SNIA STORAGE SECURITY SUMMIT
Wednesday, May 11, 2022 • Virtual

Securing Access to Network Files whether on-premises or in the Cloud

SMB3.1.1 Security Overview

Presented by Steve French
Principal Software Engineer
Azure Storage - Microsoft

A SNIA Event

SNIA Legal Notice

- The material contained in this tutorial is copyrighted by the SNIA unless otherwise noted.
- Member companies and individual members may use this material in presentations and literature under the following conditions:
  - Any slide or slides used must be reproduced in their entirety without modification
  - The SNIA must be acknowledged as the source of any material used in the body of any document containin material from these presentations.
- This presentation is a project of the SNIA.
- Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion please contact your attorney.
- The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.

  NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.

Legal Statement

- This work represents the views of the author(s) and does not necessarily reflect the views of Microsoft Corporation
- Linux is a registered trademark of Linus Torvalds.
- Other company, product, and service names may be trademarks or service marks of others.

Who am I?

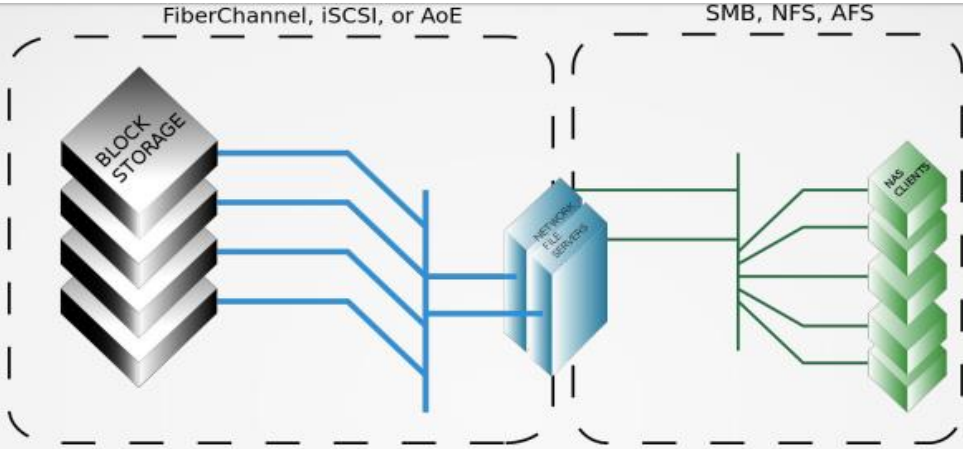
- Steve French smfrench@gmail.com
- Author and maintainer of Linux cifs vfs for accessing Samba, Windows, various SMB3/CIFS based NAS appliances and the Cloud (Azure)
  - Co-maintainer of the kernel server (“ksmbd”)
- Member of the Samba team, coauthor of SNIA CIFS Technical Reference,former SNIA CIFS Working Group chair
- Principal Software Engineer, Azure Storage: Microsoft

Outline

- Overview of SMB3
- The Security Challenges and Motivations
- The Security Features
  - Authentication
  - Identity
  - Access Control
  - Encryption
  - Data Integrity
- Where do we go from Here?

Why Network File Systems?

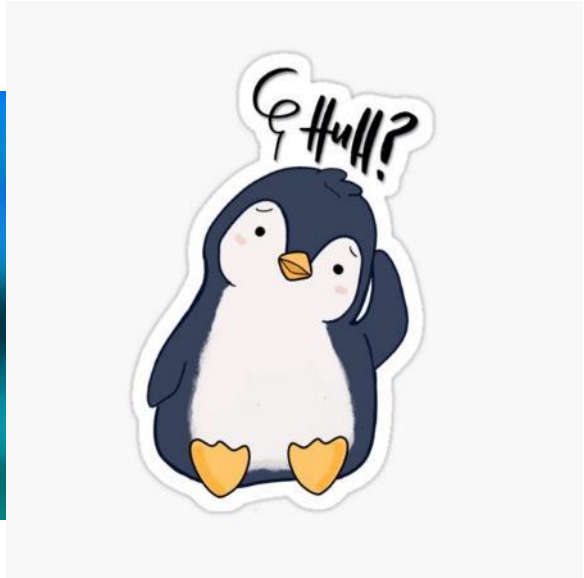
- Better Security, more features …



- NAS is superset of block (SAN) and object … but easier to manage
- NAS (now) can get 90+ of the performance of SAN with lower administrative costs and more flexibility. Easier to setup
- Attributes at the right granularity (file/directory/volume)
- Ownership information, easier to understand security, easy backup, optimizable with useful info on application access patterns, intuitive archive/encryption/compression policy, quotas, quality of service

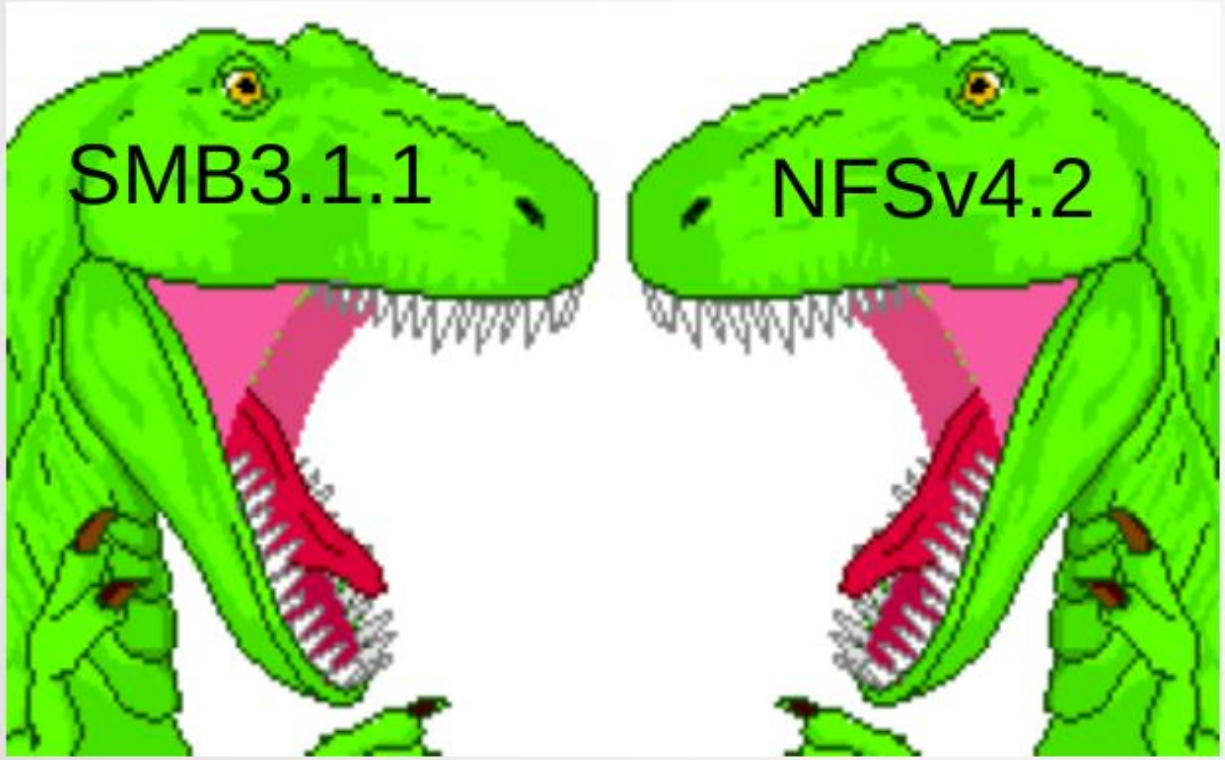
Disadvantages of Network/Cluster File Systems …

