SDC
STORAGE DEVELOPER CONFERENCE
SNIA ■ SANTA CLARA, 2014

Introduction to SMB 3.1

Greg Kramer
Senior Software Engineer
Microsoft

David Kruse
Principal Software Engineer
Microsoft

The SMB3.1 preview document is available online

http://msdn.microsoft.com/en-us/library/ee941641.aspx

Agenda

1. Extensible Negotiation
2. Preauthentication Integrity
3. Encryption Improvements
4. Cluster Dialect Fencing
5. Cluster Client Failover (CCF) v2
6. Questions

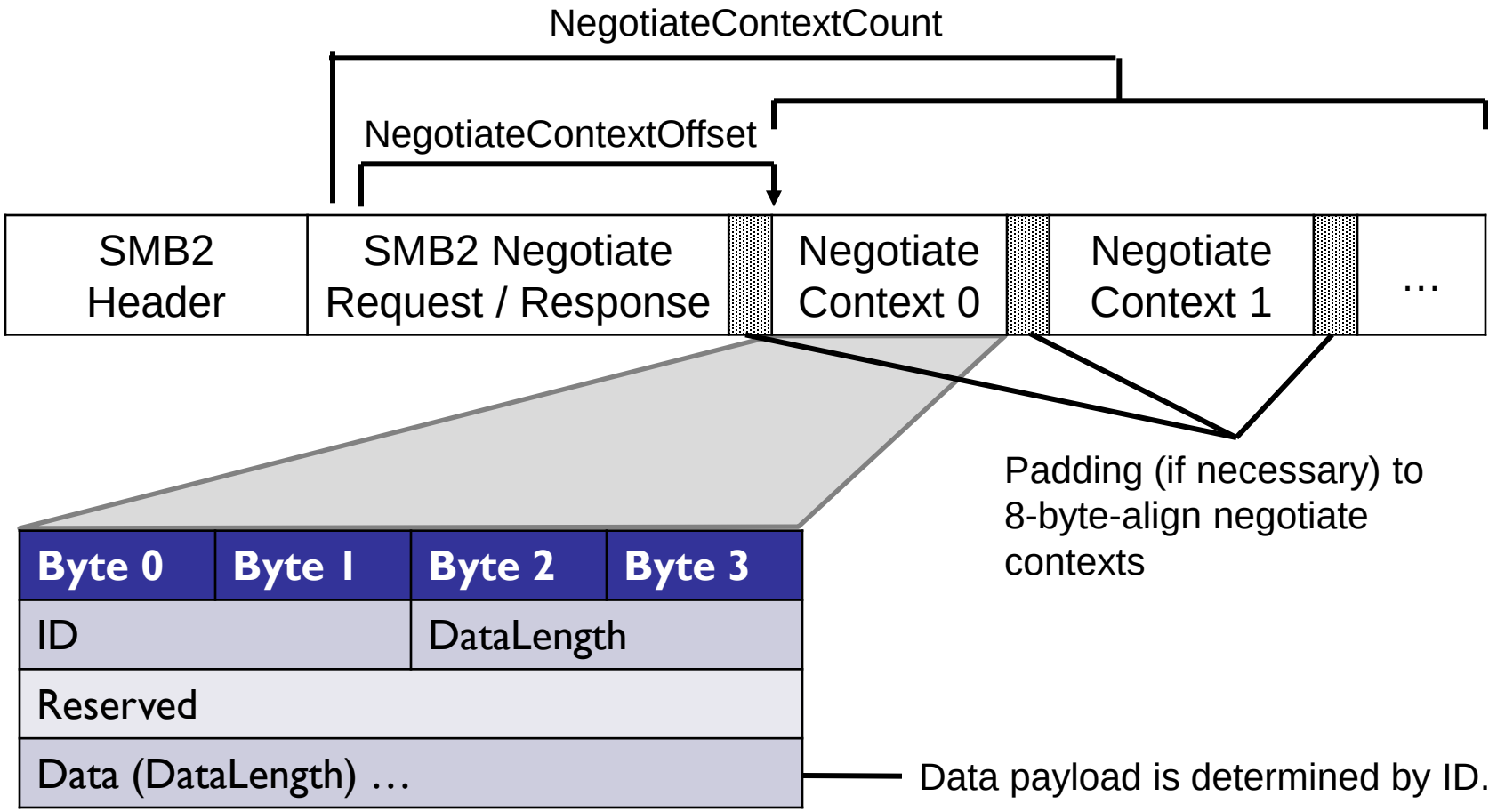
Agenda

1. Extensible Negotiation
2. Preauthentication Integrity
3. Encryption Improvements
4. Cluster Dialect Fencing
5. Cluster Client Failover (CCF) v2
6. Questions

