Linux, Samba and ACLs: past, present, and future

Andreas Grünbacher
Developer
SUSE Labs, Novell

Novell

Linux, Samba and ACLs

- POSIX and POSIX ACLs
- Windows/CIFS ACLs
- Samba 3: POSIX ACL ↔ CIFS ACL mapping
- Samba 4 today: CIFS ACLs in user space
- Beyond?

Novell

Traditional POSIX model

Model

- Each FS object has an owner and an owning group
- Permission sets for owner, owning group, and others
- Read, Write, Execute permissions

```
$ ls -l file
-rw-r--r--  1 agruen users 5 4 May  4 00:00 file
```

Can be viewed as a minimum, three-entry POSIX ACL:

```
$ getfacl –omit-header file
user::rw-
group::r--
other::r--
```

POSIX permission checking (1)

Process requests access. Relevant are:

- effective user ID,
- list of group IDs,
- set of requested permissions.

Two phases:

- Find the best-matching ACL entry,
- Check if the chosen entry contains the requested permissions.

```
# owner: agruen
# group: users
user::rw-
group::r--
other::r--
```

POSIX ACLs

- Each ACL entry specifies a type, qualifier, and a set of permissions.
- Permissions are still only Read, Write, and Execute.
- Permissions for arbitrary additional users and groups:

```
$ setfacl -m user:tux:rw,group:mascots:r file
$ getfacl file
# owner: agruen
# group: users
user::rw-
user:tux:rw-
group::r--
group:mascots:r--
mask::rw-
other::r--
```

- Inheritance model: Default ACLs (next slide)

POSIX ACLs: Default ACLs

Default ACLs are similar to *access* ACLs, but they define which permissions new FS objects obtain:

- Without a default ACL, the *umask* determines the file's permissions.
- With a default ACL, the default ACL determines permissions, and the *umask* is ignored.

Static inheritance: changing the default ACL has no effect on existing child objects.

POSIX ACLs: Default ACLs (2)

```
$ setfacl -d -m user:tux:rwx .
$ getfacl --omit-header .
user::rwx
group::r-x
other::r-x
default:user::rwx
default:user:tux:rwx
default:group::r-x
default:mask::rwx
default:other::r-x
$ touch file2
$ getfacl --omit-header file2
user::rw-
user:tux:rwx    #effective:rw-
group::r-x      #effective:r--
mask::rw-
other::r--
```

POSIX permission checking (2)

Process: effective UID and GIDs; requested permissions

Again, two phases:

- Find the best-matching ACL entry (the ACL entry order does not matter)
- Check if the chosen entry contains the requested permissions

```
# owner: agruen
# group: users
user::rw-
user:foo:rw-
group::r--
mask::rw-
other::r--
```

Named user entry may be “shadowed” by owner entry

CIFS ACLs

(CIFS has DACLs for permissions and SACLs for auditing)

- DACLs are ordered lists of entries.
- Each entry specifies whether the entry is *access-allow* or *access-deny*, a SID, a set of permissions, and a set of inheritance flags.
- Permissions include the equivalent of read, write, execute, things like create, delete, change permissions, take ownership, etc.
- Change permissions and take ownership allow delegation.
- Effective and inheritable permissions are not split.
- Semi-dynamic inheritance: changing inheritable permissions of a directory changes child objects' permissions.

CIFS permission checking

Process: list of SIDs (users and groups treated alike); requested permissions

Go through the list of ACL entries:

- Skip entries that don't apply to the requesting process.
- Deny access if an access-deny entry denies a requested permission.
- Grant all requested permissions that matching access-allow entries allow. Grant the access if all requested permissions have been granted.

If any permissions have not been granted when reaching the end of the ACL, deny access.

Samba 3: POSIX ACLs ↔ CIFS ACLs

User mapping:

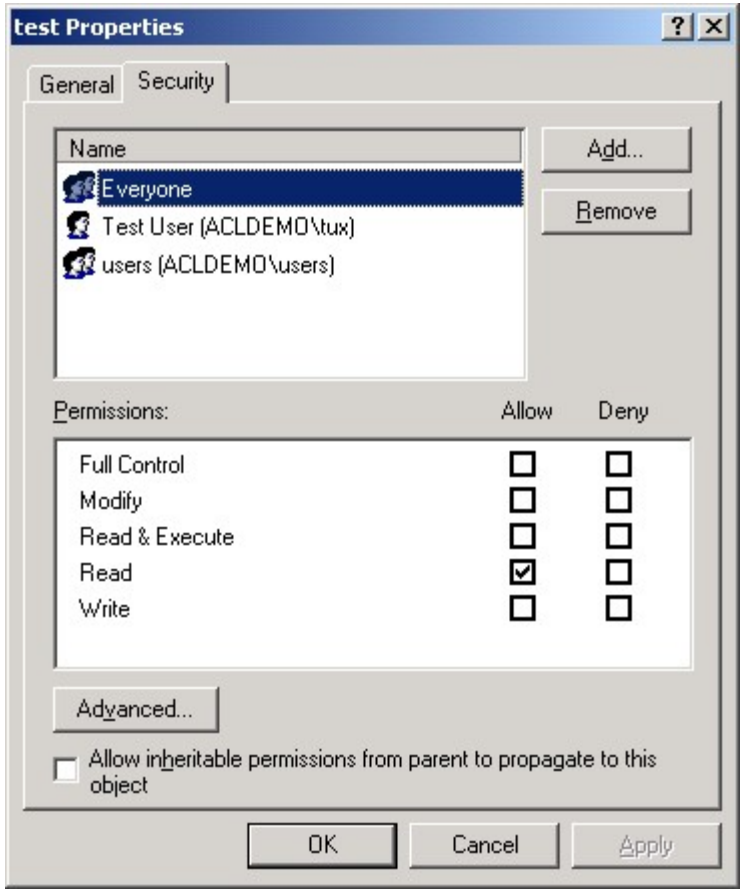
- Owner ↔ current owner
- Owning group ↔ current owning group
- Other ↔ Everyone

