dCache.org

NFSv4.1, ACL and Co.

Tigran Mkrtchyan
For dCache Team

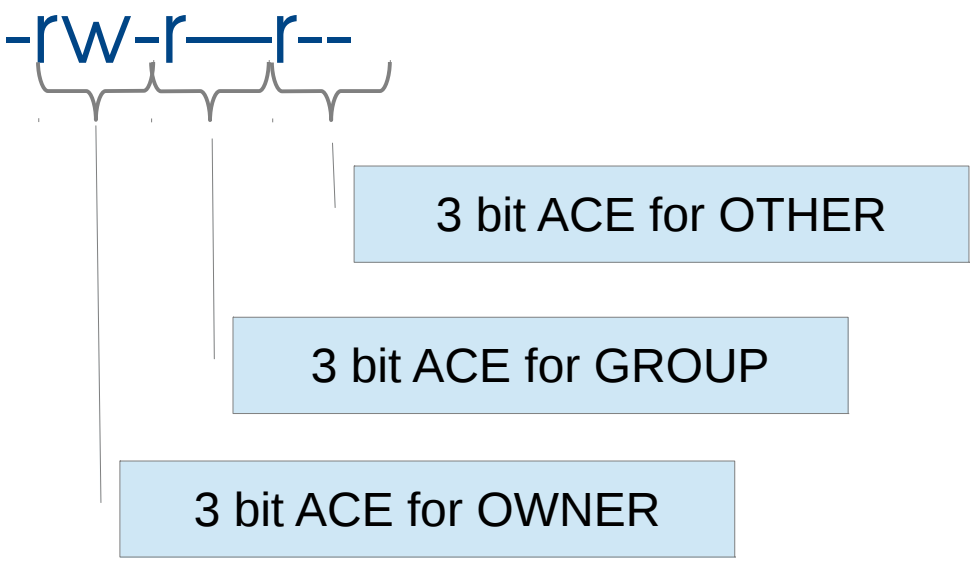
Fermilab NDGF Nordic DataGrid Facility DESY HELMHOLTZ ASSOCIATION LSDMA

ACL basics (for file system)

- ACLs is a list of Access Control Entries attached to a file or a directory
- ACEs grant or deny a principal an action on a file or a directory

ACL basics (for file system)

Traditional UNIX mode:

