Permissions for a Wider World

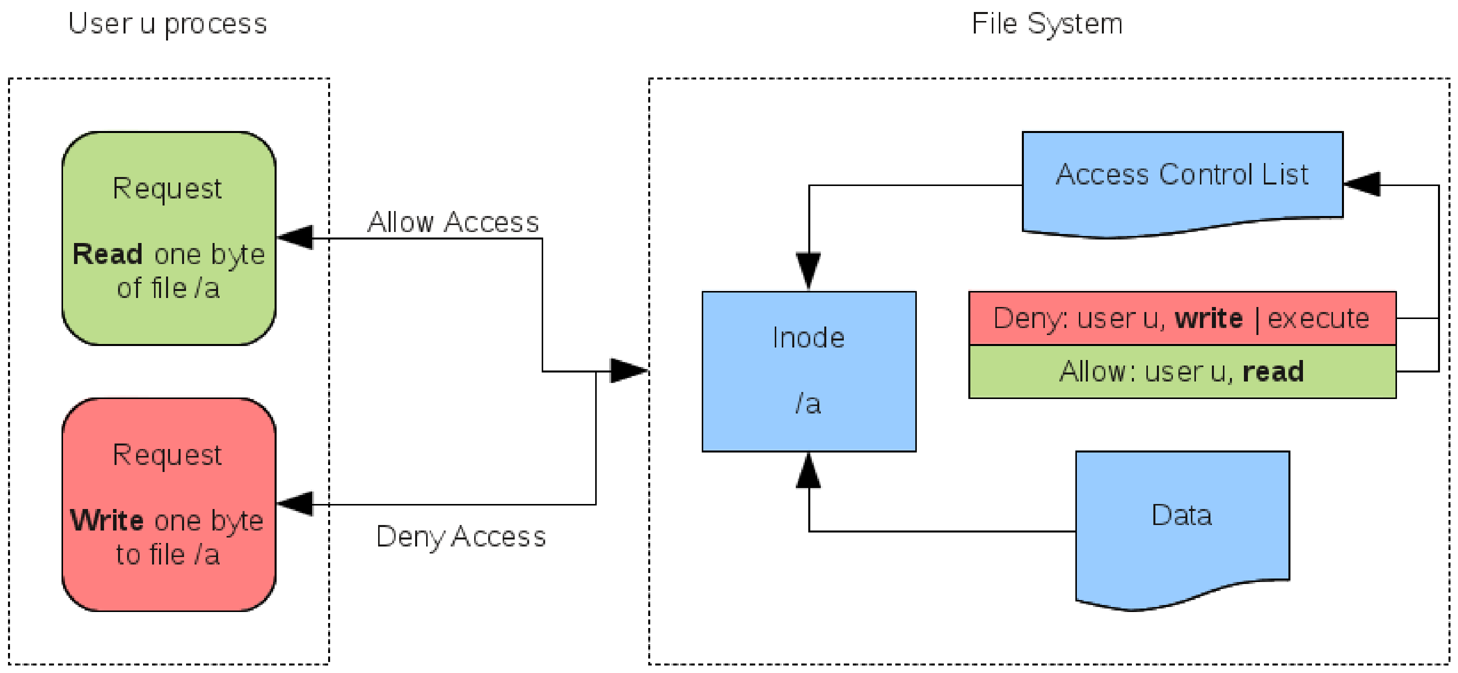
David Disseldorp
SUSE LINUX GmbH

What is Access Control?

- Dictate who can access an object
  - Object = file, directory, device etc.
  - Who = user, group
- The restrictions placed on access
  - Allow read and execute, deny write etc.

- These components can be combined to form an Access Control Entry (ACE)
  - ACEs are bundled together to form an Access Control List (ACL)

What is Access Control?



Problem Statement

Across Operating Systems and protocol implementations, definition of access control rules and how such rules are enforced varies greatly. In particular there is a vast difference between Windows and traditional Unix methods.

This talk aims to:

- Provide an overview of existing UNIX permissions models
- Outline Windows security descriptors
- Discuss existing permissions mapping schemes implemented in Samba
- Introduce Rich ACLs - a new permissions model for Linux

Traditional UNIX Permissions

The POSIX specification defines a file *mode:*

- Files have an owner (UID), and an owning group (GID)
- Requests classed into owner, owning group or other
- Grant who is allowed to *read*, *write* or *execute* a file
- Create/delete permission governed by parent directory
  - Write permission required on parent directory
  - Sticky bit for non-owner deletion restrictions (/tmp)
- Creates require a corresponding mode for the new object

POSIX Draft ACLs

- POSIX 1003.1e / 1003.2c draft 17
- Extends the owner/group/other scope of the *mode*
  - ACEs can refer to a specific UID or GID
  - Same read/write/execute restrictions
- Default ACLs can be defined for basic create time inheritance
- Implemented in many modern file systems
  - Supported by XFS, EXT3, EXT4 and btrfs

Windows Security Descriptors

- Made public with the release of Windows NT
- Attributed to various system entities
  - Files, registry keys, active directory objects, group policies, printers and faxes
- Security Descriptor
  - Discretionary ACL (DACL) defines who is granted access to a file
  - System ACL (SACL) defines access auditing rules
- Vastly more expressive and complex (confusing)
- All users and groups are assigned a security identifier (SID)
  - Single namespace for both

