Opening Windows to a Wider World

# New Features in Samba 3.2

Jeremy Allison
jra@samba.org

# Covered in this talk

- There are many, many, new features in Samba 3.2, added by all members of the Team.
- I'm just going to cover the ones I added:
  - Removal of static buffers (pstrings).
  - SMB level encryption.
  - IPv6 conversion.
  - Large SMB transport / recvfile support.

# Death to pstrings ! (a refactoring bed-time story)

- What was a pstring ?
  - pstring is a data type in all versions of Samba from initial creation to 3.0.x.
  - pstring (Path String) represents a pathname internal to the Samba code.
  - Is a *fixed size* type, set to 1024 bytes very early in Samba creation.
  - Is very efficient (often pushed on the stack frame of a function) and very fast.

# Why must pstrings die ?

- pstrings don't fit the modern world.
  - 1024 bytes is a reasonable maximum path name limit, but people don't like limits.
  - Users are starting to report errors with paths too long.
- utf8 encodings make 1024 bytes much too small.
  - Theoretically a pathname using 6-byte utf8 encodings reduces the limit to 170 characters.

# “Blame Volker”

- Volker started the work by moving to dynamically allocated pathnames on many of the packet in/out code paths.
- Removing limits on pathnames looked easy, until we examined all the other places where pstrings interacted with the rest of the code (utility functions etc.).
- Conclusion was the only safe way to fix this was to eliminate pstrings entirely.
  - At this point Volker handed it over to me :-).

# How to replace pstrings ?

- An initial test was to remove the pstring data type definition from the include files.
  - Examine the number of errors..
  - **Way** too large to cope with in one commit ! pstring used in 1432 places in the code.
- Removal in stages instead.
  - Many dependencies on other code had to be examined. Many functions took a 'char *' parameter where they were actually assuming a pstring (1024 bytes of space).
  - Fixing this took a while..

# Malloc madness

- With no static pstrings Samba performance becomes very dependent on malloc (Volker added a clever hack to help with talloc).
- For vendors packaging Samba, linking with Google's tcmalloc gains performance (still working on the numbers).
- For non-threaded link instances of Samba, tridge's alloc_mmap is the fastest malloc.

# Hey – you forgot fstrings..

- Final commit message was here :

```
A requiem for pstring.

                    ______________
                   /              \
                  /                \
                 /      REST       \
                /        IN        \
               /        PEACE       \
              /                      \
              |  The infamous pstring  |
              |                        |
              |                        |
              |      7 December        |
              |                        |
              |         2007           |
             *|     *  *  *            | *
    _________)/\\_//(\/(/\)/\//\/\///|_)_______
```

- Removing fstrings is left as an exercise for the reader..

# SMB level encryption

- SMB will sign packets, but does not do encryption of packet data.
- With the ability of the UNIX extensions to add new calls to SMB, Steve French and I decided to add packet level encryption.
  - Goal was not to modify any existing calls – stay within the UNIX extension trans2 space.
  - NFSv3/4 has this (kerborized NFS). We need this to compete.

# Desired semantics

- This should be at a level below the creation / parsing of an SMB packet.
  - Keep the layer separate, allow SMB signing to continue as-is.
- Use “standards-based” encryption – GSSAPI / SSPI framing of packets.
- Allow “secure” shares to be marked “encrypt only” - whilst other shares are non encrypted.
- Don't require kerberos setup for small / home networks.

# Encrypted packet framing

- Standard SMB packets start with the signature 0xFF 'S' 'M' 'B'.
- Encrypted packets start with 0xFF 'E' <2 byte encryption context in little endian order>
  - Allows server to set up multiple separate encryption contexts for a single client.
  - Samba 3.2 server doesn't do this yet (always uses a context of zero).
- Standard packets are passed down to the encryption layer and gss_wrap() / gss_unwrap() or NTLM equivalents are used to sign and seal the packet.
  - Packet length can change at this layer.

# Setting up the encryption context.

- No changes to sessionsetupX allowed.
- UNIX extensions allow trans2 changes.
  - Client does a tconX to the share.
  - Client detects server is capable of doing encryption on this share via a capability bit returned in QFSinfo. A second capability bit specifies mandatory encryption.
  - Client then calls a trans2 SetFSinfo with SMB_REQUEST_TRANSPORT_ENCRYPTION (0x203). Associated data is a SPNEGO security negotiate (same as sessionsetup).

# Encryption contexts continued

- Just like sessionsetupX, client continues trans2 SetFSinfo calls until a successful authentication or returns NT_STATUS_ACCESS_DENIED.
  - Final reply is unencrypted, contains as reply params the 16-bit encryption context to use for this tree id.
  - Any subsequent client requests on this tid must be encrypted, or the server will reject them.
  - Client can re-key at any time by doing another SetFSinfo.

# A warning on security

- For userspace implementations (no shared file cache) the sessionsetupX user credentials can be re-used.
  - smbclient does this.
- For kernel (CIFSFS, Apple smbfs) implementations in a domain the machine account **must** be used to ensure proper security.
  - Buffer-cache poisoning.

# A small demo..

- Please don't crash.... please don't crash.... please don't crash.....

# IPv6 Conversion

- Lots of internal code conversion needed from the 3.0.x code base.
- Main change was from IPv4 specific internal structures and API calls to protocol independent structures and API calls.
  - Still works on IPv4 – only machines.
  - Mapping library created that implements the protocol independent calls on top of older IPv4 calls.
  - nmbd and all NetBIOS naming ignores IPv6 and binds to IPv4 only interfaces.

# IPv6 Conversion

- Please go to David Holder's talk to understand the gory details of IPv6.
- Modern Linux distributions start out of the box with active IPv6 interfaces.
- Samba now has interface code that will detect and bind to IPv6 interfaces.
  - smbclient can even use link-local addresses.
- Hosts allow/deny can be Ipv6.
- Bottom line is Samba 3.2 is fully IPv6 enabled – we can bid on US federal contracts :-).

# Large SMB frames

- NetBIOS over TCP frames are limited to 17 bits (128k).
  - But 7 bits are marked as “unused”.
  - So Steve Franch and I used them..
- Gives a frame size of 16 Mb.
- Samba can now easily send 16Mb return frames in an SMBreadX reply.
  - sendfile() makes this zero-copy.
- Accepting SMBwriteX calls of 16 Mb was a little harder.

# Large SMB frames continued.

- smbd now reads the SMB header first, then recognizes an incoming SMBwriteX call.
  - Calls down to a new VFS call, recvfile() which on some platforms can do a zero-copy write from the TCP stack into the buffer cache.
  - On platforms without recvfile(), smbd fakes the call by doing 128k staging reads from the socket into memory, then to the disk.
  - Strangely enough this makes the Linux client go fast, even without kernel support.

# Large SMB frames continued.

- Client support for this has been added in smbclient and libsmbclient code.
  - “Direct” reads and writes now remove one of the memcpy calls from incoming / outgoing data.
  - Should make Gnome nautilus file manager faster when doing network transfers.
- Does not work with new encrypted SMB support, or with SMB signed connections.
  - Complete buffer must be assembled in memory to sign the packet.

# Questions and Comments ?

Email: jra@samba.org

Technical support mailing list : samba@samba.org

Developer mailing list: samba-technical@samba.org