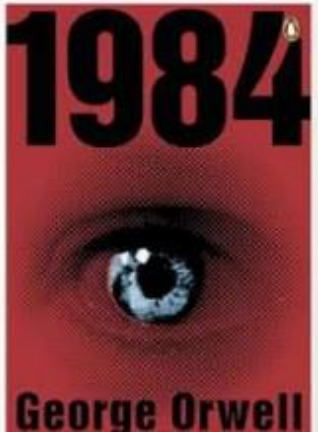
- MUCH more complex to write than alternatives (block or blob)
- Harder to optimize perf, can be slower
- And … we have more security features/challenges to talk about …



More than one thing happened in 1984

- The two Network File Systems were born in the 80s, devoured their competitors, now reborn stronger …





SMB (3.1.1) vs. NFS (4.2) comparison

- SMB3.11 includes a set of loosely related protocols that makes it much broader in scope
  - DFS (Global Name space)
  - Claims Based ACLs
  - File Replication
  - Witness Protocol and unique clustering and HA features
  - User/Group management (and many other admin and management functions)
  - Broader set of file system operations
  - Branch Cache (content addressable)
  - Volume Shadow Copy
  - MS-RSVD “SCSI over SMB3”
  - SMB3 RDMA was developed after NFS RDMA (and added some performance features – making SMB3 RDMA popular)
  - MultiChannel allows better adapter load balancing for SMB3

- NFS is more posix compatible
  - e.g. advisory byte range locking and unlink behavior are only emulated on SMB3 on Linux
- PNFS and layout operations are unique to NFS (SMB3 can not separate data and metadata)
- NFS operations are layered over SunRPC (while SMB3 goes directly over TCP) which complicates some optimizations
- Labelled NFS (SELinux security labels)
  - Other xattrs are not as easy to support in NFS although Marc Eshel has draft NFS xattr RFC for this

Overview of SMB3

- SMB3.1.1 is current dialect (introduced in late 2015, and extended multiple times)
- Most broadly deployed network/cluster file protocol (default on Windows, Macs, some embedded devices, and clients and servers available on all major operating systems)
- Part of a family of protocols e.g. DFS, smbdirect (RDMA), DCE/RPC, witness protocol, branch cache that offers the broadest set of function
- SMB3 is the best documented of the major file protocols (e.g. MS-SMB2 alone is 474 pages) with exhaustive test cases (Samba's test suite is over 200KLOC, and the Microsoft open source ones are huge) and multiple annual test events (some coordinated by SNIA)
- On Linux, cifs.ko is the kernel client. Samba (smbd) is most popular server, but there is also an open source SMB3.1.1 kernel server (ksmbd) and various user space libraries (including "libsmb2"), clients and tools which are built into various apps
- Servers range from small embedded devices running ksmbd, to Windows to Macs to NetApp, EMC and many others, including large Samba clustered deployments and even the "largest file server in the world," Azure Files.

Overview of SMB3.1.1 – key security features

- Key security features can be extended/negotiated via “negotiate contexts” sent in first frame
- Authentication:
  - Kerberos (encapsulated in SPNEGO)
    - integrates well into directory services like AD & AAD
- PreAuth Integrity
  - The initial session establishment verifies that the packets up to the first encrypted frame (SMB3 tcon) have not been tampered with
- Encryption:
  - GCM128 (GCM256 now possible as well)
    - Fast and secure, offloaded to hardware
  - Can be required on a per share (export) or per-server basis by server, or demanded by client
- Authorization/Access Control (see MS-DTYP)
  - Various mechanisms on server possible but usually use SMB3 ACLs with optional DAC (claims based ACLs)
  - Rich Auditing features (SACLs)
- Identity
  - Users are represented by globally unique “SIDs” – rather than local UIDs that POSIX/Linux rely on. SIDs includes optional domain prefix(s) e.g. "S-1-5-21-3623811015-3361044348-30300820-1013“
  - Mapping primitive “uids” to globally unique identifiers like SIDs can be done via Winbind or SSSD using hashing or from LDAP (Active Directory) via RFC2307

Goals and Motivations

- Access of files from small devices to large servers, and now the cloud, matters and it must be highly secure in today's hostile world. SMB3 is the most broadly implemented file protocol. Make sure it is secure enough (and continues to improve) for the many common file use cases, now, and in the future. Make sure users don't move away to worse alternatives due to missing features in SMB3

What about the kernel server (ksmbd) security?

- Many exciting improvements. In the first two releases after it was merged, multiple security problems were addressed.
- NTLMSSP Key exchange support added (Kerberos supported too)
- Limits on maximum outstanding requests enforced
- Throttling session setup failures to avoid dictionary attacks
- Removing (local, server side) symlink support to reduce symlink race attacks
- Remove insecure NTLMv1 auth (not just removing old SMB1 support)
- Strict packet processing checks

Lots of exciting improvements on the SMB3.1.1 client

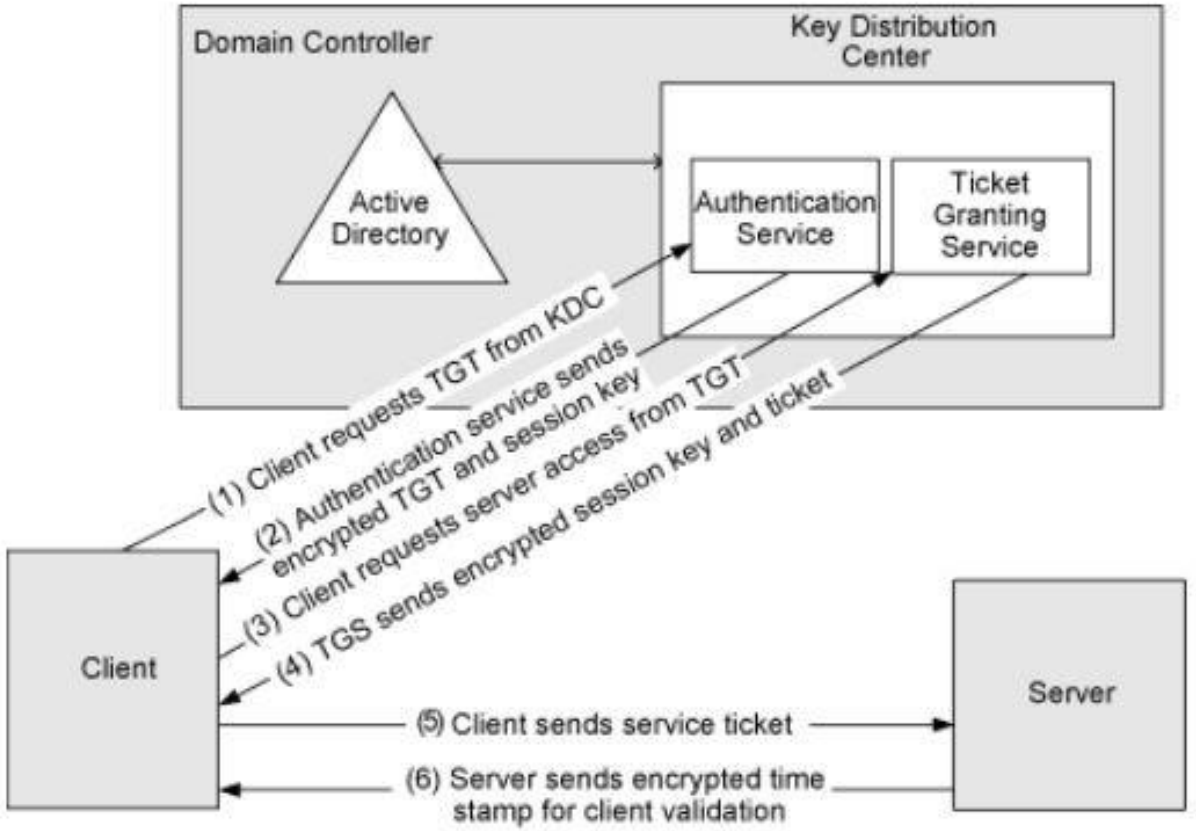
- Addition of support for strongest encryption
- Improvements to NTLMSSP
- Gradual deprecation continuing of older, less safe auth mechanisms and dialects (SMB1 e.g.)
- Additional fixes to userspace utilities to improve performance
- Improved support for additional security user cases
- Let’s get into details …