Windows Permissions

- *traverse folder / execute file*
- *list folder / read data*
- *read attributes*
    - File system attributes for a file or folder (hidden etc.)
- *read extended attributes*
    - View application defined attributes for files or folders
- *create files / write data*
- *create folders / append data*
    - Append data restricts overwriting existing file data
- *write attributes*
- *write extended attributes*

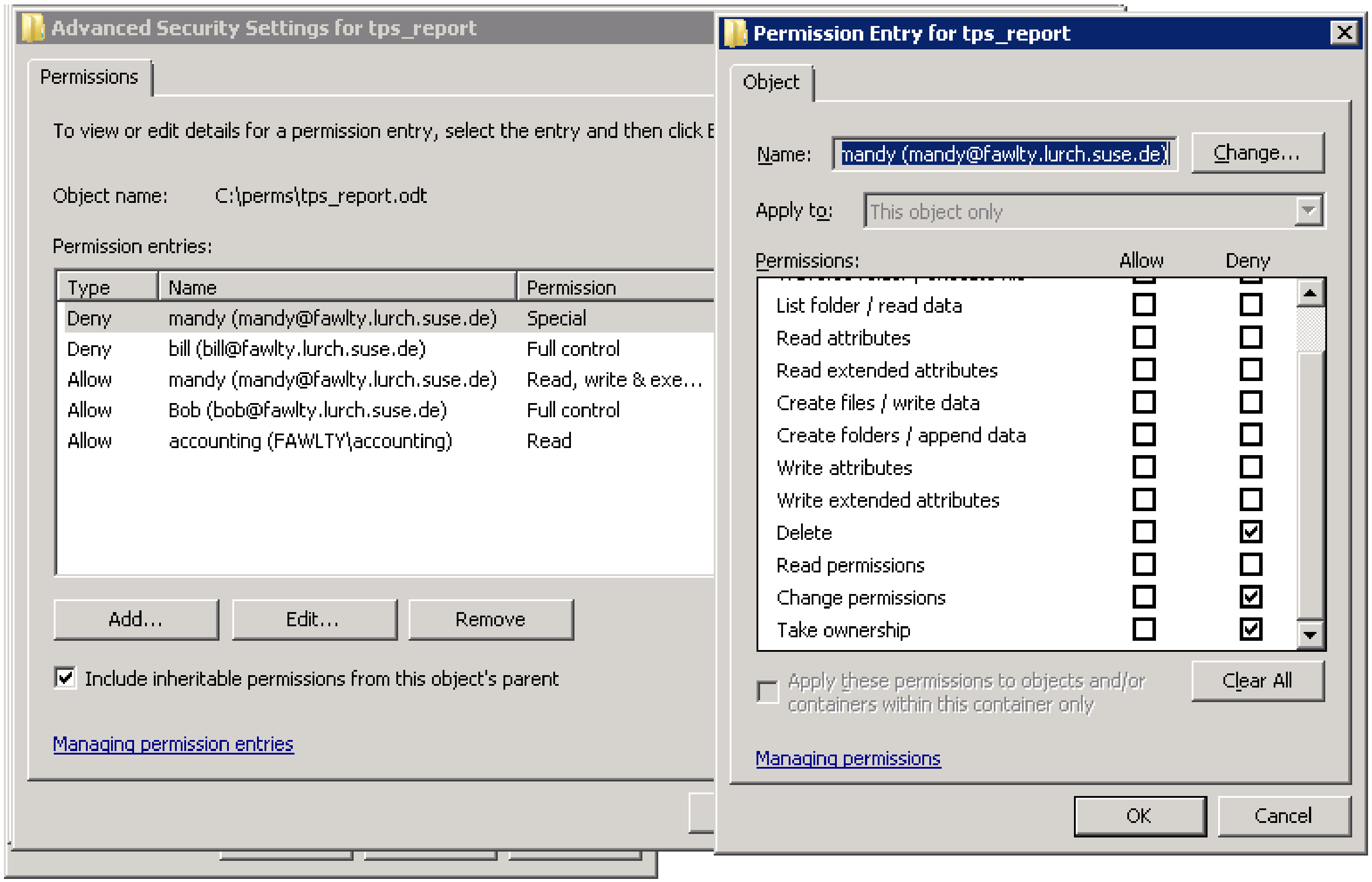
Windows Permissions cont.

- *delete sub-folders and files*
- *delete*
    - Parent delete sub-folders and files **or** explicit delete
- *read permissions*
- *change permissions*
    - File owner always has permission to read and change the ACL except under special circumstances
- *take ownership*
- *synchronize*
    - Allows threads to wait on the handle for the file or folder and synchronize with another thread.

Windows Access Control Example

- Allow Bob full access to the file.
- Allow Mandy to read, write and execute the file
- Do not allow Mandy to change the DACL, owner or to delete the file.
- Allow everyone in the accounting group to read from the file except for Bill.

Windows DACL Example cont.



