Juicing the Fruit

NO TESTS
NO PROBLEMS
„IMPLEMENT TEST CASES [WIP(ISN’T IT ALWAYS? . . .)]“
STATE OF OS X SUPPORT
Apple’s SMB2 protocol extension: AAPL
Spotlight
Interoperability with Netatalk
STATE OF OS X SUPPORT

AAPL
Name of an SMB2_CREATE context

Refer to Apple’s SMB2 extensions as AAPL

How is it used?

After first tcon do SMB2_CREATE on share base directory

AAPL request/response blob used to negotiate certain capabilities (see next slide)
/* Define Client/Server Capabilities bitmap */
enum {
    kAAPL_SUPPORTS_READ_DIR_ATTR = 0x01,
    kAAPL_SUPPORTS_OSX_COPYFILE = 0x02,
    kAAPL_UNIX_BASED = 0x04,
    kAAPL_SUPPORTS_NFS_ACE = 0x08
};

/* Define Volume Capabilities bitmap */
enum {
    kAAPL_SUPPORT_RESOLVE_ID = 0x01,
    kAAPL_CASE_SENSITIVE = 0x02
};
What can we do with it?

- Faster Finder browsing
- Poors man’s POSIX extensions
Faster Finder browsing:

- in extreme cases 5 seconds instead of 2 minutes for 5000 files in a single directory (ctdb cluster or high-latency network link)
- changes how the OS X client retrieves metadata

Without AAPL:

- SMB2_FIND to get list of files, then 5000 requests to retrieve metadata

With AAPL:

- Single SMB2_FIND request and response, done!
- Hack alert: `FILE_ID_BOTH_DIR_INFORMATION` structure elements repurposed
Poor man’s SMB2 POSIX Extensions:

- read POSIX mode, uid and gid
- change mode
- SMB2_{GET|SET}INFO security descriptors with ACEs using special SIDs:
  - S-1-5-88-1-<uid>
  - S-1-5-88-2-<gid>
  - S-1-5-88-3-<mode>
- Used by MS for Services for UNIX (NFS)
STATE OF OS X SUPPORT

SPOTLIGHT
What is Spotlight?

- searchable index of files and their metadata
- searching locally on a Mac, or remotely on a server
- SMB connection to server
- search protocol uses MS-RPC as transport
- similar to MS-WSP
- Samba is just a search query proxy
So if Samba is just a proxy, who does the hard work?

- server backend is Gnome Tracker
- limitations: not cluster aware, primary focus is desktop
- possible other backends: Apache SOLR, ElasticSearch
- targeting servers, enterprisy, clustered
- Samba backend code for SOLR found in NAS vendor GPL source drop
- code is here, needs upstream integration work
STATE OF OS X SUPPORT

INTEROP
Many still run Netatalk based AFP servers

Samba VFS module vfs_fruit adds interop sugar to be compatible with Netatalk:

- metadata storage
- filename encoding
- locking

It mostly works, some known issues:

- OS X xattrs are lost when vfs_fruit is added to a share (Finder tags)
- xattrs incompatible between Netatalk and Samba (streams_xattr stores a trailing 0 byte)
Great new features:

- Apple’s SMB2 protocol extension: AAPL
- Spotlight
- Interoperability with Netatalk

How many tests do we have for this stuff?
NO TESTS
NO PROBLEMS

bad_fruit?
NO TESTS / NO PROBLEMS

$ make test
...
[1970(19486)/1972 at 3h3m34s]
samba4.blackbox.dbcheck(fl2008r2dc)
[1971(19490)/1972 at 3h3m50s]
samba4.blackbox.dbcheck(vampire_dc)
[1972(19494)/1972 at 3h4m18s]
samba4.blackbox.dbcheck(promoted_dc)

ALL OK (19498 tests in 1972 testsuites)
...
$
large number protocol conformance test

this ensures we provide Windows semantics

for OS X clients what matters is OS X semantics

when OS X exports HFS+ via SMB it does not care about conforming to specs

just dumps HFS+ filesystem behaviour on the network

as a result it subtly deviates from the specs and that’s where the fun begins...

