Fun with NFSv4 ACLs

Prepared for Capital One

Chris Keller (ckeller@redhat.com)
Solutions Architect
May 10th, 2013

Agenda

What are NFSv4 ACLs?

NFSv4 Permissions Explained

Using nfs4_{get,set}facl

NetApp Considerations

Issues w/ chmod

Implementing NFSv4 ACLs

Removing NFSv4 ACLs

Questions?

What are NFSv4 ACLs?

- Part of RFC 3530 (NFS version 4 protocol)
- More fine grained access control when compared to previous file permission models (e.g. POSIX)
- Similar to ACLs use by CIFS
- NOT POSIX ACLs
  - POSIX ACLs are managed through the use of {get,set}facl commands in Linux
  - POSIX ACLs are saved in extended attributes on file system, supported by ext3, ext4, xfs, btrfs, etc.

NFSv4 Permissions Explained

- NFSv4 ACL is a composition of ACEs (Access Control Entries)
- ACE Format (acl_spec)
    - {TYPE}:{FLAGS}:{PRINCIPAL}:{PERMISSIONS}
- ACLs are 'default deny', meaning if a permission is not explicitly granted by an allow ACE, it is denied.

```
{TYPE}:{FLAGS}:{PRINCIPAL}:{PERMISSIONS}
```

```
user@redhat.com
group@redhat.com
OWNER@
GROUP@
EVERYONE@
```

```
A - allow
D - deny
U - audit
L - alarm
```

```
g - group
d - directory inherit
f - file inherit
n - no propagate inherit
i - inherit only
(S - successful access)
(F - failed access)
```

```
r - read data (files)
w - write data (files)
a - append data (files)
x - execute (files) / change directory (directories)
d - delete the file/directory
D - delete child
t - read attributes
T - write attributes
n - read named attributes
N - write named attributes
c - read ACL
C - write ACL
o - write owner
Y - synchronize IO
```

ACE Types

- Allow – allow {PRINCIPAL} to perform actions requiring {PERMISSIONS}
- Deny – prevent {PRINCIPAL} from performing actions requiring {PERMISSIONS}
- Audit – log attempted access by {PRINCIPAL} requiring {PERMISSIONS}
- Alarm – generate a system alarm when {PRINCIPAL} attempts to performs actions requiring {PERMISSIONS}

ACE Flags

- g - group - Indicates that a principal represents a group instead of a user.
- d - directory inherit - Can be placed on a directory and indicates that this ACE should be added to each new directory created.
- f - file inherit - Can be placed on a directory and indicates that this ACE should be added to each new non-directory file created.
- n - no propagate inherit - Can be placed on a directory. Normally when a new directory is created and an ACE exists on the parent directory which is marked 'directory inherit', two ACEs are placed on the new directory. One for the directory itself and one which is an inheritable ACE for newly created directories. This flag tells the server to not place an ACE on the newly created directory which is inheritable by sub-directories of the created directory.
- i - inherit only - Can be placed on a directory but does not apply to the directory, only to newly created files/directories as specified by the {d,f} flags.

ACE Principals

- A particular user or group in the format:
  - {user,group}@domain
- USER@ - File Owner
- GROUP@ - Group Owner
- EVERYONE@ - “The World”
  - The behavior of the EVERYONE@ principal is different than the “other” entity in traditional UNIX/Linux permissions.
  - For example, setting the permissions of a file to 004 in UNIX/Linux would deny access to the owner/group.
  - With NFSv4 ACLs, EVERYONE@ includes the user and group associated with the file or directory.

ACE Permissions

- r – read data (files) / list directory (directories)
- w – write data (files) / create file (directories)
- a – append data (files) / create subdirectory (directories)
- x – execute (files) / change directory (directories)
- d – delete the file/directory
- D – delete child (directories only)
- t – read attributes of file/directory
- T – write attributes of file/directory

ACE Permissions Continued

- n – read named attributes
- N – write named attributes
- c – read file/directory ACL
- C – write file/directory ACL
- o – change ownership of file/directory
- y – allow clients to use synchronous I/O w/ the server

