SDC 18

September 24-27, 2018
Santa Clara, CA

www.storagedeveloper.org

A New Open Source SMB2/3 Server Running on Windows in User Mode!

James Westland Cain, Ph.D
Grass Valley, A Belden Brand

Version 0.5.5

Introduction – James Westland Cain

- Principal Architect – Software @ Grass Valley
- I code every day – as well as being arm waver in chief!
- Been coding for nearly 40 years, at GV for nearly 20!
- My research interests include file systems innovation and modern video production.
- PhD in Advanced Software Engineering from Reading University
- Visiting Research Fellow at Brunel University

My Previous SDC Presentations

- 2010: RESTful Filsystems: https://www.snia.org/sites/default/orig/sdc_archives/2010_presentations/thursday/JamesCain_RESTful_Filesystems.pdf
- 2011: Hidden Gems in the NAS Protocols: URL Missing
- 2013: Exploiting the High Availability features in SMB 3.0 to support Speed and Scale
- https://www.snia.org/sites/default/files/files2/files2/SDC2013/SMB3/JamesCain_Exploiting_High_Availability_SMB30%20to%20support-v1.pdf
- 2015: A Pausable File System:
- http://www.snia.org/sites/default/files/SDC15_presentations/file_sys/JamesCain_A_Pausable_File_System.pdf
- 2017: Programming the Path:
- https://www.snia.org/sites/default/files/SDC/2017/presentations/File_Systems/Cain_James_Westland_Programming_the_Path.pdf

Agenda I

- Introduction, Rationale & Competition
- Code tour
  - Architecture
  - Building
  - Running
  - Plugins

Agenda II

- What is working
- Research Topics
- Debts & To Dos
- Ideas for projects

Learning Objectives I

- Learn how elegant and simple the SMB2/3 protocol really is
- Learn how hard it is to develop an SMB2/3 server.
  - Hint: we provide solutions for (some of) the hard bits ☺

Learning Objectives II

- ❒ Learn how much of Windows can be re-used to develop an SMB2/3 server
- ❒ Get some ideas of what you can do with a user mode file system for your own projects

Introduction

- This talk is about writing an SMB2/3 server.
- The server runs on Windows.
  - This has some interesting benefits
- The server runs in user mode.
  - This also has some interesting benefits
- The server is now open source
  - You can play too ☺

Rationale

- Why implement SMB2/3 on Windows?
  - I have data I’d like to offer as files
    - The data is within the Windows eco-system.
  - Writing Kernel mode filesystems is notoriously hard
    - This approach is easier – honestly!
  - It’s a NAS not a DAS!
    - Having direct access to the SMB server offers many interesting benefits for filesystems innovation

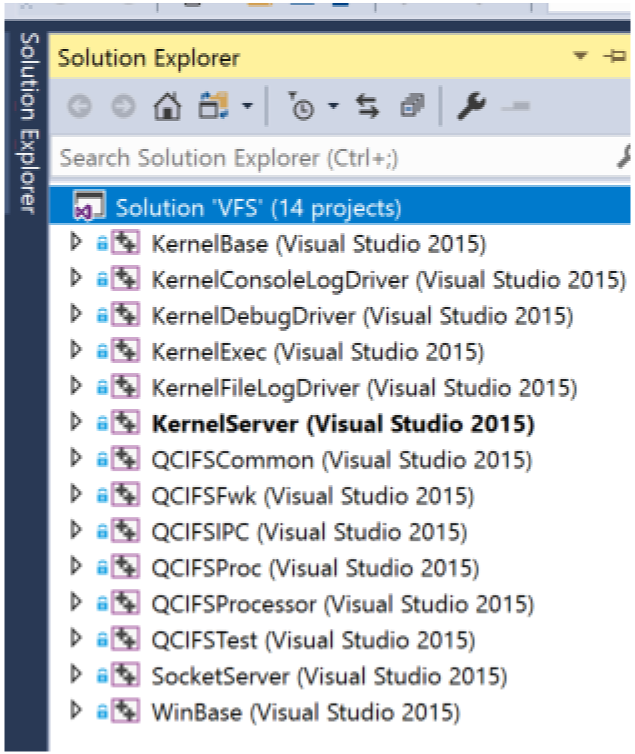
Fuse Example Uses (from Wikipedia)

- archivemount
- CloudStore (formerly, Kosmos filesystem): By mounting via FUSE, existing Linux utilities can interact with CloudStore
- EncFS: Encrypted virtual filesystem
- ExpanDrive: A commercial filesystem implementing SFTP/FTP/S3/Swift using FUSE
- FTPFS
- GlusterFS: Clustered Distributed Filesystem having ability to scale up to several petabytes.
- GmailFS: Filesystem which stores data as mail in Gmail
- GVfs: The virtual filesystem for the GNOME desktop
- KBFS: A distributed filesystem with end-to-end encryption and a global namespace based on Keybase.io service that uses FUSE to create cryptographically secure file mounts.
- Lustre cluster filesystem will use FUSE to allow it to run in userspace, so that a FreeBSD port is possible.[8] However, the ZFS-Linux port of Lustre will be running ZFS's DMU (Data Management Unit) in userspace.[9]
- MinFS: MinFS is a fuse driver for Amazon S3 compatible object storage server. MinFS[10] lets you mount a remote bucket (from a S3 compatible object store), as if it were a local directory
- MooseFS: An open source distributed fault-tolerant file system available on every OS with FUSE implementation (Linux, FreeBSD, NetBSD, OpenSolaris, OS X), able to store petabytes of data spread over several servers visible as one resource.
- NTFS-3G and Captive NTFS, allowing access to NTFS filesystems
- Sector File System: Sector is a distributed file system designed for large amount of commodity computers. Sector uses FUSE to provide a mountable local file system interface
- SSHFS: Provides access to a remote filesystem through SSH
- Transmit: A commercial FTP client that also adds the ability to mount WebDAV, SFTP, FTP and Amazon S3 servers as disks in Finder, via MacFUSE.
- WebDrive: A commercial filesystem implementing WebDAV, SFTP, FTP, FTPS and Amazon S3
- WikipediaFS: View and edit Wikipedia articles as if they were real files
- Wuala: A multi-platform, Java-based fully OS integrated distributed file system. Using FUSE, MacFUSE and Callback File System respectively for file system integration, in addition to a Java-based app accessible from any Java-enabled web browser (service discontinued in 2015).

