SNIA DEVELOPER CONFERENCE

SDC 24

BY Developers FOR Developers

September 16-18, 2024
Santa Clara, CA

Troubleshooting/Debugging Issues on Linux SMB client

Bharath S M

Introduction

- Developer in the Azure Files team at Microsoft.
- Focused on improved Linux customer experience with Azure Files.
- Contributing to the development and enhancement of the Linux SMB client.

SMB(Server message block) protocol

- File system protocol developed by IBM in 1983
- Latest major version: SMB 3.1.1
- Commonly used default port for TCP connections: 445

Linux kernel SMB Client

- Linux kernel filesystem
- Source code located at fs/smb/client

Debug methods in Linux SMB Client: dmesg

- dmesg
  - On unexpected behavior, this should be the first place to check if some error was logged
- dmesg with verbose logging
  - Can roll over logs very quickly, but can be useful to pin-point the issue
  - Enabling extended debug messages on cifs/smb
    - echo 'module cifs +p' > /sys/kernel/debug/dynamic_debug/control
    - echo 'file fs/cifs/* +p' > /sys/kernel/debug/dynamic_debug/control
    - echo 7 > /proc/fs/cifs/cifsFYI
  - Disabling extended debug messages on cifs/smb
    - echo 0 > /proc/fs/cifs/cifsFYI

Example: Mounting share name which doesn't exist

- mount –t cifs ….
  - mount error(2): No such file or directory
  - Refer to the mount.cifs(8) manual page (e.g. man mount.cifs) and kernel log messages (dmesg)
- Error in dmesg:
  - CIFS: Attempting to mount //officestorage1.file.core.windows.net/noshare
  - CIFS: VFS: BAD_NETWORK_NAME: \\officestorage1.file.core.windows.net\noshare
  - CIFS: VFS: cifs_mount failed w/return code = -2
- -2: ENOENT : No such file or directory

Capturing SMB dynamic trace-points

- Linux SMB client supports ftrace to trace ongoing operations
- Easy-to-use tool to capture these tracepoints
  - Install trace-cmd
  - Start capturing
    - trace-cmd record –e cifs
    - If required only a specific trace points
      - trace-cmd record –e smb3_enter –e smb3_write
  - Decode traces:
    - trace-cmd report | less

SDC 24

Example: Mounting share name which doesn't exist

trace-cmd output:

…

```
mount.cifs-4015  [006]   174.605404: smb3_enter:              cifs_get_tcon: xid=27
mount.cifs-4015  [006]   174.605407: smb3_waitff_credits:  conn_id=0x5 server=officestorage1.file.core.windows.net current_mid=5 credits=71 credit_change=-1 in_flight=1
mount.cifs-4015  [006]   174.605407: smb3_cmd_enter:           sid=0x2918028d8000f8d tid=0x0 cmd=3 mid=5
cifsd-4017  [003]   174.663052: smb3_add_credits:     conn_id=0x5 server=officestorage1.file.core.windows.net current_mid=6 credits=104 credit_change=33 in_flight=0
mount.cifs-4015  [006]   174.663081: smb3_cmd_err:              sid=0x2918028d8000f8d tid=0x0 cmd=3 mid=5 status=0xc00000cc rc=-2
mount.cifs-4015  [006]   174.663084: smb3_tcon:          xid=27 sid=0x2918028d8000f8d tid=0x0 unc_name=\\officestorage1.file.core.windows.net\noshare rc=-2
mount.cifs-4015  [006]   174.666003: smb3_exit_err:             cifs_get_tcon: xid=27 rc=-2
mount.cifs-4015  [006]   174.666005: smb3_enter:                __cifs_put_smb_ses: xid=28
mount.cifs-4015  [003]   174.669898: smb3_waitff_credits:  conn_id=0x5 server=officestorage1.file.core.windows.net current_mid=6 credits=103 credit_change=-1 in_flight=1
mount.cifs-4015  [003]   174.669899: smb3_cmd_enter:           sid=0x2918028d8000f8d tid=0x0 cmd=2 mid=6
cifsd-4017  [003]   174.726930: smb3_add_credits:     conn_id=0x5 server=officestorage1.file.core.windows.net current_mid=7 credits=113 credit_change=10 in_flight=0
mount.cifs-4015  [005]   174.727046: smb3_cmd_done:             sid=0x2918028d8000f8d tid=0x0 cmd=2 mid=6
mount.cifs-4015  [005]   174.727068: smb3_exit_done:            cifs_mount_put_conns: xid=24
```