Diving into details …

Authentication and SMB3

- Various ways to authenticate
  - Kerberos service tickets (e.g. obtained from an ActiveDirectory DC) encapsulated in SPNEGO
  - NTLMv2 (NTLMv2) password hashes can be encapsulated in NTLMSSP
    - rawNTLMSSP
    - Or encapsulated again in SPNEGO
  - Other mechanisms (SPNEGO OIDs) also can be supported such as PKU2U (for peer to peer authentication). Some OS support a peer to peer Kerberos variant

Kerberos Authentication



SMB Authentication (continued)

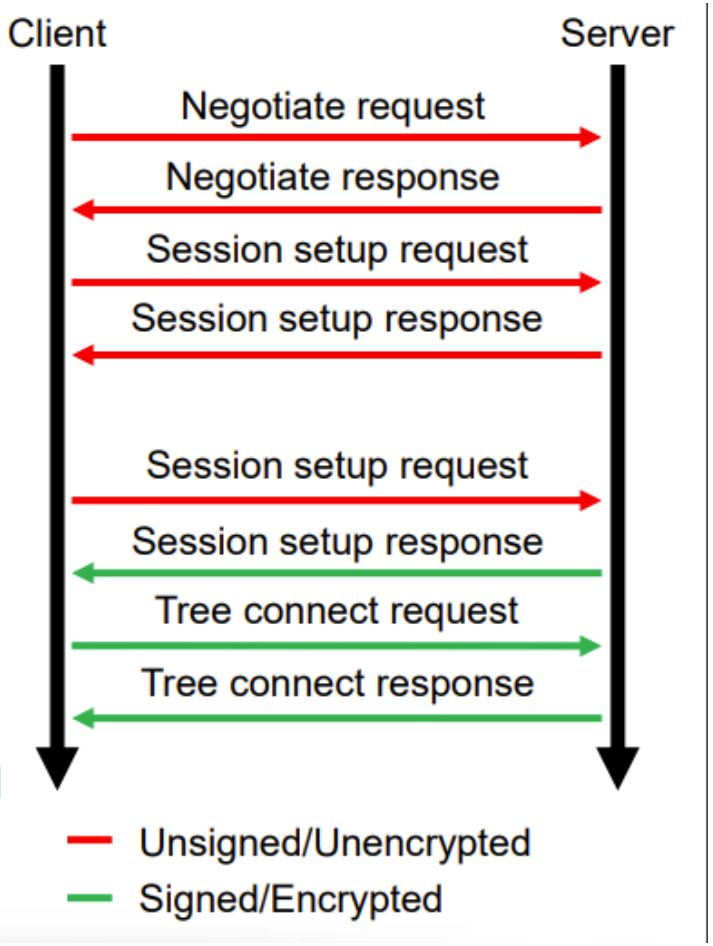
- Authentication mechanisms are largely opaque to SMB3 as their tokens are embedded in SPNEGO blogs
- Does require that OIDs are reserved with the IETF
- Obvious ones to add would be:
  - Peer to peer Kerberos (since Heimdal and others already support it, including Macs)
  - SCRAM (RFC5802 and 7677)
  - Perhaps OAUTH (may be less useful for system services)
- See RFC 2478

The problem with SPNEGO

- We don’t have an easy way to add libraries that make authentication “opaque” to the network fs
  - E.g. an /etc/spnego directory with config files with the IETF reserved OID for each of the supported (and configured/enabled) auth protocols on the machine, and standard library functions the fs (client or server) could call to get the blobs to package and send to the remote system
  - What if future security disasters cause us to have to invent new authentication protocols? How can they be dropped in seamlessly to network fs can use them?

Authentication – PreAuth Integrity

- How to protect preauthentication messages from tampering?
  - No protection prior to SMB 3.0
  - SMB 3.0x Negotiate Validation doesn't protect negotiate contexts or session setup messages.
- SMB 3.1 Preauthentication Integrity
  - Provides end-to-end protection of preauthentication messages.
  - Session's secret keys derived from hash of the preauthentication messages.
  - Signature validation/decryption of subsequent authenticated messages will fail in case of preauthentication message tampering.



- 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.

Result of an attacker tampering with negotiate and/or session setup messages based on the resulting connection's SMB dialect for a client and server that both attempt to negotiate SMB 3.1.

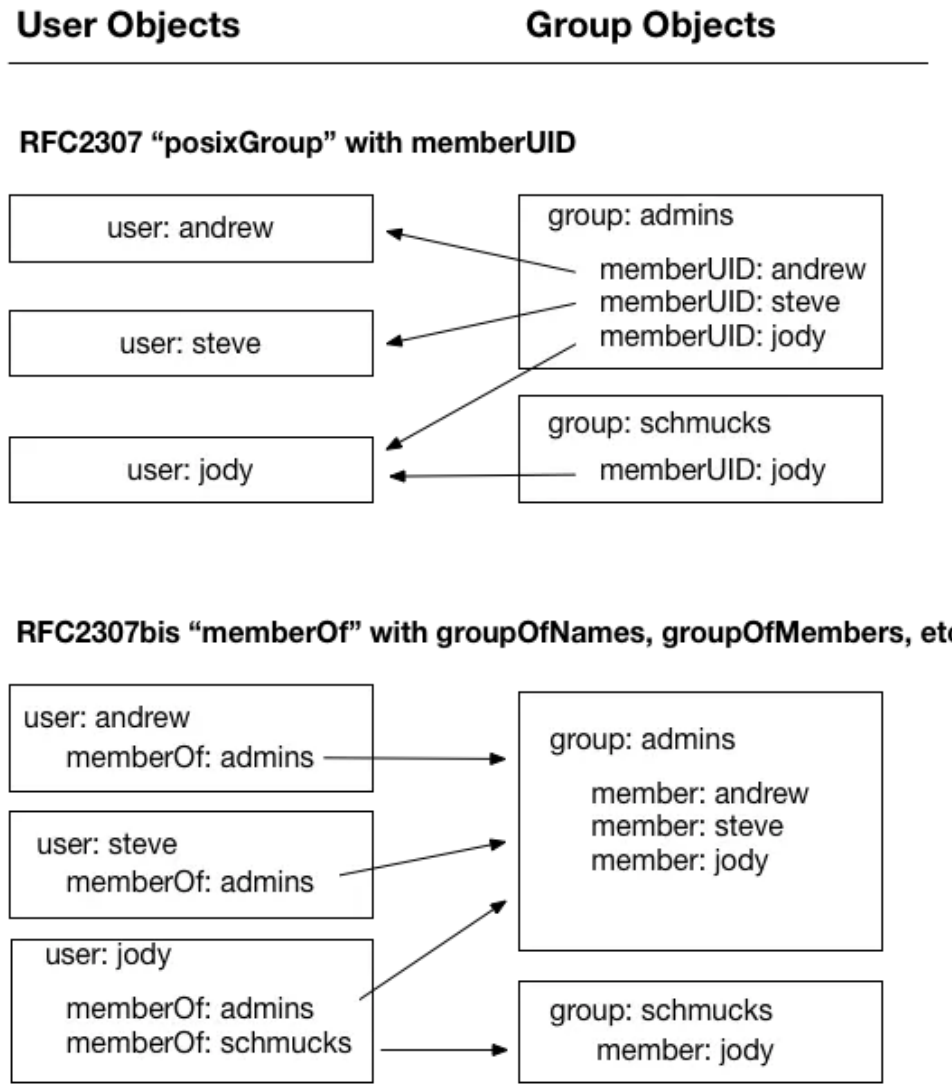
Connection Dialect	Result
3.1	Attack is detected when client fails to validate the signature of the final session setup response.
3.0x or 2.x	Dialect downgrade attack is detected by SMB 3.0x Negotiate Validation upon first tree connect.
1.x	Attack succeeds! SMB 1.x has no MITM attack mitigations