Competition (on Windows)

- Commercial Fuse like offerings on Windows:
  - https://www.callbacktechnologies.com/cbfsconnect/
- Open Source Fuse like offerings on Windows:
  - https://github.com/dokan-dev/dokany
  - https://github.com/billziss-gh/winfsp
- All use Kernel Mode drivers
  - This is hard!
- They are local file systems
  - SMB2/3 is a network protocol …

The Code – Architecture I

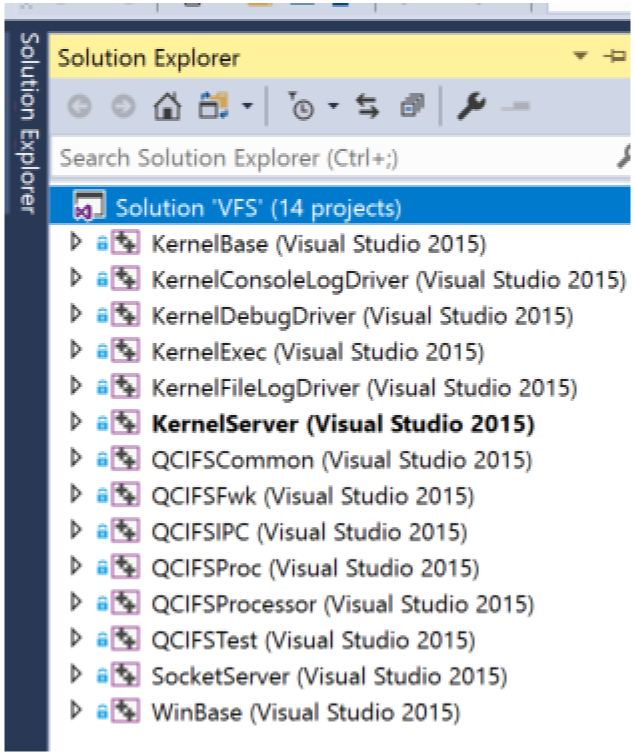

Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'VFS' (14 projects)
- KernelBase (Visual Studio 2015)
- KernelConsoleLogDriver (Visual Studio 2015)
- KernelDebugDriver (Visual Studio 2015)
- KernelExec (Visual Studio 2015)
- KernelFileLogDriver (Visual Studio 2015)
- **KernelServer (Visual Studio 2015)**
- QCIFSCommon (Visual Studio 2015)
- QCIFSFwk (Visual Studio 2015)
- QCIFSIPC (Visual Studio 2015)
- QCIFSProc (Visual Studio 2015)
- QCIFSProcessor (Visual Studio 2015)
- QCIFSTest (Visual Studio 2015)
- SocketServer (Visual Studio 2015)
- WinBase (Visual Studio 2015)

- Approx 80k Lines of code
- 14 projects that build many DLLs, a LIB and one EXE that loads the DLLs.
- Taken from a *much* bigger project that is millions of lines of code.

The Code – Architecture II



Solution 'VFS' (14 projects)
- KernelBase (Visual Studio 2015)
- KernelConsoleLogDriver (Visual Studio 2015)
- KernelDebugDriver (Visual Studio 2015)
- KernelExec (Visual Studio 2015)
- KernelFileLogDriver (Visual Studio 2015)
- **KernelServer (Visual Studio 2015)**
- QCIFSCommon (Visual Studio 2015)
- QCIFSFwk (Visual Studio 2015)
- QCIFSIPC (Visual Studio 2015)
- QCIFSProc (Visual Studio 2015)
- QCIFSProcessor (Visual Studio 2015)
- QCIFSTest (Visual Studio 2015)
- SocketServer (Visual Studio 2015)
- WinBase (Visual Studio 2015)

- Written in C++ (pre-modern, so own smart pointers etc)
- Uses Windows SDK and The Boost C++ libraries.
- Builds to 64 bit Windows binaries.

The Code – Architecture III

- The Kernel{*} projects have no knowledge of SMB2/3
- They offer useful services that support the layers above
  - KernelServer is an exe that loads the DLLs.
  - KernelServer calls `moduleInit` on each DLL.
  - The DebugDiver catches GPFs and logs call stacks
  - There are two logging DLLs
    - A Console and A File Based Logger.
    - By default, file based logs are written to c:\data\logs

The Code – Architecture IV

- The QCIFS{*} Dlls implement the SMB2/3 server code and offer some simple example file systems to play with.
- QCIFSTest is a simple RAM only file and folder system.
  - QCIFSTest is a good place to start.

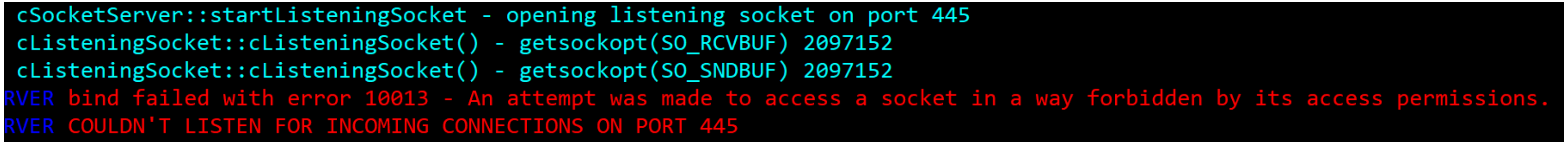
Building

- README.md is in the root folder – start here.
  - Get Visual Studio (2015 or better),
  - Get a modern version of Boost (1.65.1 or better),
  - Configure a path to boost in the props file
  - Build a release build

Running I

- How does it work?
  - It’s just a Network Server offering a well defined protocol.
  - We bind to port 445 and offer everything that an SMB2/3 server is expected to offer.
- Run “RUNME.bat”

Freeing port 445



```
 cSocketServer::startListeningSocket - opening listening socket on port 445
 cListeningSocket::cListeningSocket() - getsockopt(SO_RCVBUF) 2097152
 cListeningSocket::cListeningSocket() - getsockopt(SO_SNDBUF) 2097152
RVER bind failed with error 10013 - An attempt was made to access a socket in a way forbidden by its access permissions.
RVER COULDN'T LISTEN FOR INCOMING CONNECTIONS ON PORT 445
```

- We need to free port 445 in order to bind to it!
- Option 1: kill Server Service!
- Option 2: (less nuclear) unbind the Server Service from port 445.