Permission mapping:

- r → Read
- w → Write
- x → Execute

sets of perms

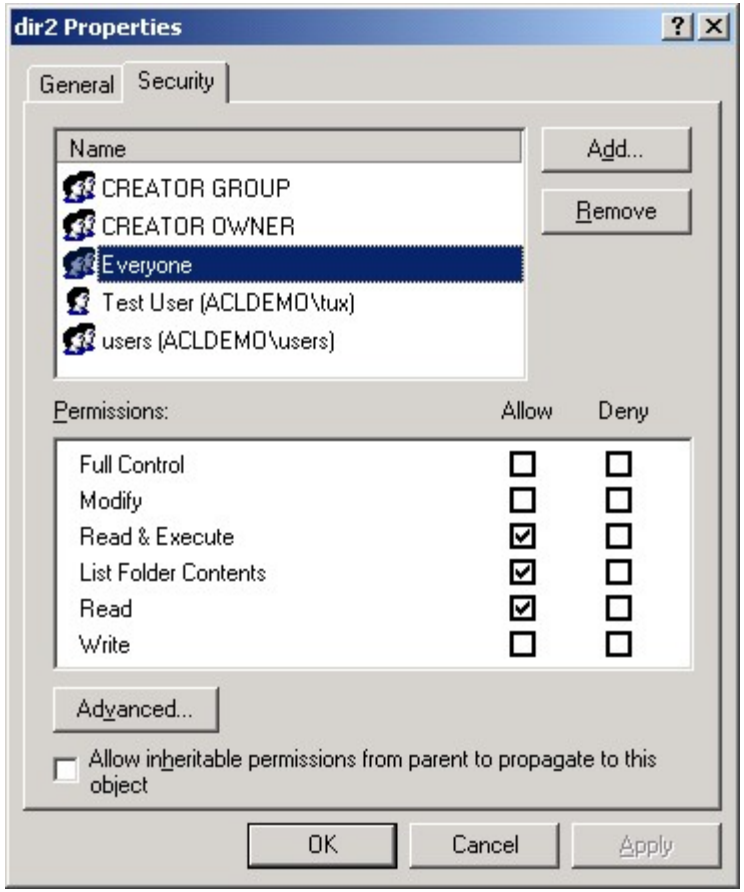
In the reverse direction, any Read/Write/Execute bit adds r/w/x.



Samba 3: POSIX ACLs ↔ CIFS ACLs (2)

Directory / inheritable permissions:

- Owner ↔ CREATOR OWNER
- Owning group ↔ CREATOR GROUP



Novell

Mapping problems

- Lossy conversion: permissions are missing on the POSIX side. The POSIX ACL model is hard/impossible to extend.
- Accumulation vs. selection: Mapping POSIX onto CIFS accurately would require mixed access-allow/access-deny entries; Windows GUI cannot handle this.
- Static vs. dynamic inheritance: CIFS inheritance flags mean something different than what Samba uses them for.
- In CIFS, everything is a SID; even groups can own files.
- Abstract owner vs. user-specific entry -> chown
- Owning group concept not really used under Windows...

Samba 4: CIFS ACLs

Samba 4 implements CIFS ACLs in user-space: no lossy mapping anymore!

- No kernel changes, so portable
- Inconsistent view between Samba and POSIX applications
- When CIFS permissions are defined, POSIX applications should really be governed by the CIFS ACL model, but they will only see traditional POSIX or POSIX ACLs.
- Last writer approach (e.g., NetApp): When a CIFS ACL is set, switch to the CIFS model. When a POSIX ACL is set, switch to the POSIX model.

Other ACL models

NetWare ACLs (called “trustees” there)

- Less complicated, but also more limited than CIFS ACLs.
- Only those files to which a process has access to are visible. Visibility propagates up to the root => ACLs become large. Reading a directory entry requires a permisison check.
- Requres client modifications!

NFSv4

- Model very similar to CIFS, but:
- Inheritance is static, so it's conceptually different

More information: http://www.suse.de/~agruen/

Thank you!

Novell®

General Disclaimer

This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. Novell, Inc., makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc., reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All Novell marks referenced in this presentation are trademarks or registered trademarks of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.

SUSE
A NOVELL COMPANY

Novell