Advanced Security Settings for tps_report

Permissions

To view or edit details for a permission entry, select the entry and then click E

Object name: C:\perms\tps_report.odt

Permission entries:

Type	Name	Permission
Deny	mandy (mandy@fawlty.lurch.suse.de)	Special
Deny	bill (bill@fawlty.lurch.suse.de)	Full control
Allow	mandy (mandy@fawlty.lurch.suse.de)	Read, write & exe…
Allow	Bob (bob@fawlty.lurch.suse.de)	Full control
Allow	accounting (FAWLTY\accounting)	Read

Add… Edit… Remove

[✓] Include inheritable permissions from this object's parent

Managing permission entries

Permission Entry for tps_report

Object

Name: mandy (mandy@fawlty.lurch.suse.de) Change…

Apply to: This object only

Permissions:	Allow	Deny
List folder / read data	☐	☐
Read attributes	☐	☐
Read extended attributes	☐	☐
Create files / write data	☐	☐
Create folders / append data	☐	☐
Write attributes	☐	☐
Write extended attributes	☐	☐
Delete	☐	✓
Read permissions	☐	☐
Change permissions	☐	✓
Take ownership	☐	✓

☐ Apply these permissions to objects and/or containers within this container only

Clear All

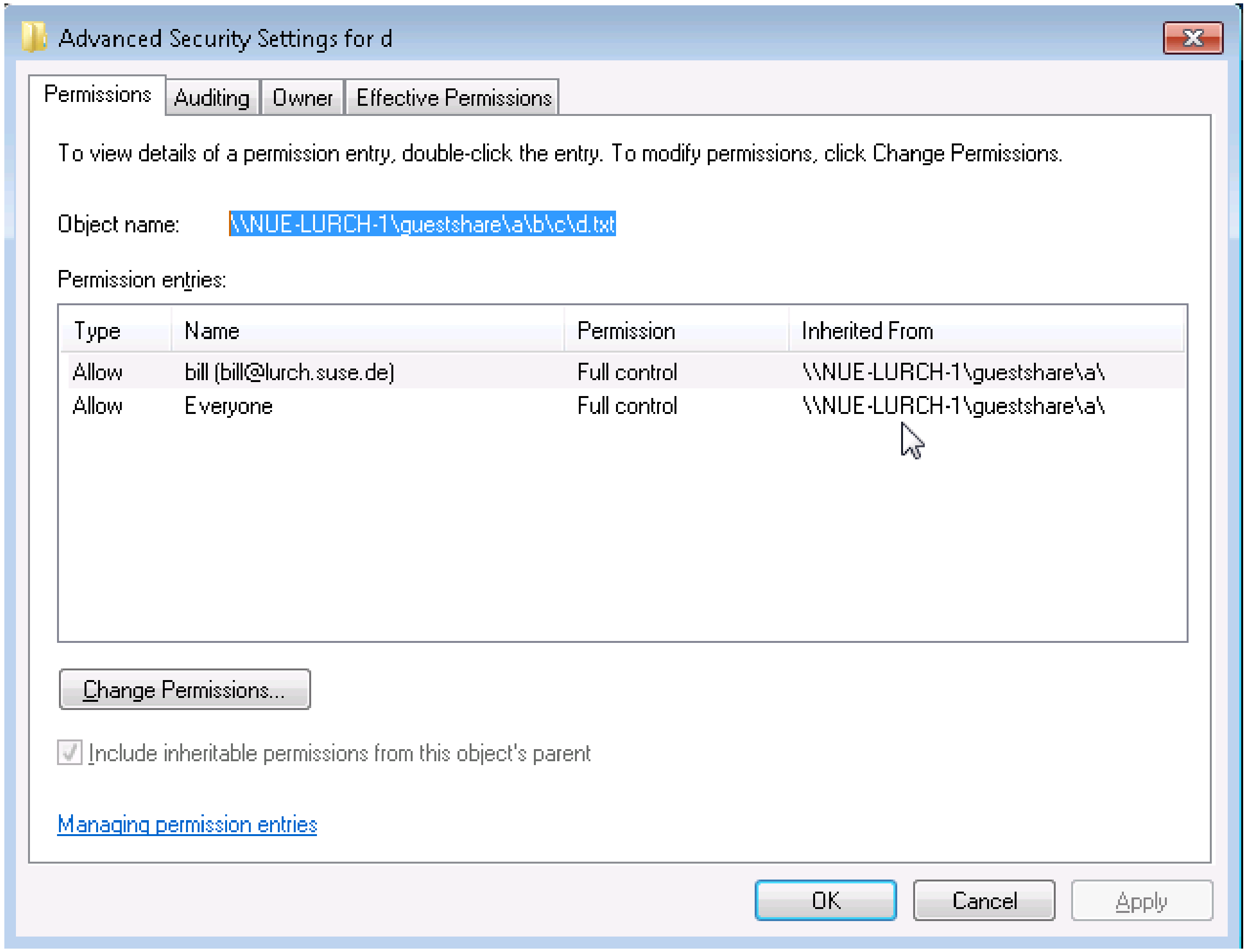
Managing permissions

OK Cancel

Windows ACE Inheritance

- *container inherit* and *object inherit*
    - Which child objects should inherit the ACE
- *inherit only*
    - Whether the ACE applies to this object or only children
- *no propagate inherit*
    - Only propagate the ACE to immediate children of this object, no further
- *inherited ace*
- A security descriptor with the *DACL protected* flag set will not inherit DACL ACEs from parent folders

Windows ACE Inheritance cont.



Advanced Security Settings for d

Permissions | Auditing | Owner | Effective Permissions

To view details of a permission entry, double-click the entry. To modify permissions, click Change Permissions.

Object name: \\NUE-LURCH-1\guestshare\a\b\c\d.txt

Permission entries:

Type	Name	Permission	Inherited From
Allow	bill (bill@lurch.suse.de)	Full control	\\NUE-LURCH-1\guestshare\a\
Allow	Everyone	Full control	\\NUE-LURCH-1\guestshare\a\

Change Permissions...