Identity

- The RFC2307 problem …
  - In the dark ages was NIS (“network information service”)
  - It was replaced in 1998 by a better, more secure alternative that used LDAP to store POSIX/Linux “uid” (e.g. 32 bit number) for some or all users in domain/tree/forest. Username and SID (globally unique identifier) and local Linux uid (not globally unique) could be mapped back and forth
- Some modules have their own idmapping options (e.g. cifs.idmap can be used to configure this for the SMB3.1.1 client, and “nfsidmap” for nfs)

RFC2307 alternatives

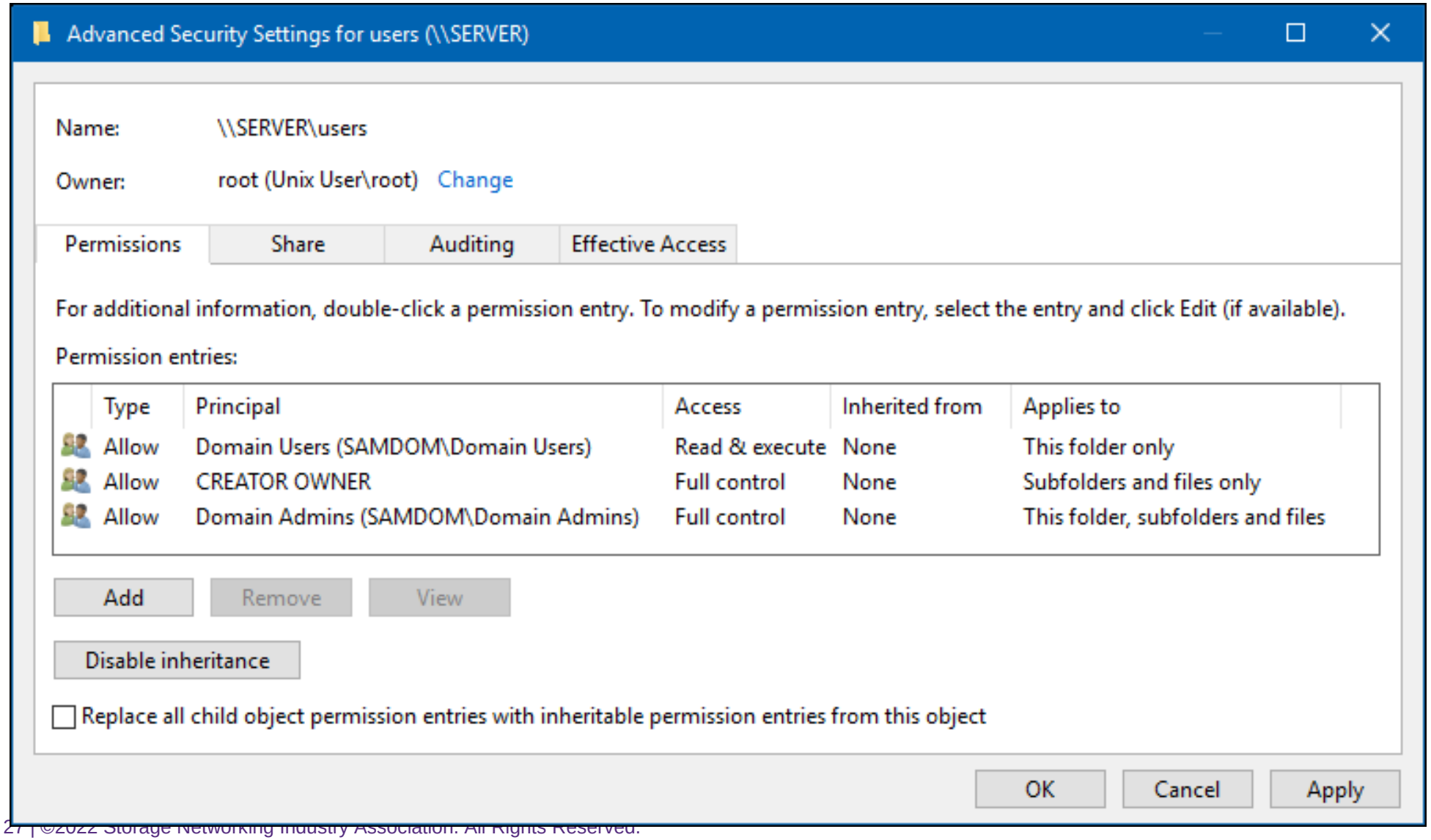
- Group entries in LDAP contained list of users but you couldn't find a user's groups by looking at the user's record in LDAP so RFC2307 was extended ("RFC2307bis")
- A replacement DBIS, "Directory Based Information Services" was proposed in 2015 https://tools.ietf.org/html/draft-bannister-dbis-mapping-06 but appears to be abandoned
- Or you can simply hash UIDs



