

MISSION: IMPOSSIBLE

Emulating Windows file serving on POSIX

SAMBA

**Jeremy Allison
Samba Team**

jra@samba.org

SAMBA

Opening Windows to a
wider world

But isn't it easy ?



**Just take a kernel, add
your own file system and..**

Not if you don't own your own kernel



or file system.

POSIX mapping challenges

- Pathnames and Privilege.
- Extra attributes.
- Open Modes / Oplocks / Leasing.
- File Locking.
- ACLs / Identity mapping.
- Alternate Data Storage / Streams / EA's.
- Change Notification.
- Exotica (Quotas / Snapshots / MSDFS etc.).

Pathnames and Privilege

- POSIX userspace API's are all pathname based.
- SMB2 on the wire protocol is handle based.
 - Less so for SMB1.
 - Incoming pathnames must be converted to handles using the userspace API's.
- Works fine with simple model of mapping Windows users to POSIX uids.

Pathnames and Privilege (continued)

- Case insensitivity can hurt performance.
 - It's not the hits that cost, but the misses.
 - Samba uses a stat-cache to speed up name translation in userspace.
 - If users are happy with canonicalized case, that's the easiest way to deal with this.
 - Some UNIX filesystems support case insensitivity.
 - 8.3 filename mapping still rears its ugly head occasionally.

Pathnames and Privilege (continued)

- POSIX symlinks can ruin your whole day.
 - Can be set via NFS or CIFS UNIX extensions.
 - Should the server follow them ?
 - Harder to decide than it looks.
- Options:
 - `O_NOFOLLOW` on `open()`.
 - Use `lstat/lchown/lchmod` functions.
 - `realpath()`.
 - `fopenat()/fstatat()` and friends.

Why following symlinks can cause problems

Exported root of share:

/safe/export/

a/realative/dir/path/file.txt

Symlink, that points elsewhere

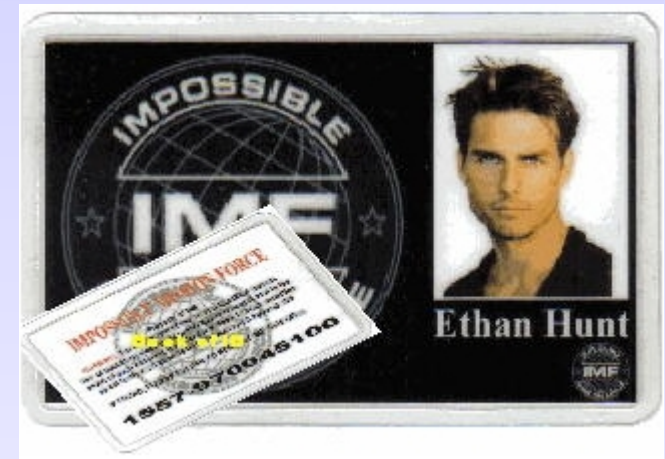
/evil/directory

Pathnames and Privilege (continued)

- Problems with preventing the following of symlinks - O_NOFOLLOW:
 - Not all systems have it.
 - It doesn't do what programmers want or expect.
 - Neither do the lstat/lchown/lchmod functions.
 - fopenat()/fstatat() is worse.
- Two fixes for this:
 - Walk the path a component at a time.
 - Use realpath().

Pathnames and Privilege (continued)

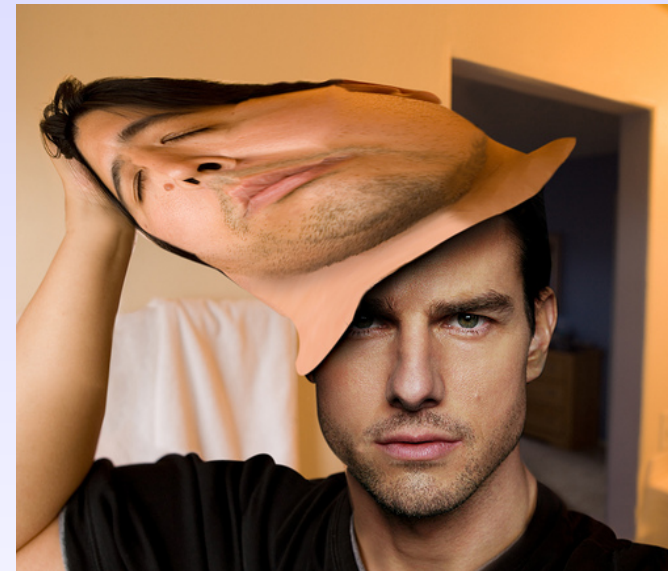
- But what about users with elevated SeBackup/SeRestore privileges ?