☑ Include inheritable permissions from this object's parent

Managing permission entries

OK | Cancel | Apply

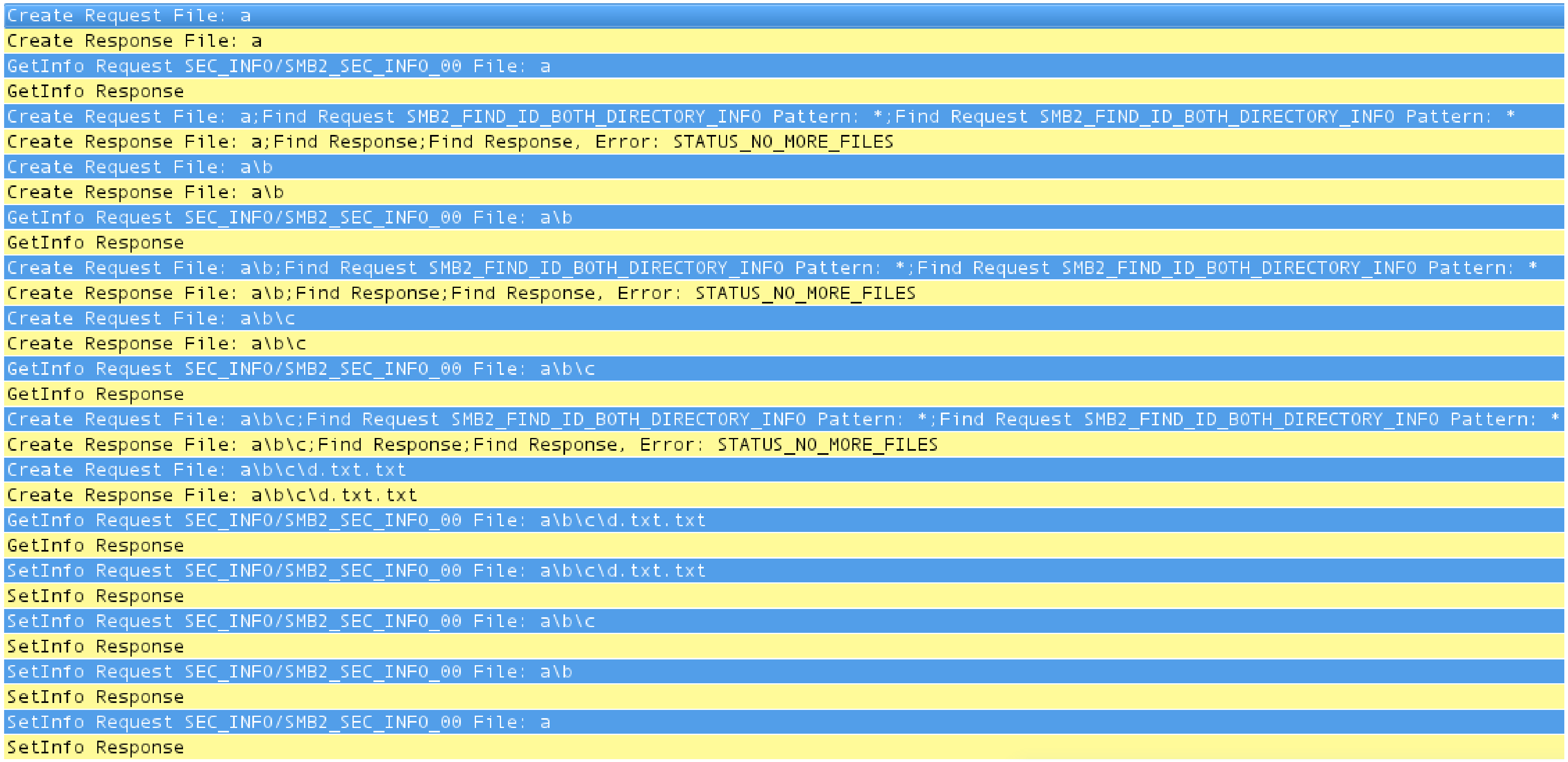
Automatic Inheritance

The Windows security settings user interface only allows modification of inherited aces through the parent DACL.

•Propagation of parent ACL modifications to child objects is not performed by the underlying file system

- Differs to create-time inheritance
- This behaviour is indicated by the following network trace taken between a Windows 2008 server and a Windows 7 client...

Automatic Inheritance cont.



```
Create Request File: a
Create Response File: a
GetInfo Request SEC_INFO/SMB2_SEC_INFO_00 File: a
GetInfo Response
Create Request File: a;Find Request SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *;Find Request SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *
Create Response File: a;Find Response;Find Response, Error: STATUS_NO_MORE_FILES
Create Request File: a\b
Create Response File: a\b
GetInfo Request SEC_INFO/SMB2_SEC_INFO_00 File: a\b
GetInfo Response
Create Request File: a\b;Find Request SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *;Find Request SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *
Create Response File: a\b;Find Response;Find Response, Error: STATUS_NO_MORE_FILES
Create Request File: a\b\c
Create Response File: a\b\c
GetInfo Request SEC_INFO/SMB2_SEC_INFO_00 File: a\b\c
GetInfo Response
Create Request File: a\b\c;Find Request SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *;Find Request SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *
Create Response File: a\b\c;Find Response;Find Response, Error: STATUS_NO_MORE_FILES
Create Request File: a\b\c\d.txt.txt
Create Response File: a\b\c\d.txt.txt
GetInfo Request SEC_INFO/SMB2_SEC_INFO_00 File: a\b\c\d.txt.txt
GetInfo Response
SetInfo Request SEC_INFO/SMB2_SEC_INFO_00 File: a\b\c\d.txt.txt
SetInfo Response
SetInfo Request SEC_INFO/SMB2_SEC_INFO_00 File: a\b\c
SetInfo Response
SetInfo Request SEC_INFO/SMB2_SEC_INFO_00 File: a\b
SetInfo Response
SetInfo Request SEC_INFO/SMB2_SEC_INFO_00 File: a
SetInfo Response
```