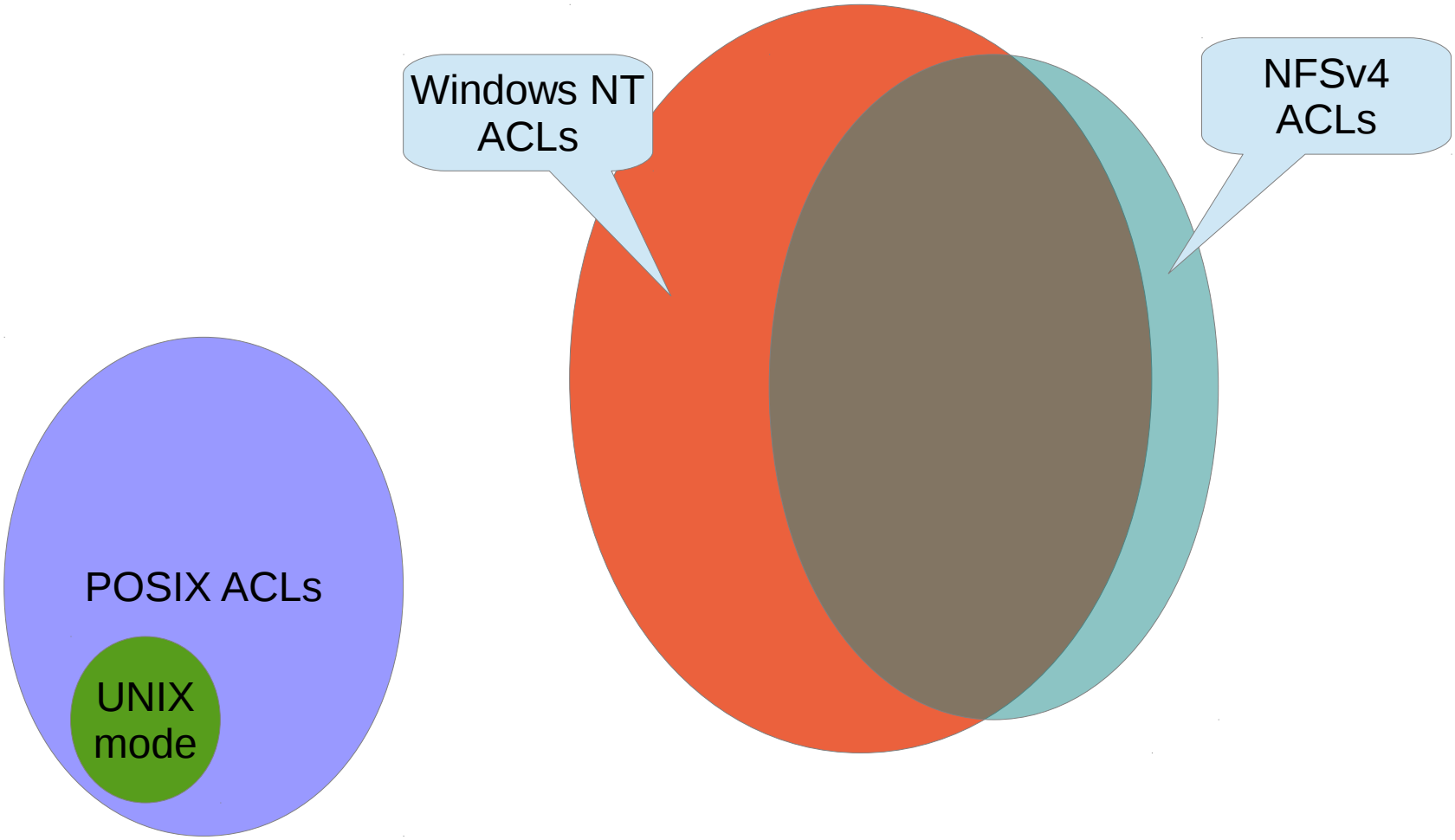


R	Read Access
W	Write Access
X	Execute/Lookup Access

When we need 'Other' ACLs?

- Two or more user principals should have the explicit permissions
- Two or more group principals should have explicit permissions
- Explicit DENY to some set of user/group principals.

ACL models



NFSv4 ACLs

- Defined in NFSv4 standard
  - Defined in rfc3010, rfc3530
  - Modified, clarified rfc5661 (v4.1)
- Users & Groups identified by UTF-8 strings
  - "user@domain" and "goup@domain"
  - Client and server responsible to map those to local representations

NFSv4 ACLs (2)

- 14 access mask bits
  - Binary values identical to Windows
  - Name and semantics similar to Windows
- Unlimited number or principals
- 3 useful special principals
  - OWNER@, GROUP@, EVERYONE@
  - Nearly identical to Windows

NFSv4 ACLs (3)

- More complex than UNIX mode
- Deny-type ACE
- Order of ACEs is significant
- EVERYONE@ != UNIX other
  - Similar enough to cause confusions
- Must retain UNIX mode compatibility
  - Chmod adjusts ACLs
  - Set ACL adjusts UNIX mode

NFSv4.1 ACE masks

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 - delete the file/directory.

D delete-child - remove a file or subdirectory from within the given directory

t read-attributes - read the attributes of the file/directory.

T write-attributes - write the attributes of the file/directory.

n read-named-attributes - read the named attributes of the file/directory.

N write-named-attributes - write the named attributes of the file/directory.

c read-ACL - read the file/directory NFSv4 ACL.

C write-ACL - write the file/directory NFSv4 ACL.

o write-owner - change ownership of the file/directory.

y synchronize - allow clients to use synchronous I/O with the server.