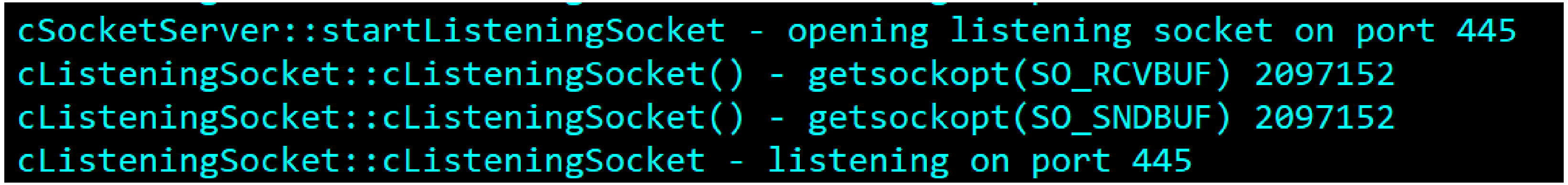
Free445

- Use NetServerTransportEnum to find
  - "\\Device\\NetbiosSmb"
- Use NetServerTransportDel to delete it.
  - This requires elevated privileges.
- This gets re-set on the next windows reboot.
- Free445 is a small project that does this.

NetManService

- A more elegant way is to install a service, that can be run up each time we run KernelServer.
- The NetManService does this.
  - It runs the same code as Free445
  - KernelServer does not normally need elevated privileges to run.
    - KernelServer only needs to be elevated when it installs the service - on first boot.

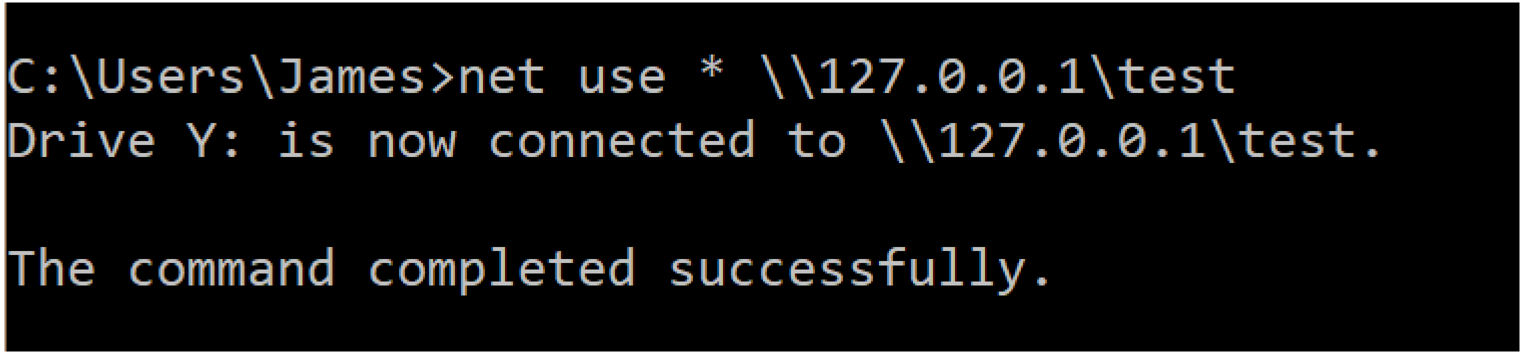
Running II



```
cSocketServer::startListeningSocket - opening listening socket on port 445
cListeningSocket::cListeningSocket() - getsockopt(SO_RCVBUF) 2097152
cListeningSocket::cListeningSocket() - getsockopt(SO_SNDBUF) 2097152
cListeningSocket::cListeningSocket - listening on port 445
```

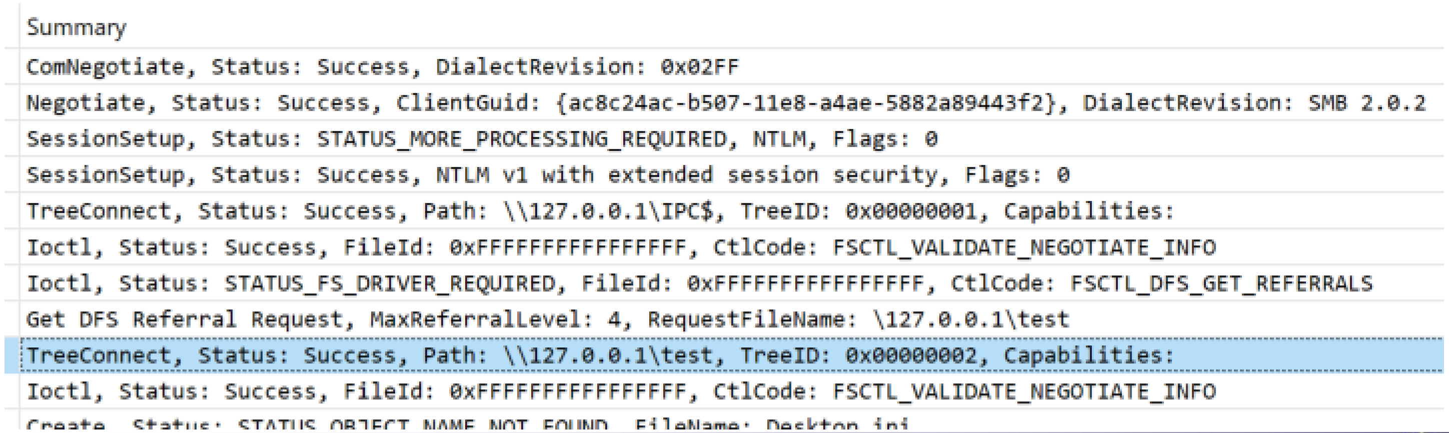
- Now we can bind, so we can listen for TCP connections on port 445.