Special SIDs

Some such SIDs demand special treatment when present in security descriptors:

- *OwnerRights*
  - Normally an objects owner is always granted READ_CONTROL and WRITE_DAC rights, however ACEs effecting OwnerRights SID can revoke these privileges.
- *CreatorOwner*
  - Placeholder SID substituted at inheritance time with the SID for the objects creator.
- *CreatorGroup*
  - Placeholder SID substituted at inheritance time with the SID for the creators primary group.

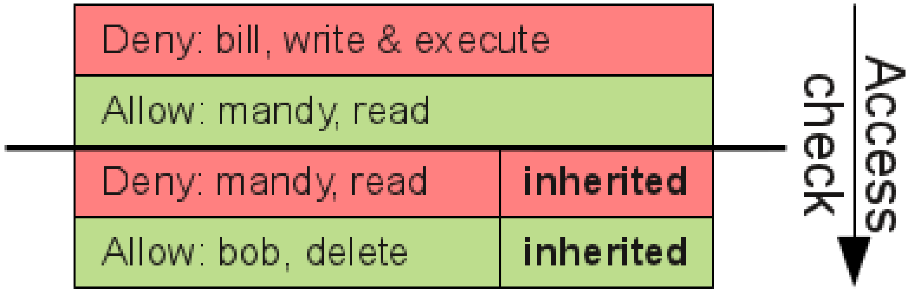
DACL Assignment for New Objects

- Use the DACL carried in the SD with the create request, merge inherited ACEs unless the DACL is protected.
- If no SD was passed with the create request, then create the DACL using only inherited ACEs.
- If there are no inheritable ACEs, then create the DACL using the creators primary or impersonation token (security information for an authenticated session).
- If no token exists then create the object with a null DACL.
    - Null DACLs allow everyone full access and can result in unexpected behaviour when inheritance occurs.

DACL Ordering Requirements

Since Windows NT4 specific *canonical* ordering of ACEs making up a DACL is expected:

•Directly applied ACEs have precedence over inherited ACEs, and must be place ahead of inherited ACEs in a DACL



•Deny ACEs take precedence over Allow ACEs only within their inheritance group

Samba Mapping Schemes

Samba POSIX ACL Mapping

Used by default when Samba build is configured with POSIX ACL libraries and the file system supports them.

- Fall back to file mode otherwise
- *create/directory mask* – upper bound mask for the mode on new files/directories
- *force create/directory mode* – mode bits to always set on new files/directories

Convert all Windows ACE permission bits

- READ_DATA|READ_EA|READ_ATTRIBUTES = Unix Read
- WRITE_DATA|APPEND_DATA|WRITE_EA|
  WRITE_ATTRIBUTES = Unix Write
- EXECUTE = Unix Execute

•Collapse DACL deny and allow ACEs

Samba POSIX ACL Mapping cont.

Various smb.conf options influence mapping:

- *map acl inherit*
    - Stores inheritance information as a special user.SAMBA_PAI extended file attribute.
    - Default ACLs are used in combination for create time inheritance.
- *force unknown acl user*
    - Unknown owner SID takes the value of the currently connected user.
    - ACEs with unknown SIDs are ignored.

Samba POSIX ACL Mapping cont.

- *profile acls*
  - ensures file ownership is always attributed to administrators and users built-in SIDs, members of users built in group are always granted full access.
- *acl compatibility*
  - Windows NT4 specific workarounds.
  - Unable to display ACEs which grant no permissions.
  - Unable to handle DACLs where an inherit only ACE exists without a corresponding non-inherit for the same SID.
- Maturity and feature set of the posix_acls.c mapping code has seen it grow to nearly 5000 lines.

Samba Extended Attribute and tdb Based Storage

- Samba 4
  - POSIX NTVFS back-end uses extended attributes
  - Linux Security Module (LSM) initially planned for enforcement outside of Samba
- Samba 3
  - *acl_xattr* and *acl_tdb* VFS modules
  - include functionality to propagate inherited ACEs from parent security descriptors at file create time.
  - tdbs used for printer and registry security descriptors, however inheritance is not supported in these cases.
- Both methods rely upon permissions enforcement and inheritance propagation outside of the underlying file system.

The Need for a New ACL Model

Existing mapping schemes provide functionality sufficient for many users, but there remain a number of shortcomings:

- Schemes which purely map between Windows security Descriptors and POSIX ACLs lose too much information.
  - SDs go far beyond what can be represented in a POSIX ACL.
  - Such information loss presents potential for security holes.
- Administrators lose the ability to express concise access rules for files on Samba shares which they have grown familiar with on Windows file shares. Subtractive allow-deny rules cannot be represented clearly.