Overview

- Very few reserved bits remain in the negotiate request / response messages.
  - How to negotiate complex connection capabilities?
- SMB 3.1 Extensible Negotiation
  - Extend negotiate request/response via negotiate contexts (same idea as existing create contexts).
  - Repurpose reserved fields in negotiate request / response as *NegotiateContextOffset* and *NegotiateContextCount* fields.
  - Add list of negotiate contexts to end of existing negotiate request / response messages.

Negotiate Contexts



Key Points

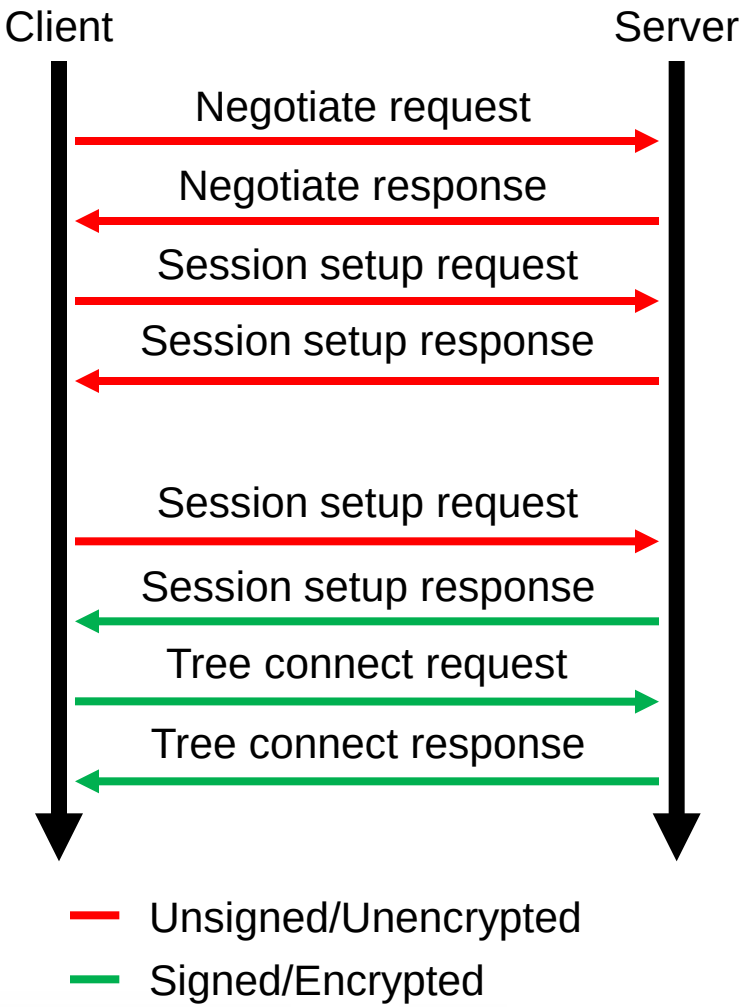
- Server processes negotiate contexts if the request's *Dialects* array indicates support for SMB 3.1.
- Server responds with negotiate contexts only if the response's *DialectRevision* is SMB 3.1.
- Receiver must ignore negotiate contexts it doesn't understand.
- SMB 2/3 server implementations MUST be willing to accept negotiate requests that are larger than the SMB2_HEADER + SMB2_REQ_NEGOTIATE + Dialects array.
  - Client does not know whether a server supports SMB 3.1. Must assume that it does and send negotiate contexts.