Capturing and extracting SMB wire captures

- tcpdump -i ethX -s0 -w /tmp/cifs-traffic.pcap host cifs_server.example.com and port 445
- In some cases, SMB traffic maybe encrypted over the wire. Wireshark supports SMB packet decryption with keys
- Decrypting SMB traces
  - smbinfo keys <path-to-a-file-on-an-smb3-mount>
  - Once you get the keys information add keys to wireshark to extract.
  - More info: https://wiki.samba.org/index.php/Wireshark_Decryption

Example: Tree connect failure

```
 7 17.441706     10.7.0.4          52.239.221.203    SMB2      286 Decrypted SMB3;Tree Connect Request Tree: \\officestorage1.file.core.windows.net\noshare
 8 17.505802     52.239.221.203    10.7.0.4          SMB2      195 Decrypted SMB3;Tree Connect Response, Error: STATUS_BAD_NETWORK_NAME
```

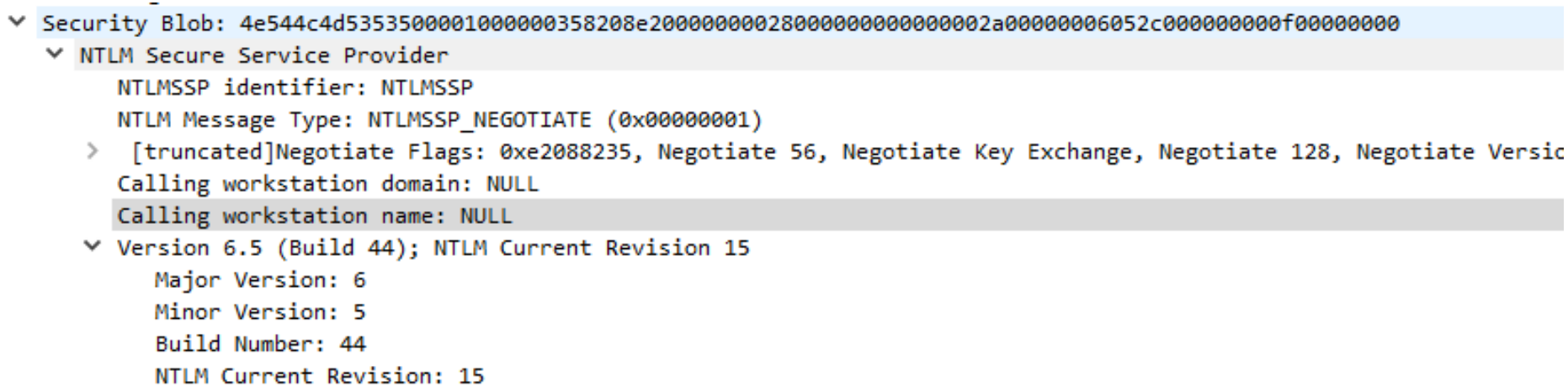
```
> Frame 8: 195 bytes on wire (1560 bits), 195 bytes captured (1560 bits)
> Ethernet II, Src: 12:34:56:78:9a:bc (12:34:56:78:9a:bc), Dst: Microsoft_be:81:ef (00:22:48:be
> Internet Protocol Version 4, Src: 52.239.221.203, Dst: 10.7.0.4
> Transmission Control Protocol, Src Port: 445, Dst Port: 48562, Seq: 202, Ack: 563, Len: 129
> NetBIOS Session Service
v SMB2 (Server Message Block Protocol version 2)
  > SMB2 Transform Header
  v Encrypted SMB3 data
    v SMB2 (Server Message Block Protocol version 2)
      v SMB2 Header
            ProtocolId: 0xfe534d42
            Header Length: 64
            Credit Charge: 1
            NT Status: STATUS_BAD_NETWORK_NAME (0xc00000cc)
            Command: Tree Connect (3)
            Credits granted: 33
          > Flags: 0x00000001, Response
            Chain Offset: 0x00000000
            Message ID: 22
            Process Id: 0x000078c3
            Tree Id: 0x00000000
            Session Id: 0x01f168633000004d
            Signature: 00000000000000000000000000000000
```