NFSv4 ACLs in dCache

- Pushed by WLCG as MUST
  - 'PRODUCTION' users should be able to delete any file
  - Never used officially
- Real use case with Photon Science @ DESY
  - No real user groups
  - Any data taking results can be shared with different set of people
- Supported by all protocols
  - Differences in enable/disable

Enable in dCache NFS door

- Controlled by exports file
- Per client option
- Can be enabled/disabled on running system
- Set/GetACL available always

Enable in dCache NFS

```
# /etc/dcache/exports

/data trusty(rw,noacl) shared(rw,acl) weak(ro)
```

Enable in dCache NFS

- Enable/disable with 'acl'/'noacl'
  - No option == noacl (old behavior)
- 'exports reload' re-reads exports file
- No re-mount required
  - Takes effect in the fly

Enable in dCache NFS

```
# /etc/dcache/exports
/data h*(ro) h1*(rw) host1(rw) 1.1.1.1(rw) 1.1.1.0/24(ro)
/data/read-only host1(ro)
```

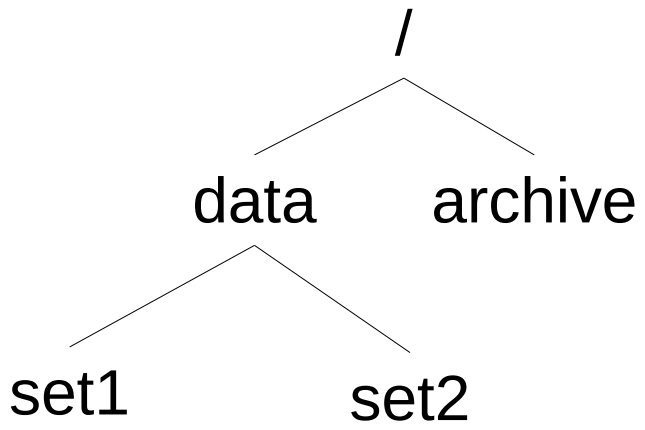
- *Internally sorted as more precise entry first*
  - 1.1.1.1, 1.1.1.0/24, host1, h1*, h*
  - First match wins
  - Shortest path wins => /data host1(rw)

PseudoFS

/data/set1

/data/set2

/archive



Clients can always mount the '/',
but will see only exports directories.

Pitfalls – per client option

- Not all client see the same access rights
  - Users access depends on the client node
- RO exports will RJECT updates even if ACL allows
- Independent of 'aclEnabled=true/false'
- Not all doors see the same access rights
- Other protocols do not sync ACL and UNIX mode
  - Update of UNIX mode doesn't adjust ACLs
  - Update of ACLs doesn't adjust UNIX mode

Enable on a Client

- RHEL6 and Clones (SL6, CentOS6)
  - Other OSes supports as well – check the docs
  - Install nfs4-acl-tools
- 'nfs4_getfacl' and 'nfs4_setfacl'
  - Hopefully, one day will be merged into setfacl and getfacl

Examples

$ ls -ld file.X

-rwxrwxr-x 2 tigran nobody 512 May 19 12:47 file.X

$ nfs4_getfacl file.X

A::OWNER@:rwaDxtTcC

A::GROUP@:rwaDxtc

A::EVERYONE@:rxtc

$

No ACLs, unix to ACL translation

Examples

$ chmod 000 file.X

$ ls -ld file.X

---------- 2 tigran nobody 512 May 19 12:47 file.X

$ nfs4_getfacl file.X

A::OWNER@:tTcC

A::GROUP@:t

A::EVERYONE@:t

$

No ACLs, unix to ACL translation

Examples

```
$ nfs4_setfacl -s 'A::tigran@desy.afs:r' file.X
$ ls -ld file.X
-rwxrwxr-x 2 tigran nobody 512 May 19 12:47 file.X
$ nfs4_getfacl file.X
  A::tigran@desy.afs:r
  A::OWNER@:rwaDxtTcC
  A::GROUP@:rwaDxtc
  A::EVERYONE@:rxtc
$
```