Agenda

1. Extensible Negotiation
2. Preauthentication Integrity
3. Encryption Improvements
4. Cluster Dialect Fencing
5. Cluster Client Failover (CCF) v2
6. Questions

Overview

- Preauthentication messages (negotiate and session setup) are vulnerable to tampering.
- SMB 3.0x Negotiate Validation doesn't protect negotiate contexts or session setup.
- How to provide end-to-end protection in a way that accommodates future protocol revisions?
- SMB 3.1 Preauthentication Integrity
  - Input hash of the preauthentication message exchanges to KDF when deriving a session's secret keys.
  - Message signature validation/decryption will fail in case of preauthentication message tampering.



Selecting a Hash Function

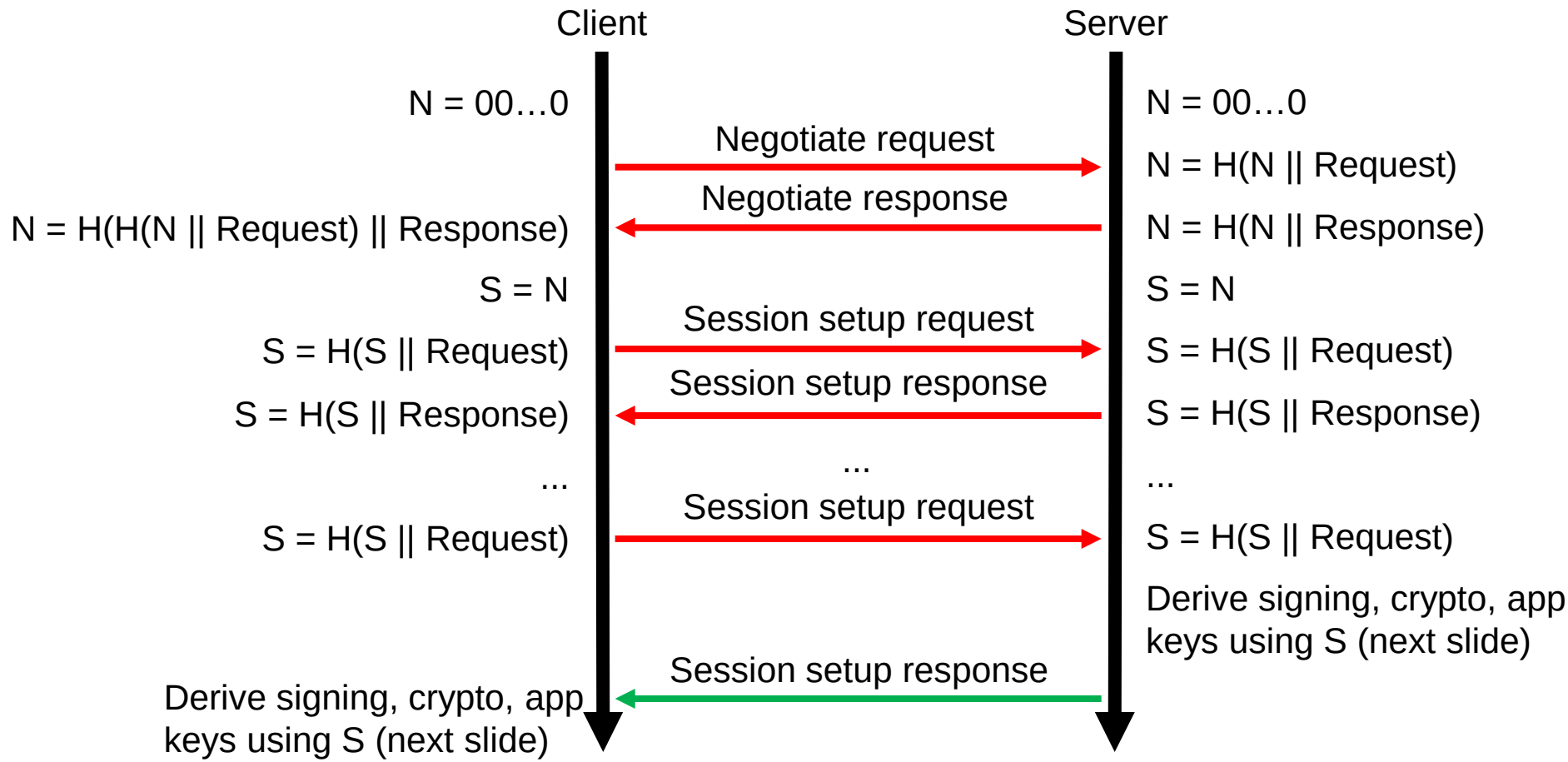
- SMB 3.1 client and server exchange mandatory negotiate contexts for each connection.
- Client's negotiate context specifies a set of supported hash functions.
- Server's negotiate context specifies the selected hash function.
- SHA-512 is currently the only supported hash function.