Useful commands

- Get Linux kernel version
  - uname -r
- Get kernel smb/cifs client module version
  - modinfo cifs | grep '^version'
- Get cifs-utils version
  - mount.cifs -V
- Get list of mounted SMB shares
  - mount –t cifs
- Get list of TCP connections for SMB port 445
  - ss –tn | grep 445

Finding client OS version from SMB NTLM traces

- Get Linux kernel version and CIFS version



Kernel Version 6.5 and CIFS module version 44

/proc/fs/cifs/DebugData

- Fields:
  - CIFS module and SMB version
  - Features
  - Address and Connection ID
  - Credits
  - Capabilities
  - Tree and Session ID
  - Status
  - MIDs

```
root@bharathsm-Virtual-Machine:~#  cat /proc/fs/cifs/DebugData
Display Internal CIFS Data Structures for Debugging
---------------------------------------------------
CIFS Version 2.49
Features: DFS,FSCACHE,STATS,DEBUG,ALLOW_INSECURE_LEGACY,CIFS_POSIX,UPCALL(SPNEGO),XATTR,ACL,WITNESS
CIFSMaxBufSize: 16384
Active VFS Requests: 0

Servers:
1) ConnectionId: 0x5 Hostname: officestorage1.file.core.windows.net
ClientGUID: 33DFEAB0-9F9B-AD46-93AF-C71F7D438CB0
Number of credits: 388,1,1 Dialect 0x311 signed
Server capabilities: 0x300057
TCP status: 1 Instance: 1
Local Users To Server: 1 SecMode: 0x3 Req On Wire: 0 Net namespace: 4026531840
In Send: 0 In MaxReq Wait: 0
Compression: disabled on mount

        Sessions:
        1) Address: 52.239.221.203 Uses: 2 Capability: 0x300057 Session Status: 1
        Security type: RawNTLMSSP  SessionId: 0x8150cc1c000889 encrypted
        User: 0 Cred User: 0

        Shares:
        0) IPC: \\officestorage1.file.core.windows.net\IPC$ Mounts: 1 DevInfo: 0x0 Attributes: 0x0
        PathComponentMax: 0 Status: 1 type: 0 Serial Number: 0x0 encrypted
        Share Capabilities: CONTINUOUS AVAILABILITY,    Share Flags: 0x30
        tid: 0x1        Maximal Access: 0x12019f

        1) \\officestorage1.file.core.windows.net\testshare2 Mounts: 1 DevInfo: 0x20 Attributes: 0xe
        PathComponentMax: 255 Status: 1 type: DISK Serial Number: 0xc0b246f9 encrypted
        Share Capabilities: CONTINUOUS AVAILABILITY, Aligned, Partition Aligned,        Share Flags: 0x0
        tid: 0x9        Optimal sector size: 0x200      Maximal Access: 0x11f01ff

        2) \\officestorage1.file.core.windows.net\testshare1 Mounts: 1 DevInfo: 0x20 Attributes: 0xe
        PathComponentMax: 255 Status: 1 type: DISK Serial Number: 0xa581c6fd encrypted
        Share Capabilities: CONTINUOUS AVAILABILITY, Aligned, Partition Aligned,        Share Flags: 0x0
        tid: 0x5        Optimal sector size: 0x200      Maximal Access: 0x11f01ff


        MIDs:
--

Witness registrations:
root@bharathsm-Virtual-Machine:~#
```


