

Non-UNIX Filesystem Permission Models

Andreas Grünbacher
Developer
SUSE Labs, Novell



Novell.



Which permissions?

Filesystem operations

- Create a directory, create a non-directory
- Open a file (for read, write, execute, append)
- Delete a file
- Read or write Extended Attributes
- **Change permissions**
- **Change ownership**
- **Read file meta-information**

Discretionary Access Control (DAC, as opposed to MAC)

- Effective UID, effective GID, supplementary groups determine access

Capabilities (e.g., CAP_CHOWN) are orthogonal



Linux access control model

Modeled in terms of Read, Write, Execute permissions

Call sequence:

- System call

- VFS operation

- Path lookup [permission(path*, x)]

- Access check [permission(dir/inode, rwx), add. checks]

- Filesystem operation (iop)

Example: Open for append

- permission(path*, x); permission(inode, w)



How does this match? Other models?

Filesystem operation: Delete a file

Linux delete access check

permission(path*, x)

permission(dir, wx)

check_sticky(dir, inode)

NFSv4 delete access check

permission(path*, x)

permission(dir, “may delete child”) ||

permission(inode, “may delete inode”)

VFS doesn't tell the filesystem what it wants

Some checks hardwired in the VFS

NFSv4 check can't be “plugged in”



Why support other models? Alternatives?

Without full CIFS (i.e., Windows) permissions, Samba is hard to sell. (Think as a migration path.)

NFSv4 has a permission model close to CIFS, so when taken to the limits of its specification, it will have the same problem.

NSS: similar situation, different permission model.

In-kernel support? Alternatives?

Support for alternative permission models in the kernel

- + Kernel can enforce the permission models
- + Many Linux tools will continue to work under the altered semantics (“the best of both worlds”)

Alternative: User-space (e.g., Samba4 implements CIFS ACLs in user-space, on top of xattrs)

- + Kernel doesn't need to bother
- Not atomic on the file system
- Permissions model does not apply to Linux processes => Linux is losing, Windows will be chosen instead!

Via Linux Security Modules?

LSM hooks into permission() and the VFS operations

Used to introduce further restrictions (MAC-like)

Relaxing POSIX access checks and granting additional kinds of accesses is not possible

Global (as opposed to e.g., per-filesystem)

No tightly coupled with filesystems (atomicity)

Original access check model too coarse =>

MAY_APPEND permission which is only seen by LSM (!)

=> not well suited for DAC extensions



Idea: per-mount permission model

- Most filesystems use POSIX/Linux semantics,
- selected special-purpose filesystems use alternative semantics.

This allows daemons like Samba4 to export different semantics without losing all the benefits of Linux.



Extend VFS permission() and add “permission2” inode operation

- Add permissions: MAY_ADD_{FILE,DIR}, (MAY_APPEND), MAY_DELETE, etc.
- Move hard-wired VFS checks into permission() [e.g., check_sticky()]
- Add (struct inode *dir) argument to VFS permission()
- If permission2 iop undefined, fall back to old logic and permission iop

Backward compatibility idea: File delete check

permission(dir, inode, MAY_WRITE | MAY_EXECUTE | MAY_DELETE, nd)

Fallback case becomes (almost) a matter of masking off bits



Thoughts?

Thanks for your feedback!

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.

No part of this work may be practiced, performed, copied, distributed, revised, modified, translated, abridged, condensed, expanded, collected, or adapted without the prior written consent of Novell, Inc. Any use or exploitation of this work without authorization could subject the perpetrator to criminal and civil liability.



Novell.