Network Working Group
Internet Draft
Document: draft-ietf-nfsv4-acl-mapping-01.txt

Marius Aamodt Eriksen
February 2004

Mapping Between NFSv4 and Posix Draft ACLs

SSttaattuuss ooff tthhiiss MMeemmoo

AAbbssttrraacctt

The NFS (Network File System) version 4[rfc3010bis] (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.

TTaabbllee ooff CCoonnttteennttss

11.. IInnttrroodduucctiioonn

The NFS (Network File System) version 4 [rfc3010bis] (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 specifying 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.

22.. NNFFSSvv44 AACCLLss

NFSv4 ACLs are rich in nature and expand 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 complements the traditional types. These include appending data to the file object, deleting children of the file object, deleting the file object, etc [rfc3010bis].

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; 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, or else behavior is undefined.

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

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

It is clear that the granularity of access control that NFSv4 ACLs specify is well beyond the standard UNIX capability of expressing file system object permissions.

33.. PPOOSSIIXX AACCLLss

POSIX ACLs refer to POSIX 1003.1e/1003.2c Draft Standard 17 [posix-acl], which was meant to specify a POSIX standard for ACLs, but unfortunately never materialized. However, many systems still use it, both in the form of it's latest draft as well as earlier drafts.

POSIX ACLs are simpler than their NFSv4 equivalents. 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, if the requester's access mask is allowed by the access mask of the ACE, the request is granted, else the request is denied.

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

3) Process the ACL_GROUP_OBJ ACE and all of the ACL_GROUP ACEs; the entity of these ACEs specify a group on the system. If none of the GIDs of the requester match 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 criteria; 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. This default ACL governs the ACL a file system object will be assigned initially when it is created as a child of the particular directory.

44.. MMaappppiinngg PPoossiixx AACCLLss ttoo NNFFSSvv44 AACCLLss

Given the difference in both extensiveness and interpretation of POSIX and NFSv4 ACLs, any conversion between the two is difficult. However, POSIX ACLs are a subset of NFSv4 ACLs. Any POSIX ACL can be emulated with an NFSv4 ACL using the following mapping.

The ACE entities are translated as follows. The non-special entities 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 ACE access mask is translated as follows. The read bit of the POSIX access mask is translated to the logical OR of the ACE4_READ_DATA and ACE4_READ_NAMED_ATTRS NFSv4 access mask fields. The write bit of the POSIX access mask is translated to the logical OR of the ACE4_WRITE_DATA, ACE4_WRITE_NAMED_ATTRIBUTES and ACE4_APPEND_DATA NFSv4 access mask fields. The execute bit of the POSIX access mask is translated into the ACE4_EXECUTE and ACE4_READ_DATA NFSv4 acess mask fields. Note that NFSv4 defines ACE4_READ_DATA, ACE4_WRITE_DATA, and ACE4_APPEND_DATA to be equal to ACE4_LIST_DIRECTORY, ACE4_ADD_FILE, and ACE4_ADD_SUBDIRECTORY, respectively, so this translation makes sense for directories as well. However, on directories the ACE4_DELETE_CHILD field must be included in the translation of the POSIX write bit.

In addition to the above, the OWNER entity must always be given ACE4_WRITE_ACL and ACE4_WRITE_ATTRIBUTES, and all entities must be given ACE4_READ_ACL and ACE4_READ_ATTRIBUTES.

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.

The POSIX ACL_USER_OBJ ACE is also always given the permission bits "ACE4_READ_ACL" and "ACE4_WRITE_ACL."

Completing the mapping reduces to being able to emulate an ACL_MASK and compensate for the difference in interpretation between 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. Note that the ACL_USER_OBJ ACE is placed first in this list.

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

Lastly, the POSIX ACL_OTHER ACE is translated into a pair of ACEs as in the user ACE case.

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" and "EVERYONE" ACEs, another ACE is prepended to the 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.

55.. SSeeccuurriittyy CCoonnssiiddeerraattiioonnss

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.

66.. BBiibblliiooggrraapphhyy

[rfc3010bis]
Shepler, S. et. al., "NFS version 4 Protocol", draft-ietf-nfsv4-rfc3010bis-05.txt, April 2003.

http://www.ietf.org/internet-drafts/draft-ietf-nfsv4-rfc3010bis-05.txt

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

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

77.. AAcckknnoowwlleeddggmmeennttss

The author would like to thank and acknowledge Bruce Fields for his careful scrutiny and excellent comments and suggestions.

88.. AAuutthhoorr''ss AAddddrreessss

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

99.. CCooppyyrriigghhtt