Loopback connection to the SMB2/3 server



```
C:\Users\James>net use * \\127.0.0.1\test
Drive Y: is now connected to \\127.0.0.1\test.

The command completed successfully.
```



```
Summary
ComNegotiate, Status: Success, DialectRevision: 0x02FF
Negotiate, Status: Success, ClientGuid: {ac8c24ac-b507-11e8-a4ae-5882a89443f2}, DialectRevision: SMB 2.0.2
SessionSetup, Status: STATUS_MORE_PROCESSING_REQUIRED, NTLM, Flags: 0
SessionSetup, Status: Success, NTLM v1 with extended session security, Flags: 0
TreeConnect, Status: Success, Path: \\127.0.0.1\IPC$, TreeID: 0x00000001, Capabilities:
Ioctl, Status: Success, FileId: 0xFFFFFFFFFFFFFFFF, CtlCode: FSCTL_VALIDATE_NEGOTIATE_INFO
Ioctl, Status: STATUS_FS_DRIVER_REQUIRED, FileId: 0xFFFFFFFFFFFFFFFF, CtlCode: FSCTL_DFS_GET_REFERRALS
Get DFS Referral Request, MaxReferralLevel: 4, RequestFileName: \127.0.0.1\test
TreeConnect, Status: Success, Path: \\127.0.0.1\test, TreeID: 0x00000002, Capabilities:
Ioctl, Status: Success, FileId: 0xFFFFFFFFFFFFFFFF, CtlCode: FSCTL_VALIDATE_NEGOTIATE_INFO
Create, Status: STATUS_OBJECT_NAME_NOT_FOUND, FileName: Desktop.ini
```

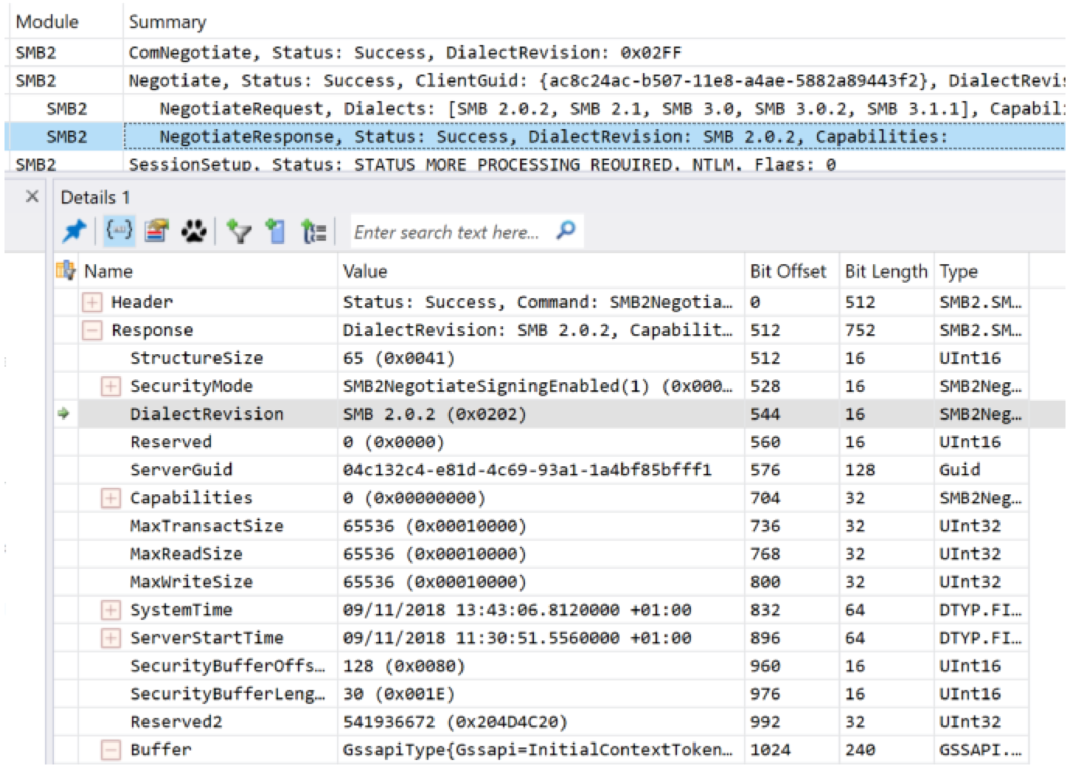
Code to Handle SMBs

- Project: QCIFSProcessor
  - (The name is a bit old now …)
- Requests: cSMB2Request.cpp
- Sessions: cSMB2Session.cpp
- Responses: cSMB2Response.cpp

SMB Dispatch & Efficient Routing

- cQCIFSPacketProcessor::AddPacket receives complete SMB packets from the wire.
- Stack based cSMB2Request wrapped around wire payload.
- cSMB2Request::dispatchCommand routes to one of 19 SMB handlers
- cSMB2Request::getResponses uses pre-allocated buffers wrapped by cSMB2Response that formats bytes to go on the wire.
- There are no memcpy calls in the pipeline
- Use of Win32 IOCompletionPort and TransmitPackets API supports high speeds with low overhead. See SocketServer project.