The need for a new ACL model

- Schemes which rely on components outside of the underlying file system for storage and enforcement of permissions add complexity and increase the potential for security holes.
    - Access methods outside of Samba (e.g. NFSv3 and local console) are impractical, as applications require knowledge of the external components to transparently manipulate permissions.
- The disparities between Windows and Unix permissions models has seen a large amount of configuration options added to smb.conf

Network File System version 4 Access Control Lists

- NFSv4 spec attempts to narrow the gap between Unix and Windows access control mechanisms
  - Defines a new NFSv4 ACL type that includes more of the fine grained properties of Windows security descriptors
  - Access mask includes flag space for all Windows permissions
  - Subtractive access rules can be defined with the combination of allow and deny ACEs
  - ACL and ACE inheritance flags are present and provide similar semantics
  - user@dns_domain strings, special OWNER@, GROUP@, and EVERYONE@ identifiers
    - UID/GIDs kept for backwards compatibility

Introducing Rich ACLs

- NFSv4 ACL implementation for Linux
- Joint project between Andreas Grünbacher while at SUSE/Novell and the SGI fileserving technologies team
    - Goal of implementing support for NFSv4 ACLs in Linux XFS, NFS and Samba.
- Paper written by Aneesh Kumar K.V, Andreas Grünbacher and Greg Banks “Implementing an advanced access control model on Linux”

Introducing Rich ACLs cont.

- Attempts to provide support without breaking existing POSIX compliant applications:
  - Restrictions placed on additional and alternate file access control mechanisms defined by the POSIX.1 standard
  - Additional file access control mechanisms may only further restrict the access permissions defined by the file permission bits.
  - Alternate file access control mechanisms may restrict or extend the access permissions defined by the file permission bits.
    - Must be enabled explicitly on a per-file basis, must not be enabled on new files by default.
    - Changing a file’s permission bits with the chmod system call must disable them.

Introducing Rich ACLs cont.

Carries a number of other characteristics to improve interoperability with applications and ease transition from existing models:

- UID and GIDs, plus special USER@, GROUP@ and EVERYONE@, rather than string based identifiers.
  - Existing applications need not be changed to handle mapping of string based identifiers.
- POSIX file masks are carried alongside ACEs and are applied as part of the access check algorithm.
  - Allows simplified logic when handling the chmod system call, as it need only change the file masks
  - Same concept as POSIX ACLs

Introducing Rich ACLs cont.

- Currently, Rich ACL support has been developed for:
  - The Linux EXT4 file system, along with user space libraries and tools for manipulation
  - The Linux NFS server and client
  - Samba, through an experimental VFS module
- All of the real work was done by others
  - I'm here to take credit for it

Rich ACLs Kernel Support

The changes touch a number of core components:

VFS

- Additional file/directory create, delete and append permission flags

EXT4

- Native storage of encoded Rich ACLs on file system objects as an extended attribute
- Logic for enforcement of new permissions and inheritance flags
- Support online migration from POSIX ACLs to Rich ACLs

Rich ACLs Kernel Support cont.

NFS client and server

- Storage and retrieval of Rich ACLs, for conversion to and from wire format

Presentations by Andreas Grünbacher at Linux Plumbers Conference and Greg Banks at the Ottawa Linux Symposium introduced the concept of Rich ACLs to the Linux community.

The kernel patches have been proposed for mainline inclusion and are currently going through concept and source review iterations.

git://git.kernel.org/pub/scm/linux/kernel/git/kvaneesh/linux-richacl.git

git://git.kernel.org/pub/scm/linux/kernel/git/agruen/linux-2.6-richacl.git

Rich ACLs User Space Libraries and Tools

richacl command line utility

- Manipulation of Rich ACLs on local file system
- Dump and restore to and from files