/proc/fs/cifs/Stats

- Provides statistics of SMB Ops
- Gives info about,
  - Request, Errors, Reconnects etc.
- Open files on server and client
- Useful for perf analysis and general debugging
- Extended Stats can be enabled with compile flag:CIFS_STATS2
- smbiostat uses this as source

```
root@bharathsm-Virtual-Machine:/# cat /proc/fs/cifs/Stats
Resources in use
CIFS Session: 1
Share (unique mount targets): 3
SMB Request/Response Buffer: 1 Pool size: 5
SMB Small Req/Resp Buffer: 1 Pool size: 30
Operations (MIDs): 0

0 session 0 share reconnects
Total vfs operations: 50 maximum at one time: 3

Max requests in flight: 3
1) \\officestorage1.file.core.windows.net\testshare2
SMBs: 39 since 2024-09-06 06:47:48 UTC
Bytes read: 0  Bytes written: 0
Open files: 1 total (local), 1 open on server
TreeConnects: 1 total 0 failed
TreeDisconnects: 0 total 0 failed
Creates: 12 total 0 failed
Closes: 11 total 0 failed
Flushes: 1 total 0 failed
Reads: 0 total 0 failed
Writes: 0 total 0 failed
Locks: 0 total 0 failed
IOCTLs: 0 total 0 failed
QueryDirectories: 0 total 0 failed
ChangeNotifies: 0 total 0 failed
QueryInfos: 12 total 0 failed
SetInfos: 2 total 0 failed
OplockBreaks: 0 sent 0 failed
2) \\officestorage1.file.core.windows.net\testshare1
```


Debugging mount issues

- Connectivity
  - Make sure server is online
  - Check routing between client and server
  - Port 445 connectivity between client and server
- Protocol version mismatch
  - Error: mount error(95): Operation not supported
  - SMB client and server may be using incompatible protocol versions
- Authentication failures
  - Error: mount error(13): Permission denied
  - Incorrect username, password etc.
- Share not found
  - Error: mount error(2): No such file or directory
  - Incorrect share name or path
- Unmount failure
  - Error: target is busy
  - Open handles to the mount point

Debugging Open handles

- Verify if app is closing the file
- Capture strace from an application
- lsof +D /path
  - Open FDs at file system level
- Check for cat /proc/fs/cifs/open_files
  - SMB handles to server
  - Eg:
    - root@bharathsm-Virtual-Machine:/mnt# cat /proc/fs/cifs/open_files
    - # Version:1
    - # Format:
    - # <tree id> <ses id> <persistent fid> <flags> <count> <pid> <uid> <filename>
    - 0x5 0x3b16c1960000021 0x431a000f 0x8441 1 150743 0 file.txt

Capturing all diagnostic data

- As you see debug information is scattered at different places.
- Capturing these information can be challenging.
- Even harder to debug infrequently occurring issues on customer systems.

- **Azure Files Linux** team is working on a utility called Always On Diagnostics for SMB & NFSv4 Linux clients to capture necessary diagnostics by monitoring anomalies.