Protocol Version Negotiation

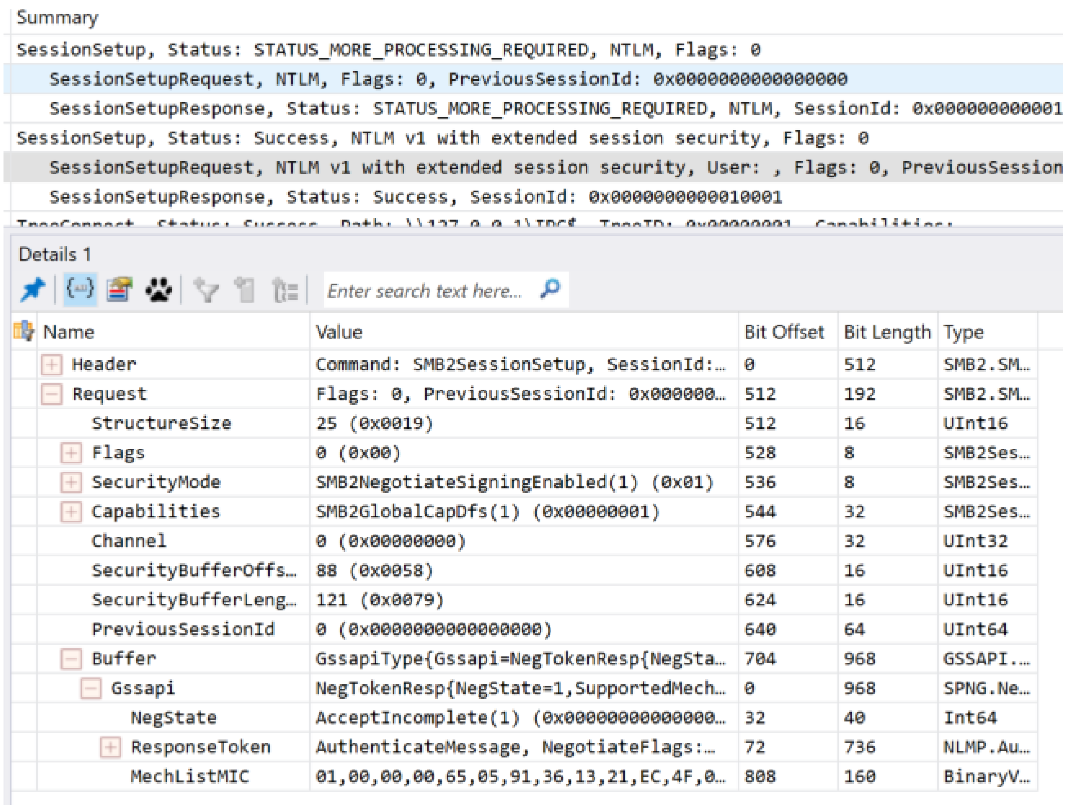


Module	Summary
SMB2	ComNegotiate, Status: Success, DialectRevision: 0x02FF
SMB2	Negotiate, Status: Success, ClientGuid: {ac8c24ac-b507-11e8-a4ae-5882a89443f2}, DialectRevi
SMB2	NegotiateRequest, Dialects: [SMB 2.0.2, SMB 2.1, SMB 3.0, SMB 3.0.2, SMB 3.1.1], Capabil
SMB2	NegotiateResponse, Status: Success, DialectRevision: SMB 2.0.2, Capabilities:
SMB2	SessionSetup, Status: STATUS_MORE_PROCESSING_REQUIRED, NTLM, Flags: 0

Name	Value	Bit Offset	Bit Length	Type
Header	Status: Success, Command: SMB2Negotia…	0	512	SMB2.SM…
Response	DialectRevision: SMB 2.0.2, Capabilit…	512	752	SMB2.SM…
StructureSize	65 (0x0041)	512	16	UInt16
SecurityMode	SMB2NegotiateSigningEnabled(1) (0x000…	528	16	SMB2Neg…
DialectRevision	SMB 2.0.2 (0x0202)	544	16	SMB2Neg…
Reserved	0 (0x0000)	560	16	UInt16
ServerGuid	04c132c4-e81d-4c69-93a1-1a4bf85bfff1	576	128	Guid
Capabilities	0 (0x00000000)	704	32	SMB2Neg…
MaxTransactSize	65536 (0x00010000)	736	32	UInt32
MaxReadSize	65536 (0x00010000)	768	32	UInt32
MaxWriteSize	65536 (0x00010000)	800	32	UInt32
SystemTime	09/11/2018 13:43:06.8120000 +01:00	832	64	DTYP.FI…
ServerStartTime	09/11/2018 11:30:51.5560000 +01:00	896	64	DTYP.FI…
SecurityBufferOffs…	128 (0x0080)	960	16	UInt16
SecurityBufferLeng…	30 (0x001E)	976	16	UInt16
Reserved2	541936672 (0x204D4C20)	992	32	UInt32
Buffer	GssapiType{Gssapi=InitialContextToken…	1024	240	GSSAPI.…

- We need to negotiate a protocol version.
- We choose the simplest (for now)

Session Setup



```
Summary
SessionSetup, Status: STATUS_MORE_PROCESSING_REQUIRED, NTLM, Flags: 0
    SessionSetupRequest, NTLM, Flags: 0, PreviousSessionId: 0x0000000000000000
    SessionSetupResponse, Status: STATUS_MORE_PROCESSING_REQUIRED, NTLM, SessionId: 0x000000000001
SessionSetup, Status: Success, NTLM v1 with extended session security, Flags: 0
    SessionSetupRequest, NTLM v1 with extended session security, User: , Flags: 0, PreviousSession
    SessionSetupResponse, Status: Success, SessionId: 0x0000000000010001
TreeConnect, Status: Success, Path: \\127.0.0.1\IPC$, TreeID: 0x00000001, Capabilities:
```

Details 1

Enter search text here...