Run-time and development libraries

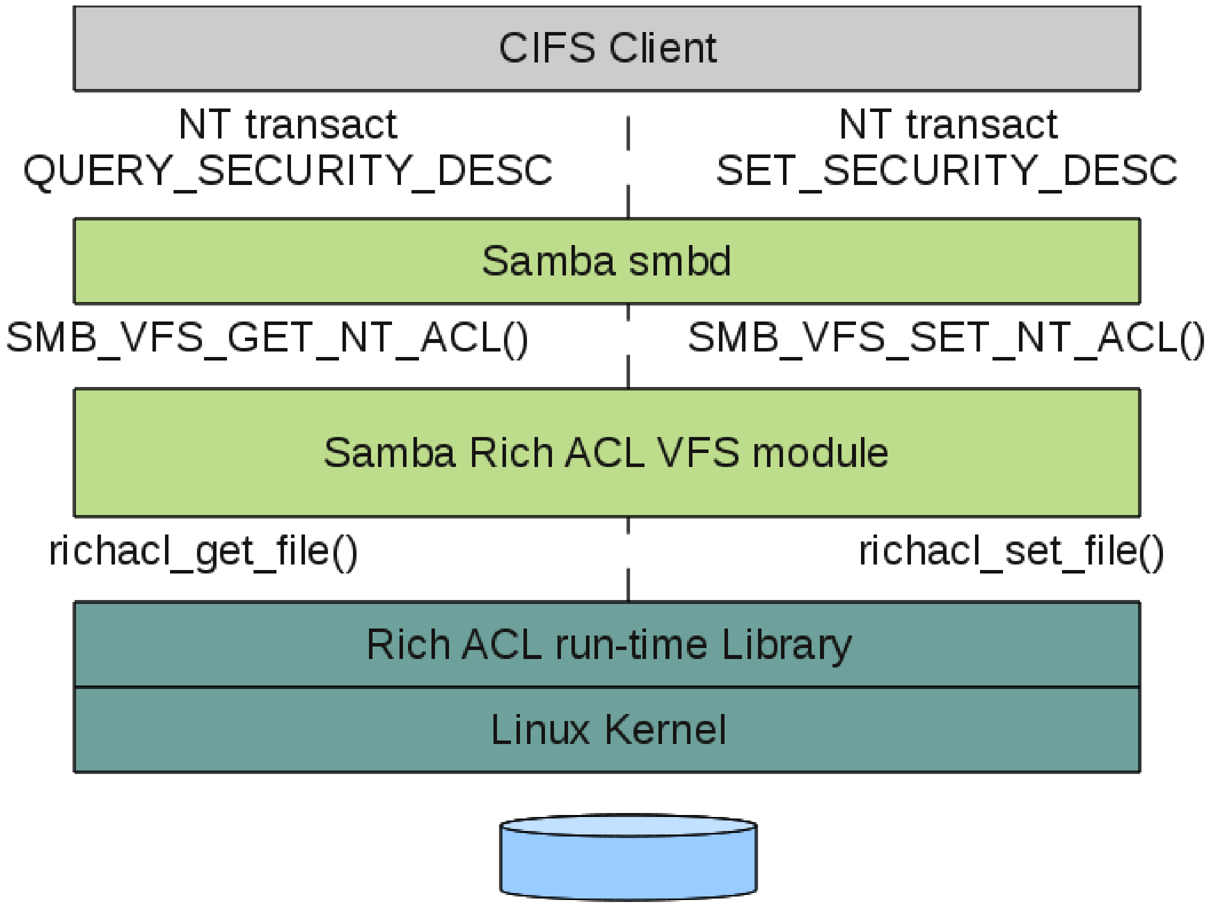
git://git.kernel.org/pub/scm/linux/kernel/git/agruen/richacl.git

Samba Rich ACL VFS Module

- Module performs mapping of Windows security descriptors to and from Rich ACLs
  - Not overly complex given the similarities
- Storage and enforcement provided by the underlying Rich ACL enabled EXT4 file system.
- Where a Rich ACL is not present for a queried file, the security descriptor is generated from the file mode.
- ACEs are reordered into canonical form (explicit allow, explicit deny, inherit allow, inherit deny) for transmission back to the client.
- not yet proposed for acceptance upstream

git://oss.sgi.com/v4acls-experimental/samba.git

Samba Rich ACL VFS Module: Architecture


CIFS Client
NT transact QUERY_SECURITY_DESC
NT transact SET_SECURITY_DESC
Samba smbd
SMB_VFS_GET_NT_ACL()
SMB_VFS_SET_NT_ACL()
Samba Rich ACL VFS module
richacl_get_file()
richacl_set_file()
Rich ACL run-time Library
Linux Kernel

Samba Rich ACL VFS Module: User and Group Mapping

- User and group mapping for file ownership and ACE assignment is handled by the existing Samba UID/GID to SID mapping utilities (winbind etc.).
  - EVERYONE@ identifier is mapped directly to the everyone well-known SID.
- Currently if ID mapping fails for file owner or ACE assigned users the client Get/Set ACL request is denied.
  - Silently dropping file ownership or ACE subject information has potential security ramifications; for example dropping a deny ACE.

Samba Rich ACL VFS Module: Inheritance Mapping

- The Samba VFS module relies on the underlying Rich ACL file system for propagation inherited ACLs at create time.
    - Flags effecting propagation of inherited ACEs preformed by the Windows client are stored.

Samba Rich ACL VFS Module: Missing Functionality

- Handle security descriptors passed in at create time
  - Proper (atomic) create with SD support requires mkdir / open with richacl argument:
  - POSIX does not offer the concept of create with ACL/extended attribute
- ACLs containing unmappable SIDs cause setacl to fail
  - support *force unknown acl user* option. Better yet, winbind support for arbitrary mapping from any Windows SID to a UNIX uid or gid.
- CREATOR_OWNER and CREATOR_GROUP ACEs are not supported

Samba Rich ACL VFS Module: Missing Functionality

- Automatic inheritance flags SEC_DESC_DACL_AUTO_INHERITED and ACL4_AUTO_INHERIT are mapped to each other
  - These flags offer differing semantics on Windows and Linux.
- Workarounds for NT4 ACL compatibility not implemented
- Support setting a security descriptor with a NULL DACL, such security descriptors should provide everyone full access.
- Documentation including a man page

Packages

The richacl openSUSE build service project:

- Linux 2.6.38 kernel with built-in Ext4 Rich ACL support
- Runtime and development libraries
- Samba 3.5.8 with the richacl VFS module

SUSE gallery richacl alpha appliance

- » Includes kernel, library and Samba components.
- » The appliance boots with an Ext4 richacl test file system mounted via loopback.
- » Allows simple and immediate testing in an isolated environment.

Testing

Samba raw.acls smbtorture test group

- Exercises security descriptor operations over CIFS
- Coverage of less obvious corner cases is included, such as NULL DACL and CREATOR_OWNER inheritance tests.

Rich ACL user space source repository

- Battery of tests to exercise permissions granting and enforcement semantics on a locally mounted Rich ACL enabled file system.