- Symlink races are deadly with root access.
 - chdir() into target directory to avoid all the previous problems.
 - It's harder than that..

Extra Attributes

- Windows files have many more attributes than POSIX - how to store them ?
- We cheat and insist on a file system with extended attributes.
 - It's not really POSIX under the rubber mask.
- If you can do without ACLs, simple mappings may be done onto POSIX permission bits.



Open Modes / Oplocks / Leasing

- Using shared memory all the extra meta-data associated with opens can be efficiently shared.
 - So long as CIFS/SMB/SMB2 access is all that is required, this is enough.
 - Issues arise when NFS exports overlap the same file systems.
 - Windows open meta-data can cause NFS clients to fail in unexpected ways.

Open Modes / Oplocks / Leasing (continued)

- Write a user space NFS server that integrates with the CIFS/SMB/SMB2 data model.
- Push required open state meta-data into the kernel.
 - First done in SGI IRIX.
 - Oplocks available as F_SETLEASE/F_GETLEASE in Linux.
 - Share modes passed into Linux kernel (for IBM GPFS) via flock() call.

File Locking

- Finally a solved problem !
 - Even the zero-zero locks :-).
- Shared memory allows a mapping layer on top of POSIXfcntl locks
 - POSIX locks are flexible enough that Windows locks can be mapped onto them.
 - See my previous talk on mapping Windows unsigned file locks onto POSIX signed locks.
 - Atomic operations are possible, but hard to get right.

File Locking (continued)

- Smbtorture suite has a great set of regression tests for file (byte range) locking.
 - Unfortunately can't test cross protocol file locking semantics (interactions with NFS).
- Good news is almost no POSIX applications care about the exact semantics of file locking (except file servers like Samba).
- No significant file locking bugs reported in the past five years or so.

ACLs / Identity mapping

- Underlying system must have some form of ACLs / access control.
 - (Draft) POSIX ACLs will do.
 - NFSv4 ACLs are better.
 - But nothing is perfect here.



ACLs / Identity mapping (continued)

- How to get perfect Windows ACL fidelity mapped onto POSIX access control ? Samba solution:
 - Map incoming Windows ACL onto local filesystem.
 - Store unmodified Windows ACL along with a cryptographic hash of the underlying mapped ACL, read back into Windows format.
 - Allows detection of external modification by NFS or local processes.

ACLs / Identity mapping (continued)

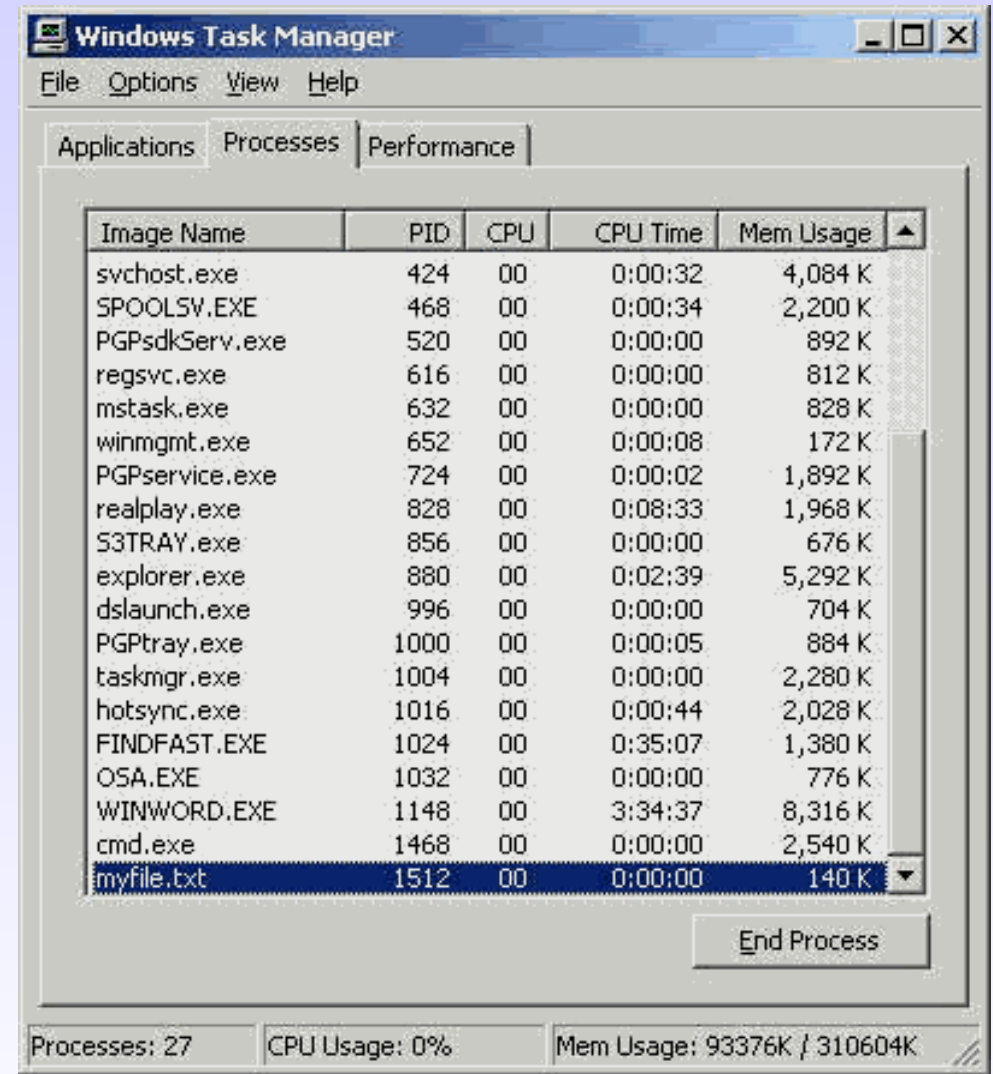
- Windows ACLs (stored with perfect fidelity) control the first level of access.
 - Any Windows DENY is correctly returned.
 - Problems can arise when Windows says ALLOW but POSIX mapping says DENY.
 - Optionally override POSIX permissions.
 - Can lead to root symlink races, see earlier section on pathname processing.