Name	Value	Bit Offset	Bit Length	Type
Header	Command: SMB2SessionSetup, SessionId:...	0	512	SMB2.SM...
Request	Flags: 0, PreviousSessionId: 0x000000...	512	192	SMB2.SM...
StructureSize	25 (0x0019)	512	16	UInt16
Flags	0 (0x00)	528	8	SMB2Ses...
SecurityMode	SMB2NegotiateSigningEnabled(1) (0x01)	536	8	SMB2Ses...
Capabilities	SMB2GlobalCapDfs(1) (0x00000001)	544	32	SMB2Ses...
Channel	0 (0x00000000)	576	32	UInt32
SecurityBufferOffs...	88 (0x0058)	608	16	UInt16
SecurityBufferLeng...	121 (0x0079)	624	16	UInt16
PreviousSessionId	0 (0x0000000000000000)	640	64	UInt64
Buffer	GssapiType{Gssapi=NegTokenResp{NegSta...	704	968	GSSAPI....
Gssapi	NegTokenResp{NegState=1,SupportedMech...	0	968	SPNG.Ne...
NegState	AcceptIncomplete(1) (0x00000000000000...	32	40	Int64
ResponseToken	AuthenticateMessage, NegotiateFlags:...	72	736	NLMP.Au...
MechListMIC	01,00,00,00,65,05,91,36,13,21,EC,4F,0...	808	160	BinaryV...

- We then need to establish a secure session.
- This generates the sessionKey for signing the session payloads.

Session Key Generation I

- **Windows Advantage: Use SSPI API!**
- PackageName = "NTLM"
- QuerySecurityPackageInfo, AcquireCredentialsHandle, QueryContextAttributes, AcceptSecurityContext, CompleteAuthToken, DeleteSecurityContext, FreeContextBuffer, FreeCredentialsHandle.

Session Key Generation II

- **Windows Advantage: Use Bcrypt API!**
- BCRYPT_RC4_ALGORITHM, BCRYPT_MD5_ALGORITHM, BCRYPT_SHA256_ALGORITHM, BCRYPT_ALG_HANDLE_HMAC_FLAG, BCRYPT_AES_CMAC_ALGORITHM.

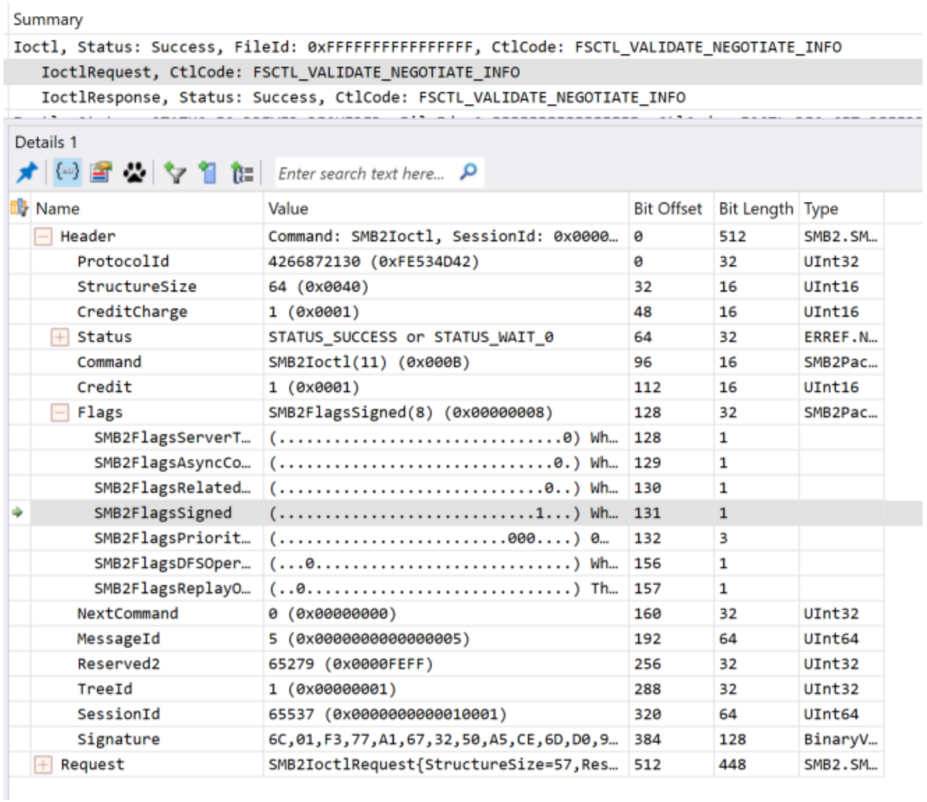
NTLM v2 Authentication - Algorithm

- NTOWFv2(Passwd, User, UserDom) HMAC_MD5(MD4(UNICODE(Passwd)), UNICODE(ConcatenationOf(Uppercase(User), UserDom)))
- "NTOWFv2("Password", "User", "Domain") is
  - 0c 86 8a 40 3b fd 7a 93 a3 00 1e f2 2e f0 2e 3f
- Taken from [MS-NLMP]: NT LAN Manager (NTLM) Authentication Protocol

NTLM v2 Authentication - Code

```
Buffer NTOWFv2(Buffer& password, Buffer& user, Buffer& domain) {
  Buffer md4Psswrd;
  hashVal(BCRYPT_MD4_ALGORITHM, password, md4Psswrd);
  Buffer userAndDom;
  NT_VERIFY(userAndDom.Create(user.GetSize() +
domain.GetSize()));
  memcpy(userAndDom.GetData(), user.GetData(), user.GetSize());
  memcpy(userAndDom.GetData() + user.GetSize(), domain.GetData(),
domain.GetSize());
  return HMAC_MD5(md4Psswrd, userAndDom);
}
```

Validate Negotiate Info



Summary

```
Ioctl, Status: Success, FileId: 0xFFFFFFFFFFFFFFFF, CtlCode: FSCTL_VALIDATE_NEGOTIATE_INFO
   IoctlRequest, CtlCode: FSCTL_VALIDATE_NEGOTIATE_INFO
   IoctlResponse, Status: Success, CtlCode: FSCTL_VALIDATE_NEGOTIATE_INFO
```

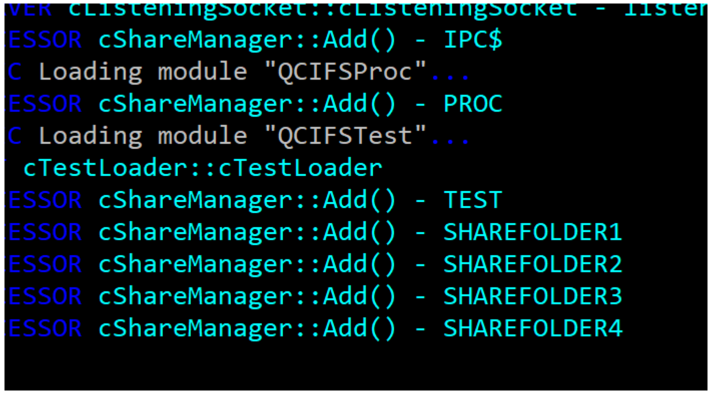
Details 1

Enter search text here...