A comprehensive NFS test suite based on newpynfs

•git://oss.sgi.com/v4acls-experimental/newpynfs.git

How to Get Involved

- More information at http://acl.bestbits.at/richacl/
- Always in need of testers and end users to apply pressure with the goal of upstream kernel acceptance.
  - Let your favourite Linux distribution know that you are interested in the features Rich ACLs offer.
- Developers on irc.freenode.net, channel #richacl

Questions?

References

POSIX Access Control Lists on Linux

http://www.suse.de/~agruen/acl/linux-acls/online/

File Security and Access Rights

http://msdn.microsoft.com/en-us/library/aa364399.aspx

Network File System (NFS) version 4 Protocol

http://tools.ietf.org/html/rfc3530

Security Identifiers (SIDs) New for Vista

http://technet2.microsoft.com/WindowsVista/en/library/ea557e82-e51c-484f-be3a

oss.sgi.com Git

http://oss.sgi.com/cgi-bin/gitweb.cgi?

File It: Understanding ACLs in NTFS

http://blogs.msdn.com/brian_dewey/archive/2004/01/20/60902.aspx

performance of filesystem xattrs with Samba4

http://lwn.net/Articles/112567/

References cont.

Microsoft Windows Internals - fourth.edition

http://book.itzero.com/read/microsoft/0507/microsoft.press.microsoft.windows.ir

ZFS ACLs

http://blogs.sun.com/marks/entry/zfs_acls

Network File System (NFS) Version 4 Minor Version 1 Protocol

http://tools.ietf.org/html/rfc5661

Implementing an advanced access control model on Linux

http://www.fmeh.org/ols-2010-implementing-richacl-paper.pdf

What is the Windows Integrity Mechanism?

http://msdn.microsoft.com/en-us/library/bb625957.aspx

NetApp multiprotocol data access

http://media.netapp.com/documents/wp_3014.pdf

The Open Group. Portable Operating System Interface.

http://www.unix.org/version3/

References cont.

DACL for a New Object

http://msdn.microsoft.com/en-us/library/aa446598%28v=VS.85%29.aspx

Well-known security identifiers in Windows operating systems

http://support.microsoft.com/kb/243330

Rich ACL Kernel patch proposal

http://thread.gmane.org/gmane.linux.file-systems/51260

richacl OpenSUSE build service project

https://build.opensuse.org/project/show?project=home:dmdiss:richacl

Automatic Propagation of Inheritable ACEs

http://msdn.microsoft.com/en-us/library/aa376326%28v=vs.85%29.aspx

Order of ACEs in a DACL

http://technet.microsoft.com/en-us/library/cc961994.aspx

Implementing Native NFSv4 ACLs in Linux

http://oss.sgi.com/projects/nfs/nfs4acl/lca2009-implementing-native-nfsv4-acls-in

Novell®

Further Questions

- Registry objects also have security descriptors attached, how to handle inheritance etc. in these cases.
- Owner always has write_attributes, write_acl and take_ownership permission for richacls.
- Size limitations for extended attributes, security descriptors and richacls
- UI “replace all child objects with inheritable acls from this object” performed unconditionally?
- avoid chmod in Samba to ensure alternate bits are retained

Samba file mode mapping

When POSIX ACL support is not available (3.5.8):

- fallback to Unix file mode
- create/directory mask – upper bound mask for the mode on new files/directories
- force create/directory mode – mode bits to always set on new files/directories
- store dos attributes – allows storage of DOS attributes in an extended file attribute
- map archive/system/hidden – enables mapping to owner/group/other execution mode bits
- map read only – by default the read only attribute clears write permission for user, group and others
- delete readonly

POSIX ACLs Example

Allow Bob and Mandy access to read and write to the file, allow anyone in the accounting or finance groups to read the file, do not allow access from anyone else.

```
bob@dragonfly:~> touch tps_report.odt
bob@dragonfly:~> setfacl -m u:bob:rw tps_report.odt
bob@dragonfly:~> setfacl -m u:mandy:rw tps_report.odt
bob@dragonfly:~> setfacl -m g::0 tps_report.odt
bob@dragonfly:~> setfacl -m g:accounting:r tps_report.odt
bob@dragonfly:~> setfacl -m g:finance:r tps_report.odt
bob@dragonfly:~> setfacl -m o::0 tps_report.odt
```

POSIX ACLs example cont.

```
bob@dragonfly:~> getfacl tps_report.odt
# file: tps_report.odt
# owner: bob
# group: users
user::rw-
user:bob:rw-
user:mandy:rw-
group::---
group:accounting:r--
group:finance:r--
mask::rw-
other::---
bob@dragonfly:~> ls -lah tps_report.odt
-rw-rw----+ 1 bob users 0 Apr 27 15:29 tps_report.odt
```

POSIX ACLs example cont.

```
bob@dragonfly:~> getfacl tps_report.odt
# file: tps_report.odt
# owner: bob
# group: users
user::rw-
user:bob:rw-
user:mandy:rw-
group::---
group:accounting:r--
group:finance:r--
mask::rw-
other::---
bob@dragonfly:~> ls -lah tps_report.odt
-rw-rw----+ 1 bob users 0 Apr 27 15:29 tps_report.odt
```

Others

Samba ships with VFS modules for mapping NT security descriptors to other file systems native ACL implementations:

- ZFS with a pure ACL model
  - Based on the original NFSv4 rfc3530 specification
- IBM GPFS (proprietary clustered filesystem)
  - Native support for NFSv4 ACLs
- Isilon OneFS supports NTFS ACLs