Examples

```
$ touch file.X
$ nfs4_setfacl -a 'A::EVERYONE@:rw' file.X
$ nfs4_setfacl -a 'A::tigran@desy.afs:rw' file.X
$ nfs4_setfacl -a 'A::paul@desy.afs:rw' file.X
$ nfs4_getfacl file.X
  A::paul@desy.afs:rw
  A::tigran@desy.afs:rw
  A::EVERYONE@:rwtc
  A::OWNER@:rwatTcC
  A::GROUP@:rwatc
$ chmod 000 file.X  <- ACL are adjusted here
$ nfs4_getfacl file.X
  A::paul@desy.afs:rw
  A::tigran@desy.afs:rw
  A::OWNER@:tTcC
  A::GROUP@:t
  A::EVERYONE@:t
$
```

Decision maker

- *ACLs processed in top-down order*
- *First DENY ACE stops evaluation*
- *ALLOW ACEs evaluated until all requested masks verified*
- *Fall-back to unix mode if decision can't be made based on ACLs*

Log messages decrypted

- *Access Deny (RO export)*
  - *The client wants to modify on read-only export*
- *Access Deny (no export)*
  - *The client doesn't have an entry in the export file*
- *Access denied: pseudo Inode*
  - *The client want's to modify an object with export path*
- *Access denied:*
  - *The client doesn't have required permission*

Log messages

17 May 2013 14:05:16 (NFSv41-dcache-dir-photon01) () Access Deny: 01caffee00000000102ce059002e303a494e4f44453a303030303133353733413834324232

413442414541363044393430383135444146313244 3a30 T rtc Subject:

Principal: UidPrincipal(16606)

Principal: GidPrincipal(1467,primary)

Principal: GidPrincipal(49)

Principal: GidPrincipal(1467)

Principal: GidPrincipal(3144)

Principal: GidPrincipal(3328)

Principal: GidPrincipal(3844)

Principal: GidPrincipal(3951)

Principal: GidPrincipal(5202)

Principal: GidPrincipal(1100356520)

Real life example

- *Unix mode to DENY*
- *ACL used to for ALLOW*

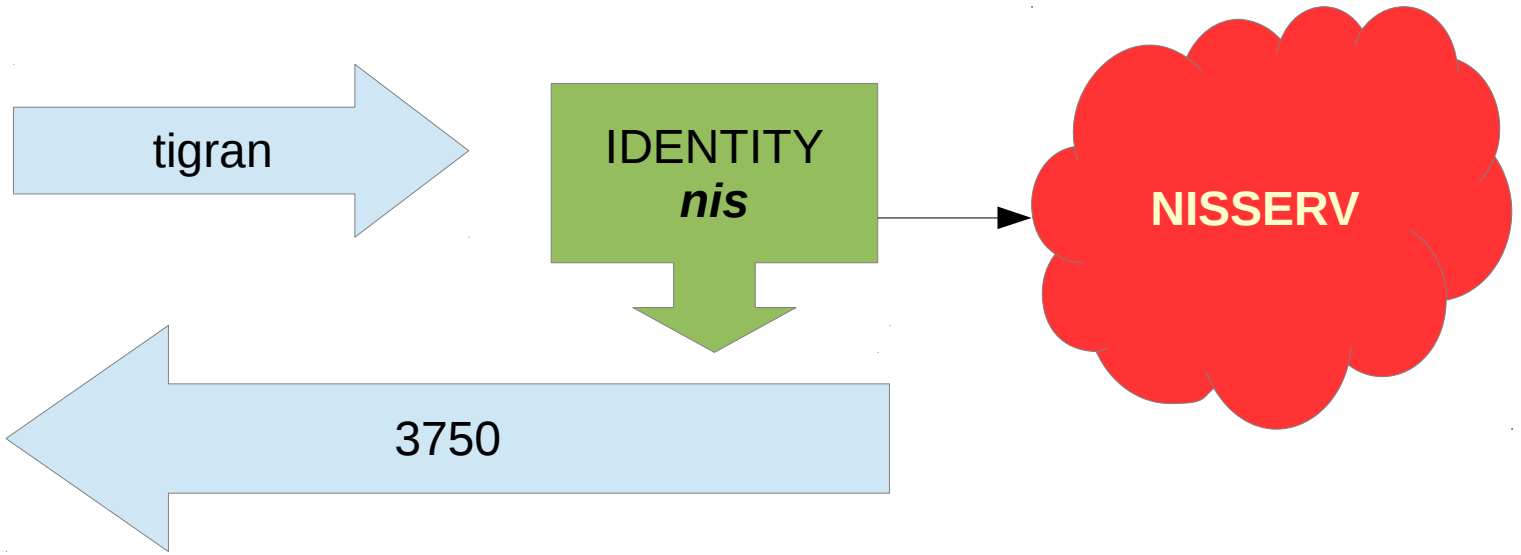
```
(p3-wgs13) /pnfs/desy.de/petra3/disk $  ls -ld dataset1
d--------- 4 psgsrv it 512 Apr 23  14:05 dataset1
(p3-wgs13) /pnfs/desy.de/petra3/disk $  nfs4_getfacl dataset1
  A::psgsrv@desy.afs:rwaDxtTnNcCy
  A::gXXI@desy.afs:rxtncy
  A::rXX@desy.afs:rxtncy
  A::fXX@desy.afs:rxtncy
  A::bXX@desy.afs:rxtncy
  A::OWNER@:tTcC
  A::GROUP@:t
  A::EVERYONE@:t
(p3-wgs13) /pnfs/desy.de/petra3/disk $
```