Name	Value	Bit Offset	Bit Length	Type
Header	Command: SMB2Ioctl, SessionId: 0x0000...	0	512	SMB2.SM...
ProtocolId	4266872130 (0xFE534D42)	0	32	UInt32
StructureSize	64 (0x0040)	32	16	UInt16
CreditCharge	1 (0x0001)	48	16	UInt16
Status	STATUS_SUCCESS or STATUS_WAIT_0	64	32	ERREF.N...
Command	SMB2Ioctl(11) (0x000B)	96	16	SMB2Pac...
Credit	1 (0x0001)	112	16	UInt16
Flags	SMB2FlagsSigned(8) (0x00000008)	128	32	SMB2Pac...
SMB2FlagsServerT...	(...............................0) Wh...	128	1	
SMB2FlagsAsyncCo...	(..............................0.) Wh...	129	1	
SMB2FlagsRelated...	(.............................0..) Wh...	130	1	
SMB2FlagsSigned	(............................1...) Wh...	131	1	
SMB2FlagsPriorit...	(.........................000....) 0...	132	3	
SMB2FlagsDFSOper...	(...0............................) Wh...	156	1	
SMB2FlagsReplayO...	(..0.............................) Th...	157	1	
NextCommand	0 (0x00000000)	160	32	UInt32
MessageId	5 (0x0000000000000005)	192	64	UInt64
Reserved2	65279 (0x0000FEFF)	256	32	UInt32
TreeId	1 (0x00000001)	288	32	UInt32
SessionId	65537 (0x0000000000010001)	320	64	UInt64
Signature	6C,01,F3,77,A1,67,32,50,A5,CE,6D,D0,9...	384	128	BinaryV...
Request	SMB2IoctlRequest{StructureSize=57,Res...	512	448	SMB2.SM...

- A server must handle this FSCTL or the client will disconnect.
- The SMB is signed, so the reply must be signed too.
- Signing requires the session key.

VFS - Adding Shares I



```
VER cListeningSocket::cListeningSocket - listen
ESSOR cShareManager::Add() - IPC$
C Loading module "QCIFSProc"...
ESSOR cShareManager::Add() - PROC
C Loading module "QCIFSTest"...
 cTestLoader::cTestLoader
ESSOR cShareManager::Add() - TEST
ESSOR cShareManager::Add() - SHAREFOLDER1
ESSOR cShareManager::Add() - SHAREFOLDER2
ESSOR cShareManager::Add() - SHAREFOLDER3
ESSOR cShareManager::Add() - SHAREFOLDER4
```

- Class iQCIFSProcessor is a singleton (it registers using some KernelExec macros).
- iQCIFSProcessor::singleton().attachResource(..) is used to add new share names to the SMB server.

VFS - Adding Shares II

- QCIFSProcessor module receives and returns payload bytes to the SocketServer.
- iQCIFSFwk::singleton().createTreeResourceFactory() returns iTreeResourceFactory instances suitable to be passed to iQCIFSProcessor::attachResource().
- QCIFSFwk module offers lots of boiler plate code to support default implementations – enabling easy File System Development.

QCIFSTest – An Example File System I

- A very simple basic file system driver.
- It can offer readonly files from local disk and RAM only mutable files and folders.
- Its small …

```
|      extension|     total code|  total comment|    total blank|percent|
-------------------------------------------------------------------------
|           .cpp|            421|            169|            146|     47|
|             .h|            161|            171|             99|     18|
```

QCIFSTest – An Example File System II

- `createTreeResourceFactory` expects an `iChildLoader` instance.
- QCIFSTest implements a `class` **`cTestLoader`** `: public cChildLoader, public iCreate, public iRename, public cRefCount`
- Inherit from:
  - **`cChildLoader`** `– Base implementation that returns the children of this folder by implementing void` **`registerListener`**`(const vfs::cPtr<`**`iChildLoaderVisitor`**`> pChildListener)`
  - **`iCreate`** `– adds creation support for files and folders`
  - **`iRename`** `– adds rename support`
  - **`cRefCount`** `– mixin reference count to enable our smart pointer`

QCIFSTest – An Example File System III

- The `iChildLoaderVisitor` interface offers lots of ways to add files and folders to this folder, using various overloads on `updateFolder` and `updateFile`.
- Variations on types of folder functionality can be built using interfaces previously mentioned (`iChildLoader, iCreate, iRename, etc`).
- Variations on types of files can be built by inheriting from interfaces such as `iReadCallback` or `iWriteCallback`.

Example File System – Adding Folders

```
virtual void iChildLoaderVisitor::updateFolder(const vfs::String& sName
    , const vfs::cPtr<iChildLoader> pChildLoader
    , const vfs::cPtr<iRename> pRename
    , const vfs::cPtr<iCreate> pFolderCreate
    , const vfs::cConstPtr<vfs::cMemoryView> pIconMem
    , const vfs::cPtr<iFileEvent> pFileEvent = vfs::cPtr<iFileEvent>()
    , const bool bDeletable=false
    , const bool bNotify = true) = 0;
```

Example File System – Adding Files

```
virtual void iChildLoaderVisitor::updateFile(
    const vfs::String& sName
    , vfs::cPtr<iReadCallback> readCallback
    , const vfs::cPtr<iFileEvent> pFileEvent = vfs::cPtr<iFileEvent>()
    , bool bDeletable = true
    , const bool bNotify = true) = 0;

  virtual void iChildLoaderVisitor::updateFile(
    vfs::cPtr<iWriteCallback> writeCallback
    , const vfs::String& sName
    , bool bDeletable = true
    , const bool bNotify = true) = 0;
```