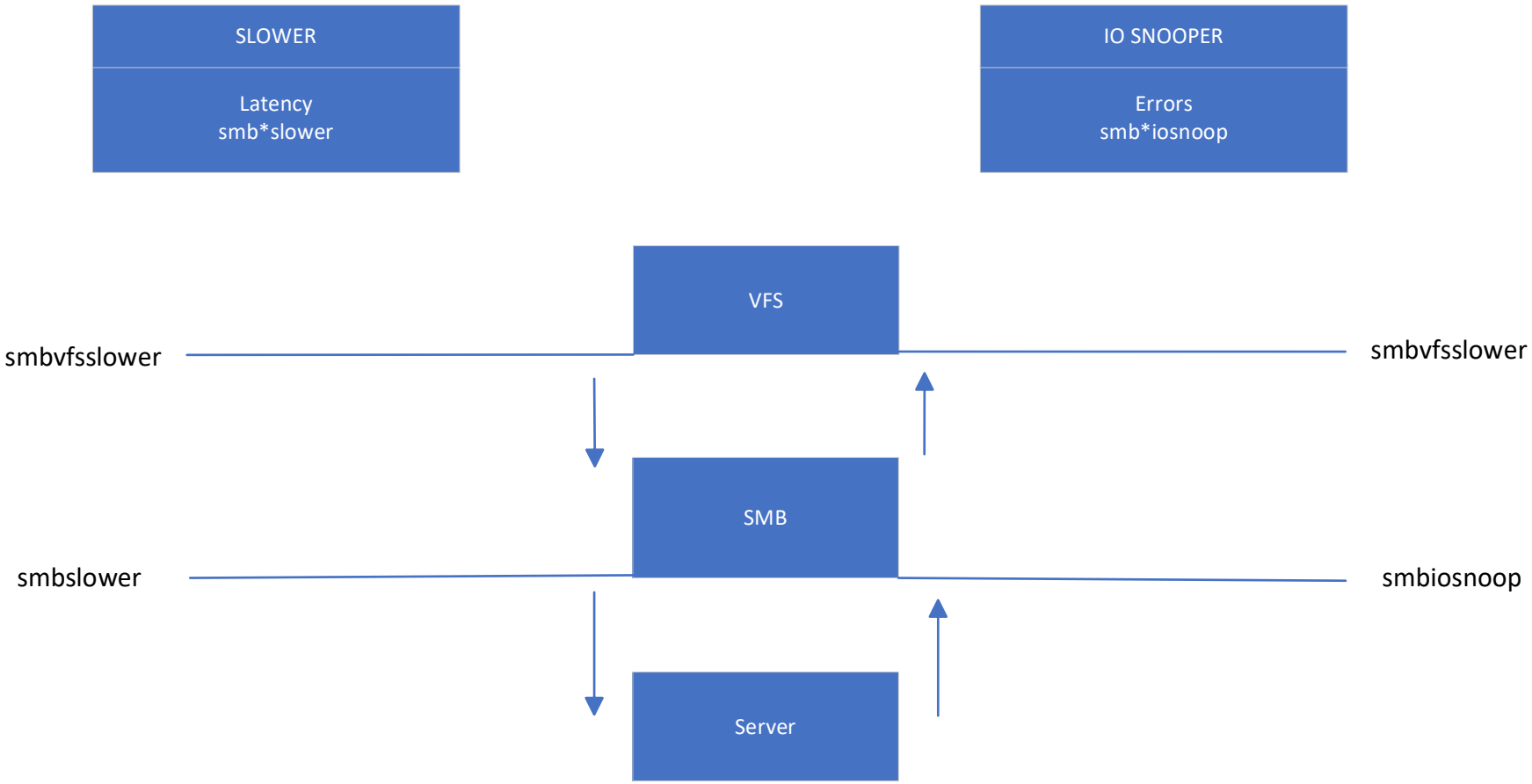
Ongoing work on AoD

- This tool will facilitate collection of necessary logs on SMB and NFSv4 clients
- Daemon to be running as a systemd service
- Can be configured to detect anomalies in variety of sources such as an error in dmesg logs, SMB debug data, error and latency metrics. This can trigger a capture of tcpdump, nfsstat, mountstat etc, along with system's CPU and memory usage.
- Easy to use for the customers and developers for collecting debug information on field issues which are hard to reproduce.

How does AoD trace anomalies?

