE" NFSv4 entity.

The access mask is translated as follows. The read bit of the POSIX

access mask is translated to the "ACE4_GENERIC_READ" NFSv4 access mask field. The write bit of the POSIX access mask is translated to "ACE4_GENERIC_WRITE" NFSv4 access mask field. the execute bit of the POSIX access mask is translated into the "ACE4_GENERIC_EXECUTE" NFSv4 access mask field. Defined in [rfc3530], "ACE4_GENERIC_READ" is a logical OR of "ACE4_SYNCHRONIZE," "ACE4_READ_ACL," "ACE4_READ_ATTRIBUTES," "ACE4_READ_DATA" and "ACE4_LIST_DIRECTORY." "ACE4_GENERIC_WRITE" is a logical OR of "ACE4_SYNCHRONIZE," "ACE4_READ_ACL," "ACE4_WRITE_ACL," "ACE4_WRITE_ATTRIBUTES," "ACE4_WRITE_DATA," "ACE4_ADD_FILE," "ACE4_APPEND_DATA" and "ACE4_ADD_SUBDIRECTORY." Finally, "ACE4_GENERIC_EXECUTE" is a logical OR of "ACE4_SYNCHRONIZE," "ACE4_READ_ACL," "ACE4_READ_ATTRIBUTES" and "ACE4_EXECUTE." These were chosen to represent the true meaning of the UNIX mode which are used by POSIX ACLs.

The ACE flag field also has a simple translation. If the file system object is a directory, and the POSIX ACE belongs to a default ACL, the "ACE4_INHERIT_ONLY_ACE" flag is set in the NFSv4 ACE. If the entity in the POSIX ACE refers to a group, the "ACE4_IDENTIFIER_GROUP" flag is set in the NFSv4 ACE.

Completing the mapping reduces to being able to emulate an ACL_MASK and compensate for the difference in interpretation between to two ACL implementations.

The difference in interpretation of the two ACL types call for a translation scheme. The scheme follows:

Every user ACE in the POSIX ACL maps into 2 NFSv4 ACEs; one ALLOW ACE which is translated as specified by the above scheme, then a complementing DENY ACE which is also translated as specified by the above scheme, with the exception that the access mask is inverted. The ACL_USER_OBJ ACE is placed first in the list.

Every group ACE in the POSIX ACL produces a similar pair, but instead of being in sequence, all of the ALLOW ACEs are placed first, followed by all the DENY ACEs. The ACL_GROUP_OBJ ACE is placed first in the list.

Lastly, the POSIX ACL_OTHER ACE translates directly into one NFSv4 ACE at the end of the group ACEs. This is an allow ACE which is translated as specified by the above scheme.

This translation strategy allows us to emulate POSIX ACL interpretation in an NFSv4 ACL.

To handle the special POSIX entity ACL_MASK, we slightly modify the

above translation:

With the exception of the "OWNER," "GROUP," and "EVERYONE" ACEs, another ACE is prepended to every ACE. The prepended ACE is a DENY ACE with the same entity as the following ALLOW ACE, but with a permission mask set to the complement of the POSIX ACL_MASK.

This method allows us to preserve the real permission bits of each

ACE should the ACL_MASK be changed.

The fact that POSIX ACLs use separate ACLs for determining access to the file system object and determining inheritance of the ACL needs compensation in the translation scheme. Whenever the server receives a request for an ACL, if the file system object in question is a directory, the server appends the default ACL to the access ACL. It is then up to the client to separate the two ACLs and translate them individually. Similarly, when the client wishes to set an ACL, it either sends the access and default ACLs individually in separate requests, or concatenates them. Again the server should separate default and access ACLs, translating and setting them individually.

The reverse mapping follows from the forward mapping described here. The forward mapping obeys a very strict template, and the implementer must ensure that when performing the reverse mapping, the ACL strictly adheres to this template.

5. Security Considerations

Since this draft deals with the mapping of Access Control Lists, it is deeply involved with security. The body of this document needs to address the issue of mapping ACLs in a way as to not disobey the intent of or mislead the user.

It is therefore important that ACLs that do not match the above scheme are explicitly rejected. Also, neither optimistic nor pessimistic translation between POSIX and NFSv4 ACLs should be carried out. This can potentially lead to unintended granting or revoking of priveliges.

6. Bibliography

[rfc3530]
Shepler, S. et. al., "NFS version 4 Protocol",
http://www.ietf.org/rfc/rfc3530.txt, April 2003.

[posixacl]
IEEE, "IEEE Draft P1003.1e", October 1997 (last draft).

http://wt.xpilot.org/publications/posix.1e/download.html

7. Author's Address

Address comments related to this memorandum to:

marius@umich.edu

Marius Aamodt Eriksen
University of Michigan / CITI
535 West William
Ann Arbor, Michigan

E-mail: marius@umich.edu

8. Copyright

HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.