Using nfs4_getfacl

- Important: Output of nfs4_getfacl does not imply a security descriptor is present on file/directory. If a security descriptor is not present, nfs4_getfacl will return equivalent acl_spec entries based on applicable POSIX mode bits.
- nfs4_getfacl example

```
 [ckeller@toast ~]$ nfs4_getfacl /mnt/nfs4
A::OWNER@:rwaxtTnNcCy
D::OWNER@:
A:g:GROUP@:rxtncy
D:g:GROUP@:waTC
A::EVERYONE@:tcC
D::EVERYONE@:rwaxTy
```

Using nfs4_setfacl

- -a acl_spec - add acl_spec to ACL
- -A file – bulk add ACLs from file
- -x acl_spec – remove acl_spec from ACL
- -X file – bulk remove ACLs from file
- -s acl_spec – set file's ACL to acl_spec
- -S file – set file's ACL to acl_spec in file
- -e – edit files ACL in editor (typically vi)
- -m from_ace to_ace – modify file's ACL in place by replacing from_ace with to_ace

nfs4_setfacl – Additional Options

- -R – recursively apply action to directory's files and subdirectories
- -L – in conjunction w/ -R, follow all symbolic links
- -P – in conjunction w/ -R, skip all symbolic links
- --test – display results of command, but do not save changes
- Specifying a '-' after the -{A, X, S} options will read from stdin instead of file, useful option for scripting

nfs4_setfacl - Examples

- Add ACE to directory granting 'ckeller@redhat.com' read and execute access to a file.
  - `nfs4_setfacl -a A::ckeller@redhat.com:rxtncy foo`
- Remove the ACE created above
  - `nfs4_setfacl -x A::ckeller@redhat.com:rxtncy foo`
- Set the ACL of bar to ACL of foo
  - `nfs4_getfacl bar | nfs4_setfacl -S - foo`

NetApp Considerations

- fsecurity – Useful debugging command
  -
```
netapp1*> fsecurity show /vol/nfs4_acl_test_two
[/vol/nfs4_acl_test_two - Directory (inum 64)]
  Security style: Unix
  Effective style: Unix

  DOS attributes: 0x0010 (----D---)

  Unix security:
    uid: 1128100509 (ckeller)
    gid: 1128100509 (ckeller)
    mode: 0750 (rwxr-x---)

  NFSv4 security descriptor:
    DACL:
      Allow - OWNER@ - 0x001601bf
      Deny  - OWNER@ - 0x00000000
      Allow - GROUP@ - 0x001200a9 (Read and Execute)
      Deny  - GROUP@ - 0x00040106
      Allow - EVERYONE@ - 0x00060080
      Deny  - EVERYONE@ - 0x00100127
```

NetApp - ACL Processing During File Creation

- If the request includes an ACL, that ACL is used.
- If the request includes only standard UNIX file access permissions, but the parent directory has an ACL, the ACEs in the parent directory's ACL are inherited by the new file or directory as long as the ACEs have been tagged with the appropriate inheritance flags.
  - Note: A parent ACL is inherited even if nfs.v4.acl.enable is set to off.
- If the request includes only standard UNIX file access permissions, and the parent directory does not have an ACL, the client file mode is used to set standard UNIX file access permissions.
- *If the request includes only standard UNIX file access permissions, and the parent directory has a non-inheritable ACL, a default ACL based on the mode bits passed into the request is set on the new object.*

Issues With chmod

- Various vendors (including NetApp) take chmod to mean setting of a six member ACL
  - Throws away ACEs outside of default principals (i.e. OWNER@,GROUP@,EVERYONE@)
  - Mitigated through nfs.v4.acl_preserve option on filer

Implementing NFSv4 ACLs

- Ensure inheritance bits are properly set (preferably at the root of the volume), otherwise it becomes incredibly difficult to navigate/enforce permissions
  - Also ensures each file contains a security descriptor
- Simplify file-system hierarchy
- Limit permissions in EVERYONE@ principal
- Limit the use of DENY ACEs (remember, ACLs are default-deny)
- Use nfs4_setfacl instead of chmod (nfs4_setfacl is script friendly)

Removing ACLs

- Easy to remove all ACEs outside of default principal
  - `nfs4_getfacl foo | egrep "OWNER@|GROUP@|EVERYONE@" | nfs4_setfacl -S – foo`
- However, security descriptor still present on filer
  - May not be necessary to remove security descriptor if all permissions will be reapplied w/ nfs4_setfacl + inheritance
- Use of chmod w/ nfs.v4.acl_preserve off
  - Still does not guarantee removal of security descriptor

Questions?