SMB2 Testing

Andrew Tridgell
Samba Team

The problem

- A monoculture of applications
  - Most software vendors tend to test only against Microsoft servers
  - How to leverage that testing?
- Needs deep comparitive testing
  - Don't want surprise differences in behaviour
  - Differences are fine, but only if carefully reviewed and tested

File system abstraction

- SMB/SMB2 is a window on the filesystem
  - A very wide window!
  - Nearly all of the details of the filesystem are exposed
- Very different from NFS
  - In NFS, filesystem details are mainly hidden
  - Software vendors targetting NFS servers test with multiple servers

Testing Approches

- There are lots of approaches
  - Some commonly used, some less common
- Main approaches used in Samba
  - Traditional Tests
  - 'RAW' tests
  - 'Full Coverage' Tests
  - Protocol Scanners
  - Stochastic + Model Tests
  - Dual-server Tests

Client Libraries

- Flexible, complete
  - The key to successful testing
  - Exposes all fields of the protocol
  - Structure oriented
  - Pad and reserved fields exposed
  - Async design
  - Auto-generated for some protocols (eg. RPC/IDL)

Structure Oriented Interface

```
struct smb2_request *smb2_close_send(struct smb2_tree *tree, struct smb2_close *io);
NTSTATUS smb2_close_recv(struct smb2_request *req, struct smb2_close *io);
NTSTATUS smb2_close(struct smb2_tree *tree, struct smb2_close *io);
```

```
struct smb2_close {
        enum smb_close_level level;
        struct {
                union smb_handle file;
                uint16_t flags;
                uint32_t _pad;
        } in;
        struct {
                uint16_t flags;
                uint32_t _pad;
                NTTIME   create_time;
                NTTIME   access_time;
                NTTIME   write_time;
                NTTIME   change_time;
                uint64_t alloc_size;
                uint64_t size;
                uint32_t file_attr;
        } out;
}
```

'RAW' Tests

- A high coverage conventional test
  - Tests each field of each operation
  - Uses internal redundency of protocol for cross-checking
  - Doesn't try many combinations of parameters
- Example
  - RAW-OPEN, RAW-QFILEINFO etc

Full Coverage Tests

- Enumerate all possibilities
  - Work in a narrow domain
  - Try all bits, or all info levels, or all operations
- Examples
  - All file_attribute bits (32 bits)
  - All desired_access bits (32 bits)

Protocol Scanners

- Discover availalable operations
  - Info levels
  - Opcodes
  - Looks for changes in error return
  - Doesn't need to send valid requests
- Examples
  - SMB2 opcode scanner
  - RPC scanners
- Largely obsolete
  - Rarely needed now that WSPP specifications are available

Stochastic + Model Tests

- Model a subsystem
  - Simple model of a small piece of semantics
- Random Exploration
  - Try random inputs
  - Compare model to target server
  - May allow control of parameter space
- Examples
  - Wildcard testing
  - NT share mode testing

Dual Server Testing

- “Protocol Differencing”
  - Compare behaviour of two servers for a protocol
  - Constrained random parameter generation
  - Automatic backtracking for minimal test case when a difference is found
  - Simple control over ignoring fields, parameters and opcodes
- Example
  - locktest (byte range locking)
  - gentest (general SMB/SMB2 testing)
- Tutorial (flash video)
  - See http://samba.org/~tridge/samba_testing/

Testable model

- Now we have a SMB2 spec ...
  - But is it any good?
  - Easy for errors to creep in – hard to find them
- Testing needs to cross subsystem boundaries
  - Filesystem layer, security layers, SMB2 layer etc
- The only way forward
  - We need a fully mountable implementation built solely from the spec
  - Needs to be tested against the windows server implementation using dual server testing