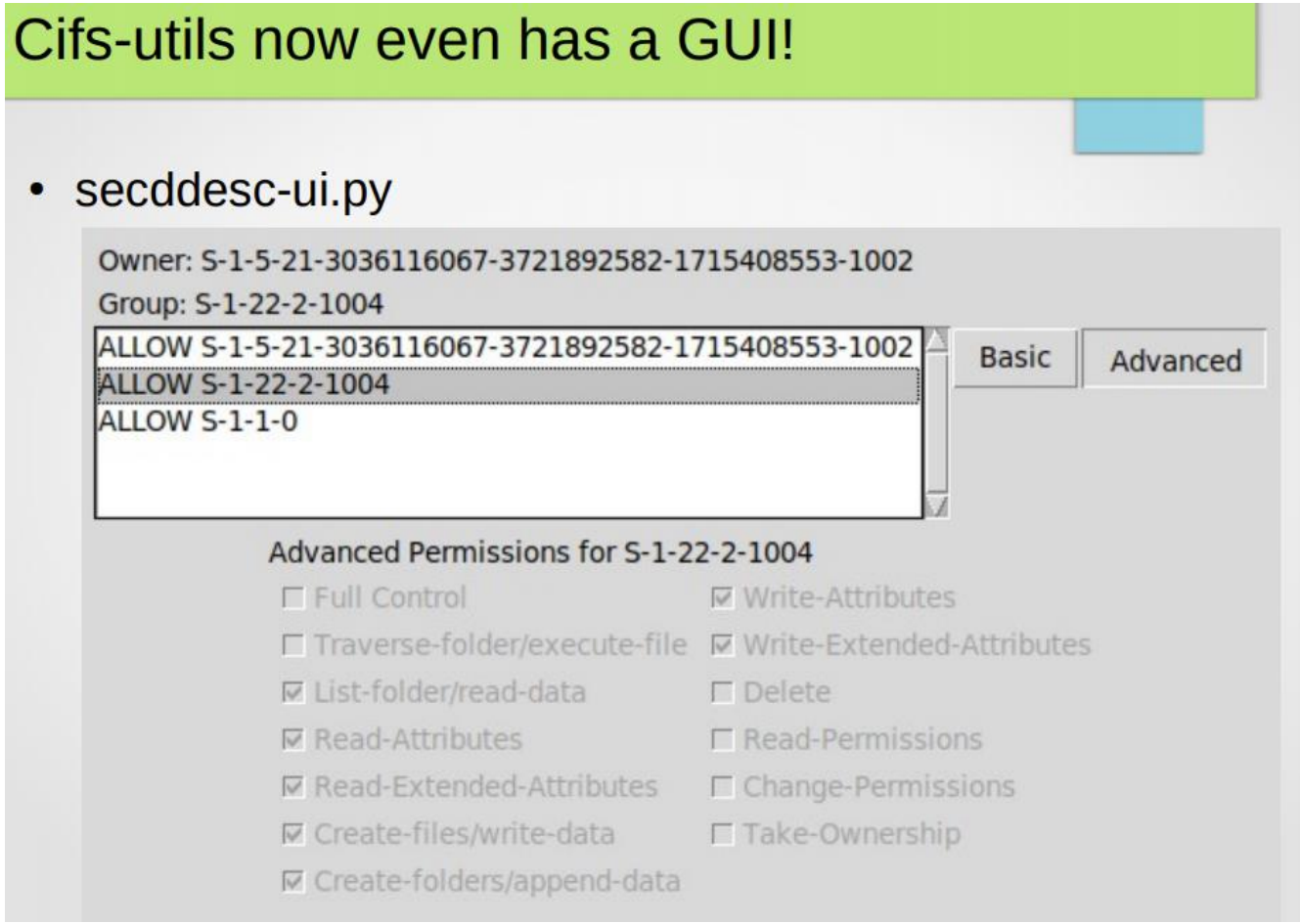
What are PAM and NSS missing …

- Go beyond getpwnam_r and getpwuid_r …
- Extended PAM/NSS (or equivalent) library interface is needed to provide standard way for file systems to query from/to: name<->uid<->SIDs<->OIDs not just from name<->uid but to globally unique identifiers
- Directory Services may need a replacement someday for RFC2307 if it is not sufficient to centrally store SIDs, UIDs, OIDs for each user in an organization
- Mapping all local users and groups to “guest” or a “default” Kerberos user is not sufficient

Access Control



Linux has ACL CLIs e.g. smbcacls getcifsacl

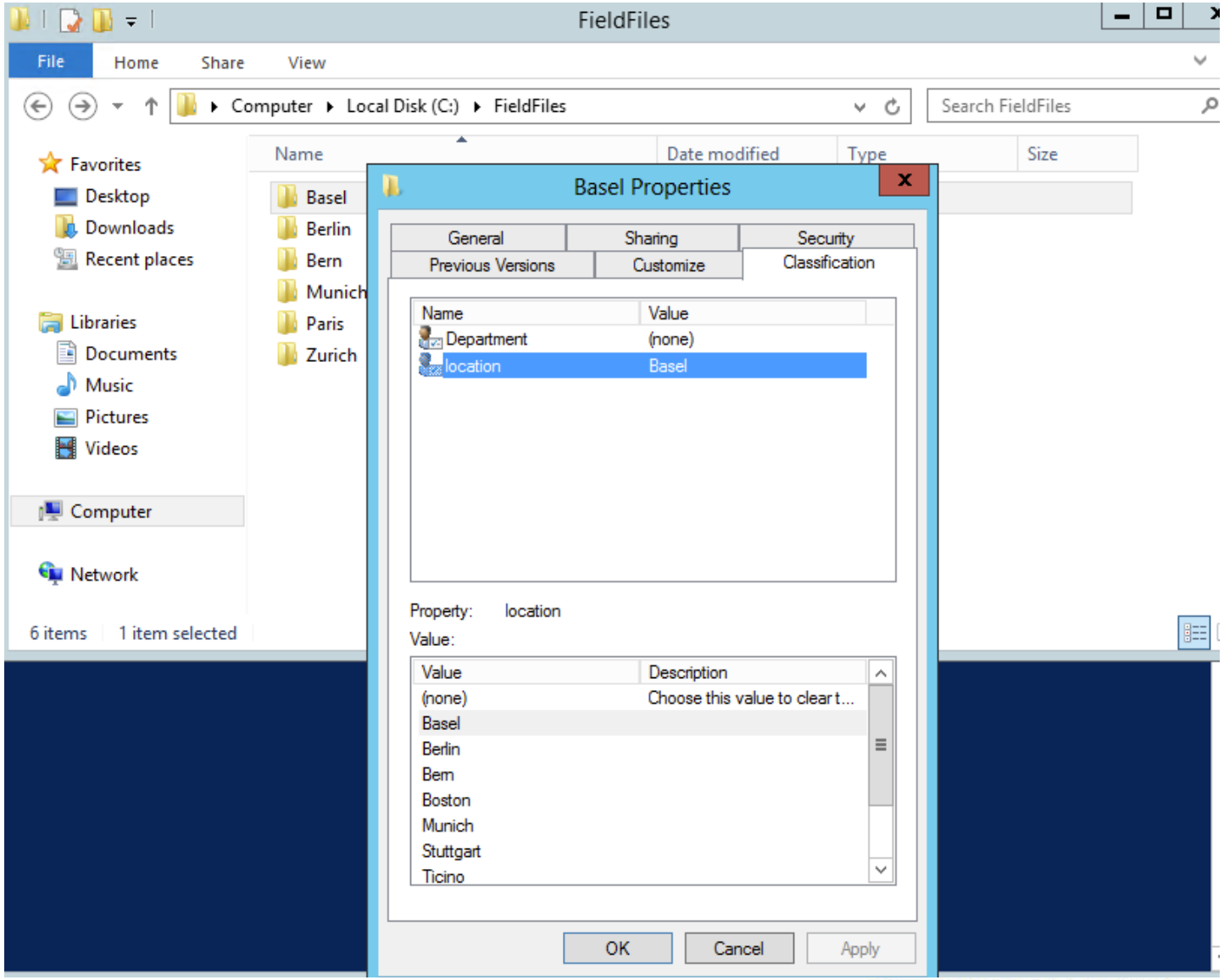


POSIX ACLs vs. “RichACLs”