SMB2_PREAUTH_INTEGRITY_CAPABILITIES
(Negotiate Context ID: 0x0001)

Byte 0	Byte 1	Byte 2	Byte 3
HashAlgorithmCount		SaltLength	
HashAlgorithms			
…			
Salt			
…			

- Preimage attack resistance is provided by a salt value that the client and server generate via a secure PRNG per request/response.

Computing the Hash Value



H(x): hash value of bit string x using the negotiated hash function
A || B: concatenation of bit strings A and B

Deriving Secret Keys from the Hash Value

DerivedKey = KDF[1](SessionKey, Label[2], Context)

Derived Key	Label
Application Key	“SMBAppKey”
Signing Key	“SMBSigningKey”
Client to server cipher key	“SMBC2SCipherKey”
Server to client cipher key	“SMBS2CCipherKey”

Session’s final preauthentication integrity hash value (S)

1. KDF is SP108-800-CTR-HMAC-SHA256 (same as SMB 3.0x)
2. Note that KDF labels have changed since SMB 3.0x

Security Analysis

Result of an attacker tampering with the exchange of negotiate and/or session setup messages between an SMB 3.1 client and an SMB 3.1 server based on the resulting connection's SMB dialect.

Connection Dialect	Result
3.1	Attack is detected when client fails to validate the signature of the final session setup response.
3.0x or 2.x	Attack is detected by SMB 3.0x Negotiate Validation upon first tree connect.
1.x	Attack succeeds! SMB 1.x has no MITM attack mitigations

Note that SMB 3.1 Preauthentication Integrity cannot protect anonymous or guest sessions. The client and server can't sign messages without an authenticated context.

Key Points

- Preauthentication Integrity is mandatory for SMB 3.1.
- Session setup hashes are only calculated for master and binding session setup exchanges, not reauthentication.
- Preauthentication Integrity supersedes SMB 3.0x Negotiate Validation for SMB 3.1 connections.
- Preauthentication Integrity protects all future protocol revisions since hashing is message agnostic.

Agenda

1. Extensible Negotiation
2. Preauthentication Integrity
3. Encryption Improvements
4. Cluster Dialect Fencing
5. Cluster Client Failover (CCF) v2
6. Questions

Overview

- SMB 3.0x mandates the AES-128-CCM cipher
  - What if a different cipher is required for performance, regulatory requirements, etc?
- In SMB 3.1 ciphers are negotiated per-connection
  - Easily retire old ciphers and add new ciphers
- SMB 3.1 adds support for AES-128-GCM
  - More efficient encryption/decryption = higher IOPS/bandwidth
- SMB 3.1 clients can require a session to be encrypted even if server does not require encryption.

Selecting a Cipher

- SMB 3.1 client and server exchange negotiate contexts for each connection if they support encryption.
- Client's negotiate context specifies a set of supported ciphers in order from most to least preferred.

SMB2_ENCRYPTION_CAPABILITIES
(Negotiate Context ID: 0x0002)

Byte 0	Byte 1	Byte 2	Byte 3
CipherCount		Ciphers	
...			

- Server's negotiate context specifies the selected cipher.
  - Selection policy is server's choice: client-preferred, server-preferred, etc.
  - Reserved cipher ID 0x0000 indicates that the client and server have no common cipher.
  - No SMB2_ENCRYPTION_CAPABILITIES context in server response indicates that the server does not support encryption.
- Encryption capabilities flag is never set in an SMB 3.1 Negotiate Response.