- At the heart of capturing anomalies without compromising on perf too much are the eBPF tools which are currently under development
- Once fully developed, these tools would be pushed into cifs-utils repo
- Special mention to Meetakshi and Karthik who played a big role in developing these tools
- eBPF tools to detect anomalies:
  - smbslower and smbvfsslower
  - smbiosnoop and smbvfsiosnoop
  - And more

smb*slower and smb*iosnooper



What are smbslower and smbvfsslower

- smbslower: traces all SMB operations slower than a threshold (majorly measures time spent over the wire).
- smbvfsslower: traces the latency at the VFS layer more than a threshold (measures time spent in the SMB VFS callbacks).
- Traces all SMB operations.
- Measures the time spent in these operations
- Prints details based on the filter(s) user provides
- Uses Kernel dynamic tracing with eBPF.

Use cases: High latency anomalies

- If some of requests/operations are taking long time to complete, then we can run “smbslower and smbvfsslower” with threshold value to find out operations taking higher than specified value.
- We can find the operation which is taking longer time
- Once we hit a specified condition we can stop capturing.
- Captured logs will provide useful information.

Use case: Demo of slow request tracing with smbslower

- # ./smbslower.py 1
- Tracing SMB operations that are slower than 1 ms.

ENDTIME	TASK	PID	TYPE	LATENCY(ms)	SESSIONID	COMPOUND_RQST	ASYNC_RQST	TREE_ID/ASYNC_ID
20:35:57	cifsd	36464	SMB2_WRITE	143.997	0x1b80118000065	0	0	5
20:35:57	cifsd	36464	SMB2_WRITE	152.001	0x1b80118000065	0	0	5
20:35:57	cifsd	36464	SMB2_WRITE	154.458	0x1b80118000065	0	0	5
20:35:57	cifsd	36464	SMB2_WRITE	169.024	0x1b80118000065	0	0	5
20:35:57	cifsd	36464	SMB2_WRITE	165.800	0x1b80118000065	0	0	5
20:35:57	cifsd	36464	SMB2_WRITE	176.944	0x1b80118000065	0	0	5

What are smbiosnoop and smbvfsiosnoop

- smbiosnoop: Traces all SMB operations and tracks their return codes at SMB client.
- smbvfsiosnoop: Traces all SMB operations at VFS layer and tracks their return codes at VFS layer.

```
TASK          PID    TYPE     FUNCTION          RETVAL    ARGS
bash          1144    FILE   cifs_open           0       inode=13835076472101928960|oplock=0|dname=file.txt
bash          1144    FILE   cifs_flush          0       inode=13835076472101928960|oplock=0|dname=file.txt
```


Use cases: Operations failing with IO errors

- If operation is failing with an errors, we can use smbiosnoop and smbvfsiosnoop, to find the errors for a specified operation
- While debugging specific errors we can specify the operation and stop the trace automatically when issues occurs.

cifs.upcall

- cifs.upcall needs to be called from the kernel via the request-key callout program.
- The current cifs.upcall program handles two different key types:
  - cifs.spnego
    - This keytype is for retrieving kerberos session keys
  - dns_resolver
    - This key type is for resolving hostnames into IP addresses
- request-key.conf
  - create dns_resolver * * /usr/sbin/cifs.upcall %k
  - create cifs.spnego * * /usr/sbin/cifs.upcall %k

Debugging Kerberos failures

- Get the TGT before you mount.
- While mounting share cifs.upcall will get the service ticket using TGT.
- To get details of TGT and service tickets
  - klist
- Look at cifs.upcall system logs to understand failures(more detail in ref)

References

- https://wiki.samba.org/index.php/LinuxCIFS_troubleshooting
- https://wiki.samba.org/index.php/Wireshark_Decryption
- https://github.com/Azure-Samples/azure-files-samples/tree/master/SMBDiagnostics
- https://sprabhu.blogspot.com/2014/12/debugging-calls-to-cifsupcall.html
- https://www.snia.org/educational-library/tracing-and-visualizing-file-system-internals-ebpf-superpowers-2020
- https://learn.microsoft.com/en-us/azure/storage/files/storage-files-identity-auth-linux-kerberos-enable
- https://learn.microsoft.com/en-us/troubleshoot/azure/azure-storage/files/security/files-troubleshoot-linux-smb

Thank you