ACLs / Identity mapping (continued)

- Mapping uids / gids to Windows SIDs is conceptually very simple, but there are many ways to do this.
 - Small cottage industry of solutions available.
 - Samba winbindd is the version we ship.
 - All solutions have to deal with the same fundamental issues (“foreign” SIDs, files with group ownership but no user ownership).
 - Some of these problems are very difficult (user/group enumeration).

Alternate Data Storage / Streams / EA's

- Are a bad idea..



- And thankfully are rarely used.

Alternate Data Storage / Streams / EA's (continued)

- Windows extended attributes map easily into UNIX EA's (no such thing in POSIX).
 - Except case sensitivity issue.
- Windows streams in theory cannot map into UNIX EA's.
 - Arbitrary size means they will overflow.
 - In practice, due to small size, this can work.
- Without kernel support, map to “shadow” directories containing data.

Change Notification

- A bad idea for scalability.
 - Allows simple client requests to cause large amounts of server resources to be used.
- A bad idea for clustered file servers.
 - Broad notify on the root of a filesystem can lead to lots of cross communication between cluster nodes.
- Not fully supported in a Windows compatible way on any UNIX.

Change Notification (continued)

- All UNIXes do this differently.
 - Linux has inotify.
 - Solaris / Nexenta has FEN.
 - FreeBSD has fsnotify.
- In the same way as for oplocks, Samba implements a user space implementation that can map onto an underlying kernel version.

Exotica (Quotas / Snapshots etc.)

- POSIX symlinks can hide a multitude of sins.
- Arbitrary blob store on disk, just like EA's but supported on all systems.
 - MSDFS links.
 - Reparse points.
 - Samba uses a prefix to ensure that such objects aren't misinterpreted by NFS clients.
- Only works for objects that don't have file stream content.

Exotica (Quotas / Snapshots etc.)

- Windows is extending the filesystem in new and interesting ways.
- So are Linux, Solaris/Nexenta, FreeBSD and other POSIX-style systems.
- The problem is each system is implementing similar functionality in different ways.
- It truly is impossible to create a perfect mapping.
 - After all, if you really **NEED** Windows..

You know where to license it



SRMBR
Opening Windows to a
Wider World

Exotica (Quotas / Snapshots etc.)

- Samba attempts to map the significant features of each Linux / UNIX implementation into methods that Windows clients can use.
 - Abstract each feature into the Samba VFS (Virtual File System) switch.
- Keeps the Samba implementors job interesting.
 - Probably for another 20 years..

“Your next mission, should you chose to accept it, is to make it all work in a cluster..”



Coming soon, in a talk by Michael Adam of SerNet



SRMBR
Opening Windows to a
Wider World

Questions and Comments ?

Email: jra@samba.org

Slides available at:

<ftp://samba.org/pub/samba/slides/samba-impossible.odp>