SMB2_TRANSFORM_HEADER Changes

Byte 0	Byte 1	Byte 2	Byte 3
ProtocolId			
Signature			
…			
…			
…			
Nonce			
…			
…			
…			
OriginalMessageSize			
Reserved		Flags	
SessionId			
…			

Nonce size determined by cipher:

Cipher	Nonce Size (bytes)
AES-128-CCM	11
AES-128-GCM	12

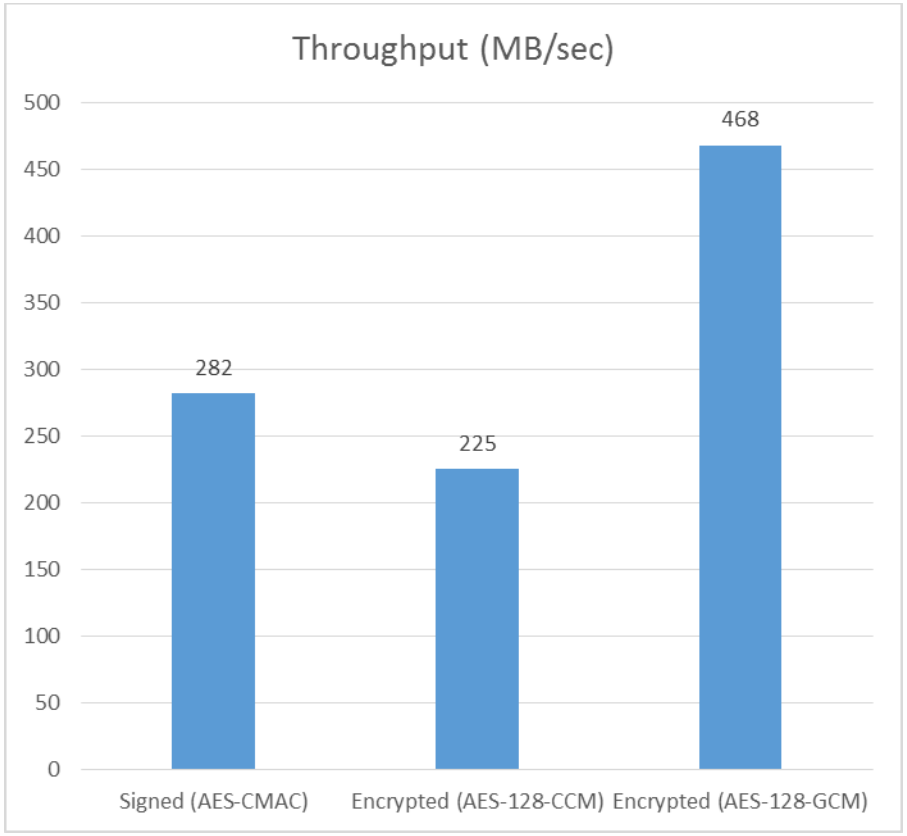
EncryptionAlgorithm field renamed to Flags:

Value	Meaning
0x0001	Payload is encrypted using cipher negotiated for the connection

AES-GCM Performance

- 2.0x faster than AES-128-CCM encryption.
- 1.6x faster than AES-CMAC signing!

Test configuration (client and server)	
CPU	2x Intel Xeon E5-2660 @ 2.2 GHz
Network	1x Intel Ethernet Server Adapter X520 @ 10 Gbps
Storage	SSD
Storage Workload	File copy (1 thread doing 8 async 1 MiB writes)



Client-mandated Encryption

- New security capability for security-hardened clients.
- Client sets the SMB2_SESSION_FLAG_ENCRYPT_DATA flag in its session setup request.
- Server acknowledges the encryption request by setting the SMB2_SESSION_FLAG_ENCRYPT_DATA flag in the session setup response.
- Server rejects all unencrypted requests for the session just as if the server had required encryption for the session.

Key Points

- AES-CCM required for SMB 3.0x crypto compatibility.
- AES-GCM provides significant performance gains.
- A session cannot be bound across channels that negotiated different ciphers.
  - Multichannel code has to be updated
- Client-mandated encryption depends on Preauthentication Integrity to guarantee secure connection.