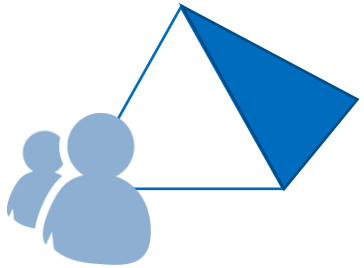
- Linux Mode bits are primitive with only 12 flags
- POSIX ACLs are more useful but lack support for Deny ACEs
- “RichACLs” first implemented in SMB/NTFS ACLs but now used by various file systems and OS are more broadly useful
- Now there are “Claims Based ACLs” (DAC)
  - Much richer in function
  - Supported by Kerberos

Claims based ACLs and DAC

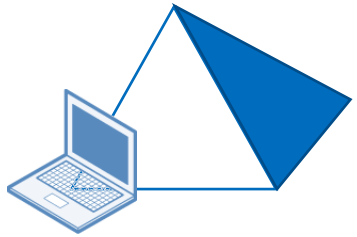
- Much richer in function (logical AND/OR allowed, different criteria like location can be used not just group membership)



Central access policies



AD DS



File Server

User claims
User.Department = Finance
User.Clearance = High

Device claims
Device.Department = Finance
Device.Managed = True

Resource properties
Resource.Department = Finance
Resource.Impact = High

ACCESS POLICY

Applies to: @File.Impact = High
Allow | Read, Write | if (@User.Department == @File.Department) AND (@Device.Managed == True)

Examples of access control problems

- Apps (especially on Linux and Macs) often check (and try to change) permissions on files. Servers always check perms (on open) of a file or directory
- Conflicts between permissions in mode bits, POSIX ACLs and RichACLs
- How to map mode 0707 into RichACL?
- Are POSIX ACLs (with no deny ACEs) only emulated? Or can the administrator or user change them remotely as well?
- How does “chown” affect the ACL?
- Semantic conflict between “remove directory entry” permission stored in the **parent** object in mode bits vs. “delete” permission on an object (**child**)

What about SELinux?

- Implements (Mandatory Access Control (MAC) support in Linux
- Individual objects have security labels stored in xattrs
- RFC7504 describes potential protocol requirements to support MAC (over NFS) and NFSv4.2 supports optional extensions for labels (“sec_label”) but this is client enforced in NFS. Should something similar be done in SMB3 (which already supports xattrs and alternate data streams)

Encryption

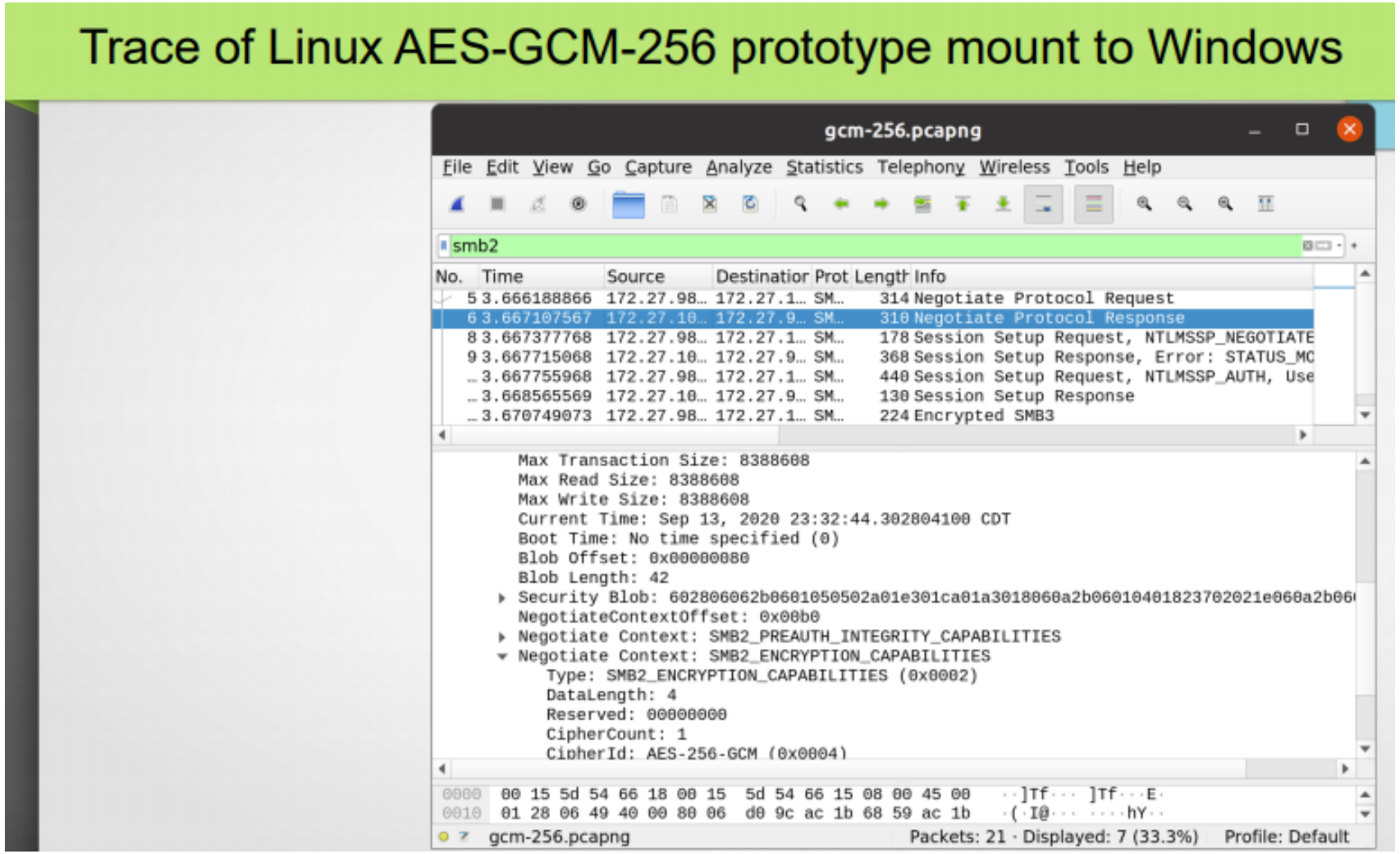
- Four currently supported
  - AES-128-CCM
  - AES-128-GCM
  - AES-256-CCM
  - AES-256-GCM
- Typically AES-128-GCM negotiated
  - Fast. Offload to hardware usually supported
  - In my testing, large I/O is often 5x or more faster to process (on client) with AES-128-GCM than AES-128-CCM on a typical VM

Why use AES-GCM-128 with SMB3.1.1 client?