Example File System – Readonly Files

```
class iReadCallback {
public:
  virtual ~iReadCallback() {}
  virtual unsigned __int64 getSize(ULONGLONG fid) = 0;
  virtual DWORD readBytes(tTransmitList &krTPM, DWORD& nBytes, const
LARGE_INTEGER &nOffset, const int sessionID, ULONGLONG fid) = 0;
  virtual bool canExecute() { return false; }//defaults to no execution rights
  virtual bool canCache() { return true; }//defaults to SMB2 BATCH OPLOCK
  virtual DWORD registerChangeListener(vfs::cPtr<iCallbackChangeListener>
listener) = 0;
  virtual DWORD close(ULONGLONG fid) {return 0;}
};
```

Example File System – Mutable Files

```
class iWriteCallback : public iReadCallback {
public:
  virtual DWORD setSize(unsigned __int64 newSize) = 0;
  virtual DWORD writeBytes(vfs::cConstPtr<vfs::cMemoryView> buffer, const
LARGE_INTEGER &offset, const int sessionID) = 0;
};

class iFileEvent {
public:
  virtual ~iFileEvent(){}
  virtual DWORD notifyDelete() = 0;
};
```

Example file implementation classes

- `class` **`cBasicFile`** `: public` **`iReadCallback`**`, public cRefCount`
  - Takes a path to a local file and offers it to the VFS as a read only byte range.
- `class` **`cTestWriteCallback`** `: public` **`iWriteCallback`**`, public` **`iRename`**`, public` **`iFileEvent`**`, public vfs::cRefCount`
  - Created inside `cTestLoader::File`, added to `iChildLoaderVisitor` using `updateFile`.
- `cTestLoader::Directory` creates extra instances of `cTestLoader` to support sub directories.

It’s not just a DAS – it’s a NAS I

- The competition listed in the front matter were all using IFSKit style Kernel mode drivers to build loopback filesystems.
- Our VFS is a NAS protocol server.
- Its has been tested against numerous clients, including OS-X, Linux, Solaris and Windows.
  - No warranties though ☺

It’s not just a DAS – it’s a NAS II

- The real advantage of being a NAS server comes with more modern SMB3 features.
- SMB3 offers clustering for scale out and failover.
- The SMB3 protocol has been used in very interesting and demanding deployments, such as storage for booting Hyper-V images.

Protocol Research I

- Credits – see QCIFSProcessor Main.cpp
  - Defaults to 40 – plenty for loopback
  - Can be overridden using registry
    - See moduleInit in QCIFSProcessor project.

Protocol Research II

- Protocol Versions - #define kUSE_SMB3
  - Enables v3 negotiation
  - SMB3 uses different session key calculation.
- **SMB3 supports:** SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY, SMB2_SHARE_CAP_CLUSTER, SMB2_SHARE_CAP_SCALEOUT

Protocol Research III

- **SMB3 experiments:**
- #define kUSE_SMB_NETWORK_INTERFACE_INFO
  - Supports experiments with FSCTL_QUERY_NETWORK_INTERFACE_INFO

Debts – Signing & Encryption

- **Debt: Sign All The Time**
- Status: Single packets work.
- Multiple (compound) packets don’t.
  - Need to configure OS-X to work around
- **Debt: Encryption**
- Not supported

Things to add – Enum Shares I

- Support net share enumerations via ‘srvsvc’.
  - OS-X needs srvsvc to re-connect after an error / timeout etc & Finder needs it anyway.
  - On Windows loopback try command
    - **net view 127.0.0.1**
  - Use MIDL compiler to make C code for the srvsvc IDL and ms-dtyp.idl.

Things to add – Enum Shares II

- Approach: use local named pipe (so LPC) to our own srvsvc!
- Pump binary using read write IO.
  - Semantics: Transact is blocking, read & write is async.
  - Test approach using: NetShareEnum(NULL, 0, &buf, MAX_PREFERRED_LENGTH, &entries, &total, NULL);

Other Service Function Calls

- OX-X calls SRVS: BIND & NetShareEnum.
- Explorer calls SRVS: BIND & NetrShareEnum, NetrServerGetInfo, NetrShareGetInfo.
- Explorer calls WKST: BIND & NetrWkstaGetInfo.

Things to add – RIO?

- Complement IOCompletionPort with RIO
  - Registered I/O - API to support high-speed networking for increased networking performance with lower latency and jitter.
  - RIO relies on registering the memory that you will use as data buffers and knowing in advance how many pending operations a given socket will have at any time.

Project: Clustering

- The VFS code base supports some of the SMB3 extensions for multi-channel, scale out and clustering.
- I gave a talk about the semantics of the implied model between a set of SMB3 cluster members here in 2013
- It would be interesting to build a clustered VFS using this code base, as we can again exploit some of the Windows ecosystem to support this.

Project: SMB Direct – In User Mode ???

- SMB3 multi-channel enables different transports, like SMB Direct.
- NDKPI is an MS Kernel Mode API for RDMA access.
- The MS HPC team use NDSPI – as user mode equivalent to the Kernel Mode API.
  - NDSPI is very poorly documented, but its API is a mirror of NDKPI, which is well documented.
- Ports are either 445 for Infiniband, or 5445 if TCP ports are in use (this new number is registered with IANA).
- We can use the same NetServerTransportEnum techniques to free ports

Legal

- The code is open source.
- The code is Licensed under the Apache License, Version 2.0 …
- … so really … no WARRANTIES !
- See here for the full terms:
  - http://www.apache.org/licenses/LICENSE-2.0

Git Clone

- The Code is hosted on Github.com
- The URL to clone from is https://github.com/DrJWCain/VFS
- Please fork freely, and send pull requests if you want to.
- I'm very excited to see what the community make of this!

Questions?

- Email: james.cain@grassvalley.com
- Clone from: https://github.com/DrJWCain/VFS