NFSv4 ACLs: Present and Future

Lisa Week
lisa.week@sun.com

Sam Falkner
sam.falkner@sun.com

Sun Microsystems, Inc.

Agenda

- ACL Basics
- Current ACL Implementations
- Internet Draft and Future ACL Implementations
- Questions?

ACL Basics

- ACLs: Access Control Lists
- Used for specifying fine-grained permissions for specific users and groups
- NFSv4 specifies a standard for ACLs

NFSv4 ACLs

- Based on Windows NT ACL model
  - Many users understand the NT ACL model
  - Eases NFSv4 server implementation on Windows
  - Eases implementation of dual protocol servers; for example, NFSv4/CIFS servers

ACL Entries

- ACLs are made of zero or more entries called “ACEs”
  - type: ALLOW, DENY, AUDIT, ALARM
  - who: who does the entry pertain to
  - flags: Inheritance, etc.
  - masks: Which permissions are covered by this ACE

ACE “type” field

- ALLOW: grants permissions
- DENY: denies permissions
- AUDIT: audits accesses
- ALARM: alarms accesses

ACE “who” field

- Can specify an individual user; example: “samf@sun.com”
- Can specify an individual group; example: “staff@sun.com”
- Can specify abstract values; example: “OWNER@” matches a file’s current owner
- “EVERYONE@” indicates the ACE applies to literally every entity

ACE “mask” field

- Specifies a set of permissions to be acted upon, based on the type of the ACE
- Some masks control file access
  - Example: ACE4_READ_DATA
- Some masks control file management
  - Example: ACE4_WRITE_OWNER

ACE “flags” field

FILE_INHERIT_ACE	ACE inherited to newly created files
DIRECTORY_INHERIT_ACE	ACE inherited to newly created directories
INHERIT_ONLY_ACE	ACE does not apply to current file or directory, but will be inherited based on two preceding flags
NO_PROPAGATE_INHERIT_ACE	ACE is inherited according to first two flags, but first four flags are cleared on new ACE
SUCCESSFUL_ACCESS_ACE_FLAG	AUDIT or ALARM applies to successful access
FAILED_ACCESS_ACE_FLAG	AUDIT or ALARM applies to failed access
IDENTIFIER_GROUP	“who” field indicates a group, as opposed to a user

ACL Enforcement

- Order dependent: 1st ACE matching “who” is used
- Unspecified permissions are usually denied

Common ACL Problem

- EVERYONE@::ALLOW:READ_DATA
- evil@bad.com::DENY:READ_DATA

Current ACL Implementations

- Some vendors have native NFSv4 ACL implementations
  - Can fully enforce all NFSv4 ACL semantics
- Some vendors have other native ACL implementations, and translate
  - Can only enforce a subset of NFSv4 ACL semantics

POSIX-draft ACLs

- Another ACL model, never became a POSIX standard
- ACL composed of a list of ACL entries that both allow and deny permission
- Permissions do not go beyond read, write, and execute

Translation between POSIX-draft and NFSv4

- Specified in internet draft by Bruce Fields and Marius Eriksen
- Possible to translate any POSIX-draft into NFSv4
- Not every NFSv4 ACL may be translated into POSIX-draft
  - Client using NFSv4 semantics may not be able to set any arbitrary ACL against a server using POSIX-draft ACLs

Areas for Improvement

- Interactions between mode and ACL
- Ambiguity in interpretation of the access mask bits
- Ambiguity in interpretation of EVERYONE@
- These areas are covered in Internet Draft

Interactions Between Mode and ACL

- What happens to an existing ACL upon chmod?
  - Most current implementations discard the ACL
  - Internet draft proposes augmenting the ACL so as to not conflict with the mode
- What happens to inheritable ACEs upon create with a mode?

Augmenting ACL based on new mode

- Inheritable ACEs are split into two: one INHERIT_ONLY, one not inheritable
- Specific users and groups are limited to the group permissions of the new mode
- Six ACEs, representing the new mode, are appended to the ACL

Interactions Between Mode and ACL

- What happens to inheritable ACEs upon create with a mode?
  - New ACL is created from inheritable ACEs
  - New ACL is augmented by the mode, as in the previous slide

What happens to the mode upon setting a new ACL

- New mode is computed based upon the ACL
- Unspecified permissions are assumed to be denied
- It is impossible for an algorithm to always give the correct mode in all cases

Ambiguities in Mask Bits

- WRITE_ATTRIBUTES
- READ_ATTRIBUTES
- WRITE_OWNER
- WRITE_NAMED_ATTRS
- READ_NAMED_ATTRS
- WRITE_ACL
- DELETE / DELETE_CHILD

The Meaning of EVERYONE@

- EVERYONE@ matches literally everyone
- NOT equivalent to UNIX “other” entity

Questions?