GCM Fast

- Can more than double write perf! 80% for read
- Works with Windows, and with complementary recent changes to Samba server, mounts to Samba also benefit (a lot)
- In 5.3 kernel

Linux client support for GCM 256… (added in 5.12 kernel)



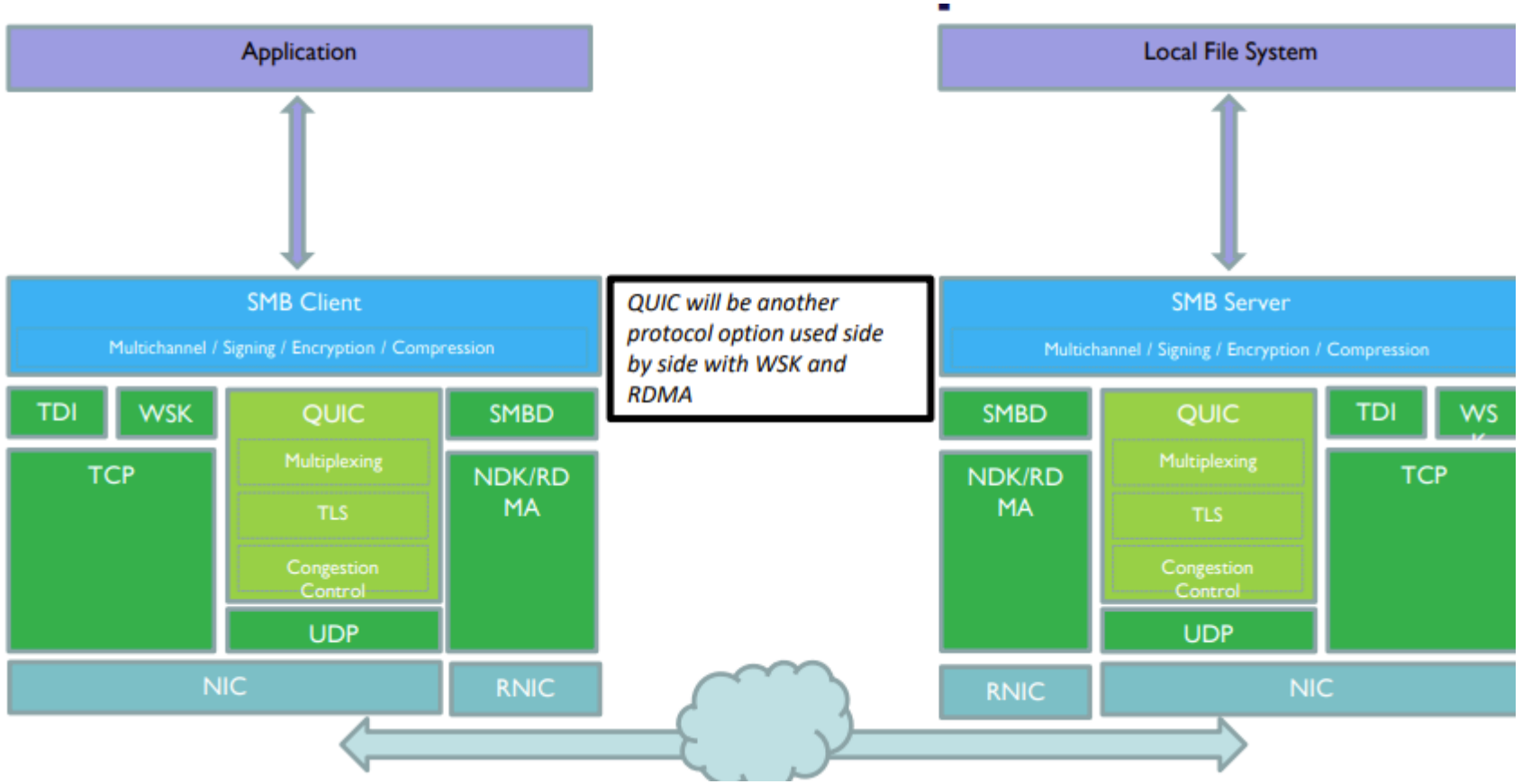
What about SMB3.1.1 over RDMA security?

- SMBDirect is very, very fast
- RDMA encryption shipped in 20H1
  - Includes 256 bit AES
- RDMA Signing
  - Feature complete
  - Includes AES-GMAC (faster)
- See presentation by Wen Xin at SDC2020

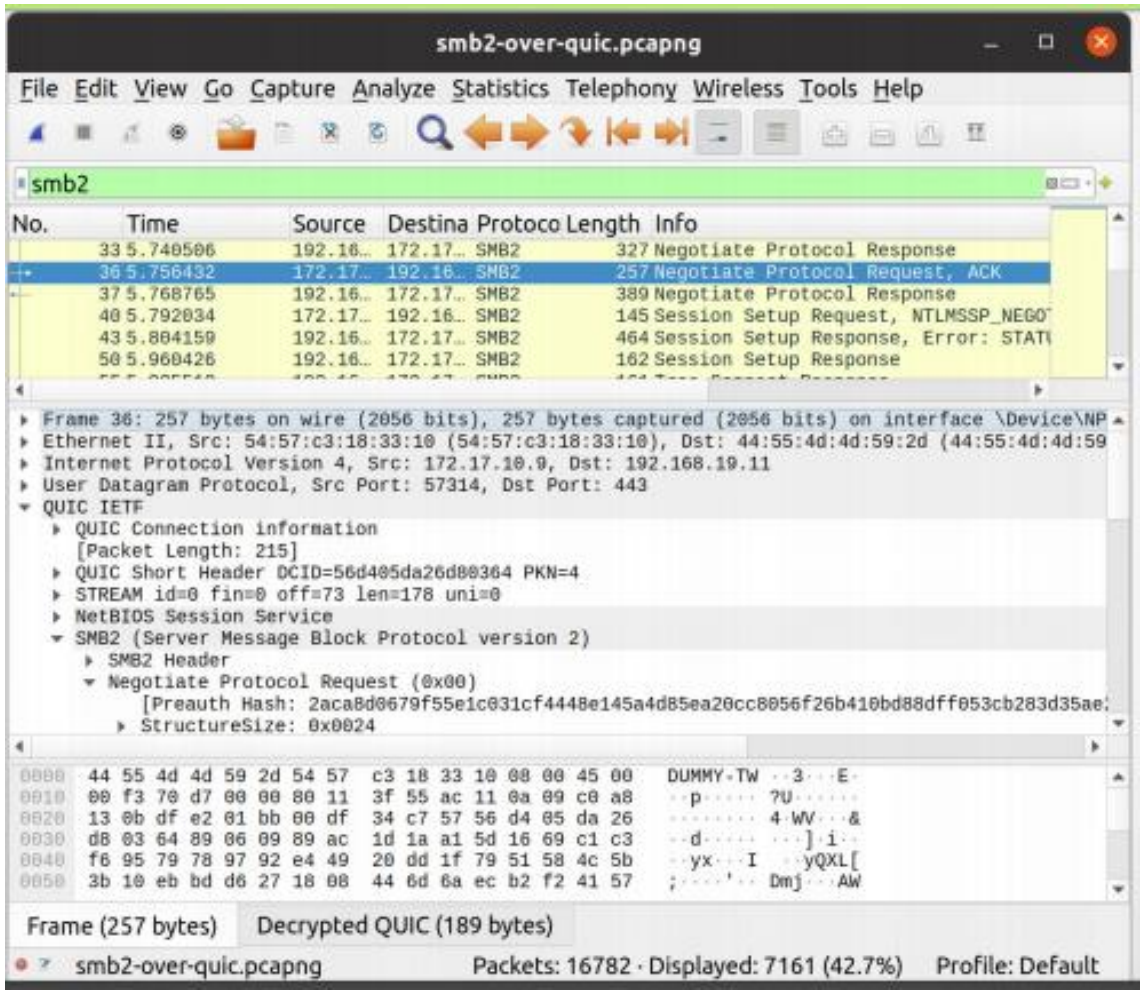
QUIC

- Solves the “port 445 problem” but has some interesting performance features too
  - Faster connection set up. 1 Round Trip(1-RTT) for initial connections. 0-RTT for resumed connections.
  - No head-of-line blocking
  - Better transitions between networks
  - Loss recovery and congestion control improvements

