Exploring the SMB2 protocol

Andrew Tridgell
Samba Team
What is SMB2?

• An old acquaintance in new packaging
  • Introduced in Vista pre-releases
  • Uses same port number as SMB/CIFS
  • new packet formats
  • new negotiation approach
  • same underlying semantics
Major SMB2 features

- Larger limits
  - expanded range of many protocol elements
  - 16 byte handles used throughout
  - command words replaced by variable length header
  - same NBT style encapsulation (4 byte packet length)
  - All strings are UTF-16

- Possible features?
  - reportedly supports filesystem transactions
  - reportedly supports arbitrary chaining
SMB2 header

- The header is quite different from SMB
  - new 0xFE 'S' 'M' 'B protocol marker
  - header length 64
  - 32 bit NTSTATUS code
  - 16 bit opcode
  - 32 bit flags field
  - 64 bit sequence number
  - 32 bit PID
  - 32 bit TID
  - 64 bit VUID
  - 16 byte signature field
Opcode parameters

- Each opcode has a parameter block
  - replaces VWV command words from SMB
  - 16 bit length field
  - 1 bit of length reserved for 'dynamic' flag, indicating a dynamic buffer, equivalent to the SMB byte buffer
  - Encoding of parameter block very similar to equivalent encoding of VWV in SMB
  - Size of dynamic part implied by packet size
SMB2 (Server Message Block Protocol version 2)

SMB Header

Server Component: SMB2
Header Length: 64
NT Status: STATUS_SUCCESS (0x00000000)
Command: Create (5)
unknown: 0000
Flags: 0x00000000
...... .... .... .... .... .... 0 = Signing: This pdu is
...... .... .... .... .... .... 0 = PID Valid: The pid fi
...... .... .... .... .... .... 0 = Response: This is a f
unknown: 00000000
Command Sequence Number: 4
Process Id: 00000000 (not valid)

Tree Id: 1 \\vista4\test
User Id: 0x0000000000000001 Acct:tridge Domain:BLUDOM Host:BLU
Signature: 000000000000000000000000000000000000000000

Create Request (0x05)
Length: 56
...... .... .... .... 1 = Dynamic Part: True
Create Flags: 0x0000
Impersonation: Anonymous (0)
unknown: 0000000000000000
unknown: 0000000000000000
Access Mask: 0x001f01ff
File Attributes: 0x00000080
Share Access: 0x00000007
Disposition: Open If (if file exists open it, else create it) (3)
Create Options: 0x00000002
Filename: test9.dat
ExtraInfo MxAc

SMB (Server Message Block Protocol)

SMB Header

Server Component: SMB
[Response in: 27]
SMB Command: NT Create AndX (0xa2)
NT Status: STATUS_SUCCESS (0x00000000)

Flags: 0x08
Flags2: 0xc803
Process ID High: 0
Signature: 000000000000000000000000
Reserved: 0000

Tree ID: 2048
Process ID: 26266
User ID: 2048
Multiplex ID: 8

NT Create AndX Request (0xa2)
Word Count (WCT): 24
AndXCommand: No further commands (0xff)
Reserved: 00
AndXOffset: 0
Reserved: 00
File Name Len: 60

Create Flags: 0x00000010
Root FID: 0x00000000
Access Mask: 0x0000201f
Allocation Size: 0
File Attributes: 0x00000080
Share Access: 0x00000003
Disposition: Create (if file exists fail, else create it) (2)
Create Options: 0x00000040
Impersonation: Impersonation (2)
Security Flags: 0x03
Byte Count (BCC): 63
File Name: \rawpen\torture_ntcreatanx.txt
<table>
<thead>
<tr>
<th>SMB2 OpCode</th>
<th>Value</th>
<th>SMB2 OpCode</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMB2_OP_NEGPROT</td>
<td>0x00</td>
<td>SMB2_OP_LOCK</td>
<td>0x0a</td>
</tr>
<tr>
<td>SMB2_OP_SESSSETUP</td>
<td>0x01</td>
<td>SMB2_OP_IOCTL</td>
<td>0x0b</td>
</tr>
<tr>
<td>SMB2_OP_LOGOFF</td>
<td>0x02</td>
<td>SMB2_OP_CANCEL</td>
<td>0x0c</td>
</tr>
<tr>
<td>SMB2_OP_TCON</td>
<td>0x03</td>
<td>SMB2_OP_KEEPALIVE</td>
<td>0x0d</td>
</tr>
<tr>
<td>SMB2_OP_TDIS</td>
<td>0x04</td>
<td>SMB2_OP_FIND</td>
<td>0x0e</td>
</tr>
<tr>
<td>SMB2_OP_CREATE</td>
<td>0x05</td>
<td>SMB2_OP_NOTIFY</td>
<td>0x0f</td>
</tr>
<tr>
<td>SMB2_OP_CLOSE</td>
<td>0x06</td>
<td>SMB2_OP_GETINFO</td>
<td>0x10</td>
</tr>
<tr>
<td>SMB2_OP_FLUSH</td>
<td>0x07</td>
<td>SMB2_OP_SETINFO</td>
<td>0x11</td>
</tr>
<tr>
<td>SMB2_OP_READ</td>
<td>0x08</td>
<td>SMB2_OP_BREAK</td>
<td>0x12</td>
</tr>
<tr>
<td>SMB2_OP_WRITE</td>
<td>0x09</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Q: Who can see the obvious omissions?
Path operations

- Handle oriented operation
  - Only one pathname operation – SMB2_OP_CREATE
  - unlink, rmdir, qfileinfo and find are all done via open handles (16 bytes each)
  - this continues a trend seen in recent windows releases
  - this makes the protocol logically simpler, especially for access checks
  - number of network round trips for common operations is increased
  - we expect chaining will be used to compensate in the future
SMB2 in Samba

- We have put quite a large effort into SMB2 in Samba4
  - implemented a quite comprehensive client library
  - implemented a test suite – not comprehensive yet
  - implemented RPC over SMB2
  - implemented a SMB2 server (not complete)
  - in cooperation with Ronnie Sahlberg, implemented a wireshark (ethereal) protocol analyser
SMB2 Test Suite

- smbtorture and SMB2
  - smbtorture now includes 10 SMB2 tests
  - Tests concentrate on protocol exploration and scanning

  SMB2-CONNECT
  SMB2-FIND
  SMB2-GETINFO
  SMB2-LOCK
  SMB2-NOTIFY
  SMB2-SCAN
  SMB2-SCANFIND
  SMB2-SCANGETINFO
  SMB2-SCANSETINFO
  SMB2-SETINFO
More Information and Credit

• Wireshark
  • Wireshark (formerly ethereal) has good support for SMB2 decoding, thanks to efforts of Ronnie Sahlberg
  • Samba4 sources, in particular libcli/smb2/ and libcli/raw/

• Credit
  • Many thanks for the efforts of Stefan Metzmacher, Volker Lendecke, Steve French, Ronnie Sahlberg and Andrew Bartlett for their great work on SMB2