Mapping Between NFSv4 and POSIX Draft ACLs

Marius A Eriksen
CITI, University of Michigan

marius@citi.umich.edu

Problem

- NFSv4 specifies a rich flavor of ACLs
- Many UNIX(-like) OSes implement a simpler flavor based on POSIX draft standards
- We would like to use the existing file system interfaces and semantics for handling ACLs with NFSv4
- Need a mapping scheme between POSIX draft and NFSv4 ACLs
- draft-eriksen-nfsv4-acl-02.txt

POSIX ACLs

- Refer to POSIX 1003.1e/1003.2c Draft Standard 17
- Never materialized into an actual POSIX standard
  - "Widely" used anyhow in form of various drafts
- Much simpler than NFSv4 ACLs:
  - A number of ACEs, each with an entity (UID/GID) and traditional UNIX mode bits
- Big difference in interpretation
  - NFSv4: "cumulative"
  - POSIX: "precise" (in some ways)

POSIX ACLs (2)

- ACL_MASK
  - A special ACE that specifies the maximum level of access granted for all ACEs except for the owner and "other" ACE

- A directory may have two ACLs associated to it
  - Access ACL (determines access)
  - Default ACL (determines ACL that’s placed on new file system objects

POSIX ACL Interpretation

- Each USER ACE is processed first
  - If the requestor's UID matches that of the ACE, the request is granted iff the request mask is allowed by the access mask of the ACE
- Every GROUP ACE is then processed
  - If the the requester's access mask is allowed by a matching ACEs, access is permitted. If there are matching ACEs, but none allow access, then access is denied
- The OTHER ACE is then processed
  - Grant access iff access mask is allowed by the ACL_OTHER ACE

NFSv4 ACL Interpretation

- Walk through the list of ACEs in order
  - Consider ACE iff "who" matches requestor and is not flagged ACE4_INHERIT_ONLY_ACE
  - Continue until all bits in the requestors access mask have been ALLOWed
  - If a particular access bit is DENYed while still under consideration, deny the request
- Grant access iff all requested bits have been ALLOWed

Mapping POSIX to NFSv4

- Translating individual ACEs is straightforward
- Identities
  - POSIX owner -> NFSv4 "OWNER@"
  - POSIX group -> NFSv4 "GROUP@" (with group flag)
  - POSIX other -> NFSv4 "EVERYONE@"
  - POSIX UID -> NFSv4 "username" (translated)
  - POSIX GID -> NFSv4 "groupname" (translated, with group flag)
- Permissions
  - POSIX read -> ACE4_GENERIC_READ
  - POSIX write -> ACE4_GENERIC_WRITE
  - POSIX execute -> ACE4_GENERIC_EXECUTE

Mapping Semantics POSIX to NFSv4

- User ACEs
  - Map into 2 NFSv4 ACEs
    - ALLOW ACE, translated from the POSIX ACE
    - DENY ACE, translated from POSIX ACE, with inverted access mask
- Group ACEs
  - Map into 2 NFSv4 ACEs as in the user ACE case
  - All the ALLOW ACEs are placed first, followed by all the DENY aces
- Other ACE
  - Map into a pair at the end of the ACL

Mapping ACL_MASK POSIX to NFSv4

- Prepend an ACE to every ACE except for the OWNER and OTHER ACE
  - DENY ACE with the same entity as the following ALLOW ACE
  - access mask is the complement of the POSIX ACL_MASK

Example ACL translation

- Given an ACL

```
u::rw-          (ACL_USER_OBJ entry)
u:marius:r--    (ACL_USER entry)
u:foobar:-w-    (ACL_USER entry)
g::r-x          (ACL_GROUP_OBJ entry)
g:users:r--     (ACL_GROUP entry)
g:wheel:-w-     (ACL_GROUP entry)
m::rwx          (ACL_MASK entry)
o::r--          (ACL_OTHER entry)
```

ACE translation example

u::rw- (ACL_USER_OBJ entry)

Translated to:

```
{type=ALLOW, flag=0,
 mask=GENERIC_READ|GENERIC_WRITE,
 who="OWNER@" }
{type=DENY,  flag=0,
 mask=~(GENERIC_READ|GENERIC_WRITE),
 who="OWNER@" }
```

The group entries

```
g::r-x          (ACL_GROUP_OBJ entry)
g:users:r--     (ACL_GROUP entry)
g:wheel:-w-     (ACL_GROUP entry)
```

Translated to:

```
{type=ALLOW, flag=ACE4_IDENTIFIER_GROUP,
 mask=GENERIC_READ|GENERIC_EXECUTE,
 who="GROUP@" }
{type=ALLOW, flag=ACE4_IDENTIFIER_GROUP,
 mask=GENERIC_READ,
 who="users@foo.citi.umich.edu" }
{type=ALLOW, flag=ACE4_IDENTIFIER_GROUP,
 mask=GENERIC_WRITE,
 who="wheel@foo.citi.umich.edu" }
{type=DENY, flag=ACE4_IDENTIFIER_GROUP,
 mask=~(GENERIC_READ|GENERIC_EXECUTE),
 who="GROUP@" }
{type=DENY, flag=ACE4_IDENTIFIER_GROUP,
 mask=~(GENERIC_READ),
 who="users@foo.citi.umich.edu" }
{type=DENY, flag=ACE4_IDENTIFIER_GROUP,
 mask=~(GENERIC_WRITE),
 who="wheel@foo.citi.umich.edu" }
```

Handling ACL_MASK

u:marius:rw- (ACL_USER entry)
m::rx (ACL_MASK entry)

Translated to:

```
{type=DENY, flag=0,
 mask=GENERIC_WRITE,
 who="marius@citi.umich.edu" }
{type=ALLOW, flag=0,
 mask=GENERIC_READ|GENERIC_WRITE,
 who="marius@citi.umich.edu" }
{type=DENY,  flag=0,
 mask=~(GENERIC_READ|GENERIC_WRITE),
 who="marius@citi.umich.edu" }
```

Default ACLs

- For directories, the server simply concatenates the default ACL with the appropriate INHERIT flags set.
- Client and server separate the two ACLs based on INHERIT flags before interpreting them.

The End

Questions? Comments?