Windows QUIC implementation



Wireshark can decode SMB3.1.1 over QUIC



SMB3.1.1 over QUIC for Linux client

- We lack a Linux kernel driver for QUIC
  - What about porting msquic to kernel?
  - TLS1.3 is already in kernel since early 2019
  - Only needs about ~30K of QUIC code ported
  - See e.g. https://github.com/microsoft/msquic for an example cross platform QUIC implementation (would need coding style changes to be mergeable into kernel)
  - Not just about encryption: would help perf and congestion control too
  - QUIC support in kernel also requested by at least three other network and cluster fs on Linux (so far) according to discussions at the annual SNIA SDC last month

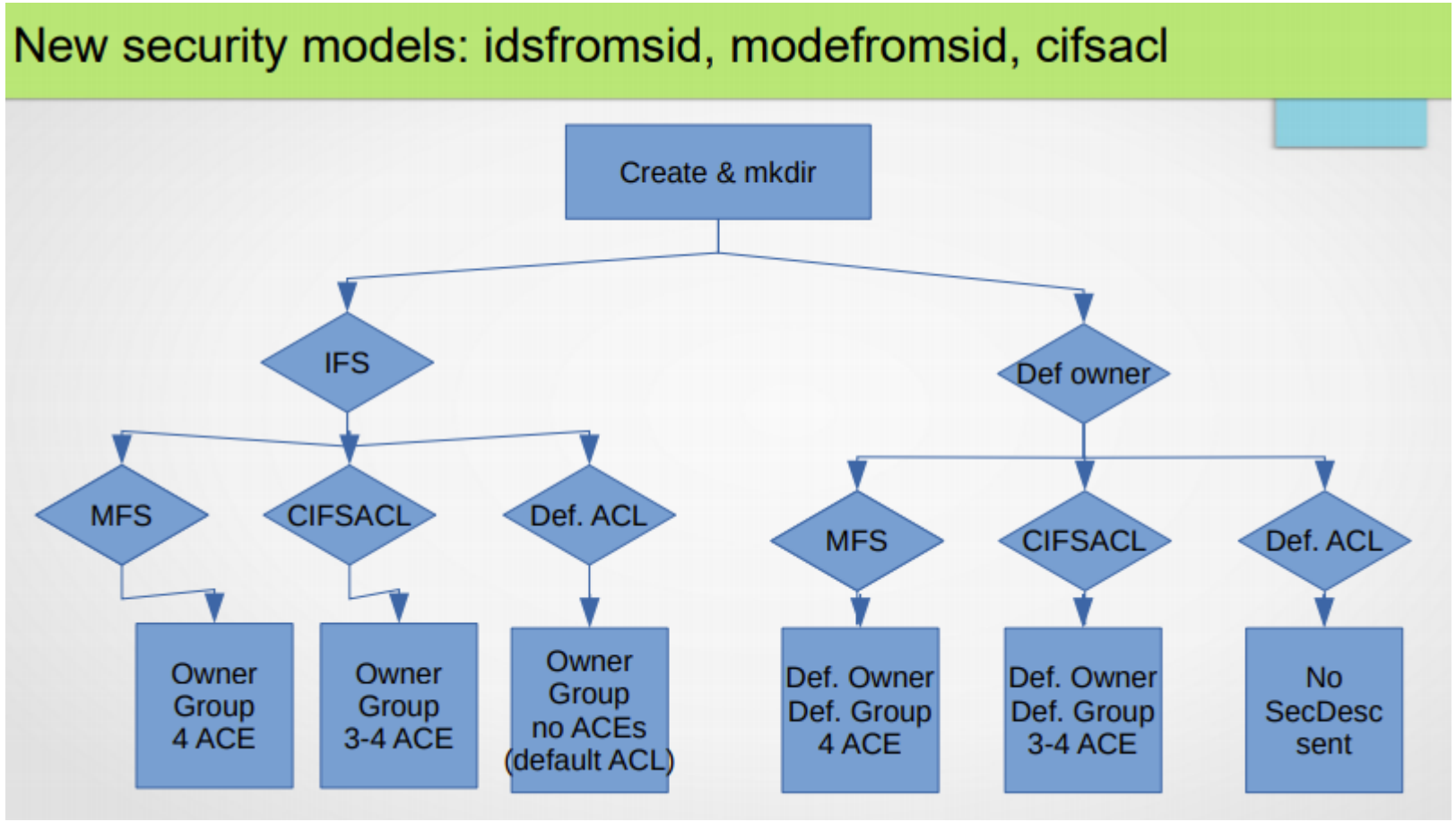
SMB3.1.1 layered on top of QUIC stack

- No difference in SMB3.1.1 multichannel
- No SMB signing/encryption by default
- SMB over QUIC will use server certificate to make sure there is no server spoofing attack
- Server listens on port 443
- No changes to SMB authentication
- QUIC multisession is not used on the server
- Negotiable SMB Connection Setting context
- For more information see Sudheer Dantuluri’s talk at 2020 SNIA SDC Conference

Data Integrity

- Ext4 supports “fs-verity” to allow enhanced integrity checking on read only files. This is not visible over SMB3 or other network fs
- SMB3 allows setting (on files or directories):
  - FILE_ATTRIBUTE_INTEGRITY_STREAM in order to enable enhanced integrity checking
  - FILE_ATTRIBUTE_NO_SCRUB_DATA to disable checks of data integrity by background scanners

Client configuration choices for security features



TODOs on the Linux SMB3.1.1 client

- Broaden the supported security scenarios
- Better SELinux integration with SMB3.1.1
- Improve the support for multiuser Kerberos mounts, winbind integration (e.g. for idmapping and ticket refresh – via cifs.upcall)
- Add stronger peer to peer auth support (e.g. PKU2U which Windows supports or peer to peer Kerberos, local KDCs, which Macs support)
- Add support for 'dummy mounts' to ease cases where krb5 credentials aren't available when mount is setup at boot
- Even stronger encryption (AES-GCM-256)
- Solve the "port 445 problem": add QUIC support (may be helpful for some non-encrypted cases in the future as well) but need a QUIC kernel driver for Linux … would the open source project msquic be worth porting?
- Improve packet signing perf (when encryption not negotiated) by adding support for the new signing negotiate context (and faster AES-GMAC)

Actions. What next …

- Auth: Is Kerberos good enough for all protocols?
  - If not add support to SPNEGO (reserve OID, construct Linux library) for OAuth (or other future auth mechanisms) so clients and servers can package keys/certificates opaquely
- Identity: extend pam/nss to allow mapping global (SIDs and OIDs), not just local identities to/from names. Winbind and SSSD will likely not be the only ones who plug in
  - And support for id mapping for containers: see the fs_context->user_ns field
- Access Control: how do we enable RichACLs in Linux (maybe someday “claims based ACLs”)? Or at least standardize API between the six or seven fs that could easily support them (not just NFS and cifs.ko)
- Integrity: Can we expose file attribute(s) in statx for marking/unmarking files for enhanced integrity checks?
- Encryption: how do we get a fast, efficient QUIC driver in kernel?

Please take a moment to rate this session.

Your feedback is important to us.