…and all this is undocumented: there’s no spec
The shocking truth:

NO TESTS / MANY BUGS

The story of four bugs found and fixed in the last year:

#1: Copying a directory to server
#2: Resource fork
#3: Rename behaviour
#4: FileIDs
NO TESTS / NO PROBLEMS

BUG#1
COPY DIRECTORY TO SERVER
copying directory to server failed with specific OS X version

when implementing vfs_fruit some research on OS X semantics was done

found behaviour that OS X always returns AFP_AfpInfo stream

this was wrong, but worked until it broke subtly with specific OS X release

lesson learned: some research with no tests is not good
NO TESTS / NO PROBLEMS

BUG#2
RESOURCE FORK
Resource fork is a second data stream that can exist per file in HFS+ filesystem

for SMB connections mapped to *AFP_Resource* stream

Where can it go wrong?

- server: a Mac
- client 1: create *AFP_Resource* stream on a file
- client 2: stat() the stream
- What would you expect?
Shocking answer: ENOENT

As long as no data is written to stream, other clients won’t see it

Lesson learned: added tests for OS X semantics of their two special streams AFP_AfpInfo and AFP_Resource
NO TESTS / NO PROBLEMS

BUG#3

RENAME BEHAVIOUR
Windows doesn’t allow renaming of directories with open files (MS-FSA 2.1.5.14.11)

POSIX does allow it, so does OS X

Samba? Doesn’t allow it, OS X clients unhappy

happens frequently because OS X Finder opens a special file `.DS_Store` for every open Finder window

solution: add optional POSIX directory rename behaviour, disabled by default, enabled for OS X clients

OS X clients now happy?
No!

- turns out there’s a bug in the OS X SMB kernel client:
  - applications with open files in renamed directories subsequently fail to save
  - happens with OS X SMB server as well
  - workaround: add just another option to disable directory renames again
  - lesson learned: when you actually know the semantics (POSIX in this case), be systematic, write many, many test
NO TESTS / NO PROBLEMS

BUG#4
FILE ID’S
two clients using an application to work on a project file on the server

occasionally one client saves and the file is gone

Locking? No. Oplocks? No. **FileIDs**!

**FileID**: Number that uniquely identifies a file (or directory):

returned as part of file metadata in FIND or GETINFO requests

OS X calls it **CNID** (Catalog Node ID)
What’s the problem?

- HFS+ doesn’t reuse **CNIDs** over the lifetime of a filesystem
- Internal OS X file lookup use **CNID/FileID** as primary key
- Samba uses filesystem inode number
- Inode numbers are reused
- Do you see the problem?
network trace showed that the saving client deleted the original file at the beginning after querying its FileID

saving involved three steps: save to a temp file, remove the original file, finally rename temp file to original name

client got confused by the previous save that changed the FileID

good news: workaround available, client can be configured not to trust FileIDs from the server

problem also seem to be fixed in latest OS X release

lesson learned: some things you just can’t test, can you?
Bugs found and fixed:

#1: Copying a directory to server
#2: Resource fork
#3: Rename behaviour
#4: FileIDs

Do we have tests now so we won’t break again?
$ make test TESTS=fruit
...
[1(0)/2 at 0s] samba3.vfs.fruit(nt4_dc)
[2(7)/2 at 5s] samba3.vfs.fruit(ad_dc)

ALL OK (14 tests in 2 testsuites)
...
$
$ make test TESTS=fruit
...
[1(0)/2 at 0s] samba3.vfs.fruit(nt4_dc)
[2(19)/2 at 5s] samba3.vfs.fruit(ad_dc)

ALL OK (38 tests in 2 testsuites)
...
$
19 tests, 14 test covering OS X spec deviations
0 tests cover Spotlight

tl;dr: we need tests, tests, tests!
THANK YOU!

QUESTIONS?

Ralph Böhme <slow@samba.org>