Internals :

$ nfs4_setfacl -a 'A::tigran@desy.afs:rw' file.X

One Essential component - gPlazma

- *Set/Getacl require proper mapping*
- *Identity plugin should provide One-to-one mapping*
  - *Current plugins NIS, LDAP, NSSWITCH*



Idmapping and gPlazma for NFS 101

- *Client and dCache should use the same nfs domain*
  - *On client: /etc/idmapd.conf*
  - *In dCache nfs.domain in dcache.conf*
- *Identity plugin should be configured in gPlazma*

Troubleshoot mapping errors

- *Use RHEL 6.4 (6.3 is OK)*
- *Check gPlazma mapping*
  - *'get identity / ridentity '*
- *Check door cache*
  - *'login dump cache'*
- *Clear door/client idmap cache*
  - *dCache: 'login clear cache'*
  - *RHEL/SL >= 6.3 : 'nfsidmap -c'*

AUTH_SYS vs. RPCSEC_GSS

Type: CALL		Type: CALL
Uid: 3750 Gids: 1000, 49	**vs.**	tigran@DESY.DE
ls -l /data		ls -l /data

- AUTH_SYS provides uid and gids dcache will use
- RPCSEC_GSS will use uid and gid provided by gPlazma
- If you don't use kerberos, client side uid/gids **MUST** match server side mapping

Troubleshoot mapping errors

(gPlazma) admin > get identity tigran UserNamePrincipal

3750

(gPlazma) admin > get ridentity 3750

(UserNamePrincipal(tigran))

(dcache-lab000.desy.de) (NFSv41-dcache-lab000) admin > login dump cache

Max Cache size: 512

Max Cache time: 30 seconds

Login:

Map:

ReverseMap:

GidPrincipal(1000) => (GroupNamePrincipal(it))

UidPrincipal(3750) => (UserNamePrincipal(tigran))

Summary

- *ACLs is a new hammer in users hand*
  - *Every thing around looks like a big thumb!*
- *Use it as a spice*
  - *Complicated ACLs points to broken model*
  - *DENY ACE mostly points to a broken model*
- *Proper mapping is essential*

More info

$ man nfs4_acl

$ man nfs4_getfacl

$ man nfs4_setfacl

NFS+Krb5+grid

```
auth     optional    x509
auth     sufficient  voms
map      optional    vorolemap
map      sufficient  authzdb
map      optional    gridmap
map      optional    krb5
map      requisite   nsswitch
identity requisite   nsswitch
session  required    authzdb
session  requisite   nsswitch
```

See Paul's presentation for details

Thanks to Andreas Haupt