Agenda

1. Extensible Negotiation
2. Preauthentication Integrity
3. Encryption Improvements
4. Cluster Dialect Fencing
5. Cluster Client Failover (CCF) v2
6. Questions

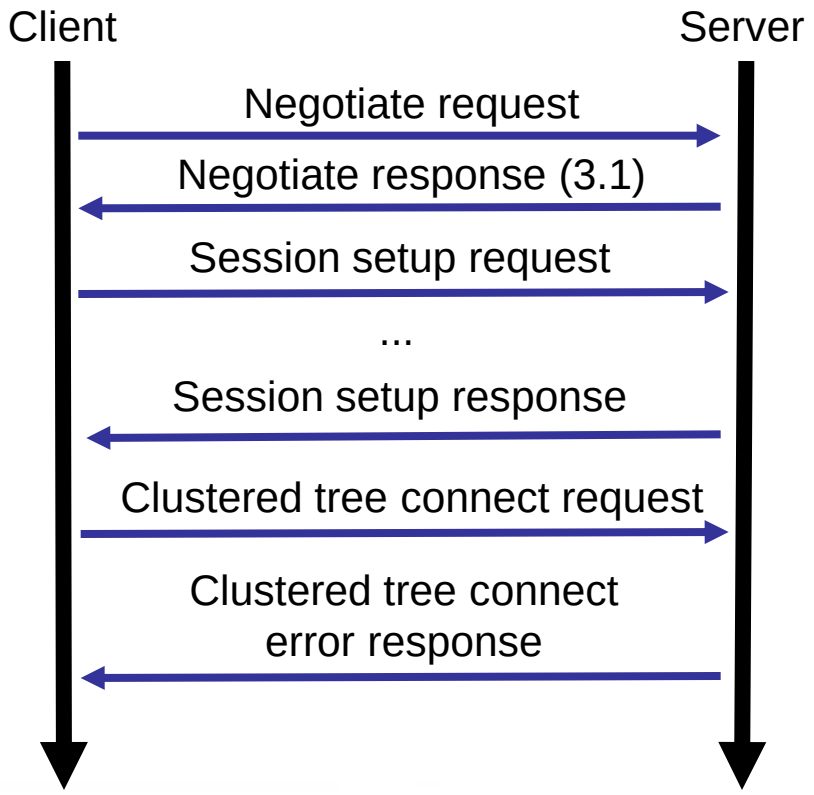
Overview

- All nodes in a cluster must support the same SMB dialects to allow a client to failover transparently between cluster nodes.
- How to allow clusters with nodes that support different maximum SMB dialects?
- SMB 3.1 Cluster Dialect Fencing
  - Define a cluster maximum SMB dialect
  - Fence access to cluster shares based on dialect.
  - Fenced clients instructed to reconnect at a cluster-supported dialect.

Fencing Clustered Tree Connects

An SMB 3.1 client accesses a clustered file share on an SMB 3.1 server that is a member of a cluster whose maximum clustered SMB dialect is 3.02.

1. Client negotiates 3.1, authenticates then issues tree connect.
2. Server determines that client is accessing a cluster share at an invalid dialect.
3. Server fails tree connect request with an extended error (status = 0xC05D0001) whose data payload indicates the maximum cluster-supported dialect (3.02).
4. Client disconnects, reconnects with new Client GUID, negotiates 3.02, authenticates, then reissues tree connect.



SMB2_TREE_CONNECT Request Changes

Byte 0	Byte 1	Byte 2	Byte 3
Structure Size		Flags	
PathOffset		PathLength	
Buffer			
…			

Reserved field renamed to Flags:

Value	Meaning
0x0001	Client has already successfully connected to a clustered file share on this server at the current SMB dialect.

- Once a client has successfully connected to a clustered share it must set the CLUSTER_RECONNECT (0x0001) flag on all subsequent clustered tree connect requests to the same server.
  - Addresses a race condition when the cluster's maximum SMB dialect is being raised.

Key Points

- Dialect fencing only affects clustered share access.
  - Clients can still access non-clustered shares using SMB 3.1 even if the maximum cluster dialect is < 3.1.
  - Can’t mix clustered and non-clustered access on same connection.
- Client implementation should protect against infinite loop of tree connect failure, disconnect, reconnect, tree connect failure, …

Agenda

1. Extensible Negotiation
2. Preauthentication Integrity
3. Encryption Improvements
4. Cluster Dialect Fencing
5. Cluster Client Failover (CCF) v2
6. Questions

Agenda

1. Extensible Negotiation
2. Preauthentication Integrity
3. Encryption Improvements
4. Cluster Dialect Fencing
5. Cluster Client Failover (CCF) v2
6. Questions