==========
SERVER 2019 SCRIPT-BUILDING CHECKLIST (CYBERPATRIOT-FOCUSED)
TARGET: WINDOWS SERVER 2019 (DOMAIN CONTROLLER OR MEMBER SERVER)
==========

GENERAL SCRIPTING PRINCIPLES

[ ] Run the script in an elevated PowerShell session (Run as Administrator).
[ ] Assume the image may be a Domain Controller. Avoid:
  - Removing AD DS, DNS, SYSVOL contents, or critical system accounts.
[ ] Log everything your script does:
  - Create C:\CP_Script_Logs\hardening.log
  - Use Add-Content in PowerShell to append "SUCCESS/FAIL" per action.
[ ] Use idempotent operations:
  - Script should safely rerun without breaking anything (e.g., only set values, not blindly delete).
[ ] Group functions logically:
  - Phase 0: Prep / Logging
  - Phase 1: Users & Groups
  - Phase 2: Password/Lockout Policy
  - Phase 3: Local Security Options
  - Phase 4: Audit Policy
  - Phase 5: Services
  - Phase 6: Shares & SMB
  - Phase 7: DNS & AD-Specific Fixes
  - Phase 8: Firewall / RDP / Remote Assistance
  - Phase 9: Software: Updates, Removal, Backdoors
  - Phase 10: Prohibited Files & Permissions
  - Phase 11: Final Checks

==========
PHASE 0 – PREP & LOGGING (SCRIPT BOOTSTRAP)
==========

0.1 Create a log directory and log file
  GOAL:
    - Central log for everything the script does.
  SCRIPT IDEAS:
    - if (!(Test-Path "C:\CP_Script_Logs")) { New-Item -ItemType Directory -Path "C:\CP_Script_Logs" }
    - $Log = "C:\CP_Script_Logs\hardening.log"
    - Use: Add-Content $Log "Starting hardening at $(Get-Date)"

0.2 Detect if server is a Domain Controller

GOAL:
- Branch logic for DC vs member server.

SCRIPT IDEAS:
- $IsDC = (Get-WmiObject Win32_ComputerSystem).DomainRole -ge 4
- If $IsDC is $true, treat as DC (add AD-specific steps).
- Log the result.

0.3 Backup critical registry keys and GPO state

GOAL:
- Fail-safe point in case something goes wrong.

SCRIPT IDEAS:
- Export registry keys before changing:
  - reg export "HKLM\SYSTEM\CurrentControlSet\Services\DNS" "C:\CP_Script_Logs\DNS_before.reg" /y
  - reg export "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" "C:\CP_Script_Logs\LSA_before.reg" /y
- Export local security policy:
  - secedit /export /cfg C:\CP_Script_Logs\secpol_before.inf

==========
PHASE 1 – USER & GROUP AUDIT (SCRIPTABLE)
==========

1.1 Remove unauthorized local users (member server only)

GOAL:
- Delete known bad local accounts (from README or answer keys).

SCRIPT IDEAS:
- Use an array of known bad usernames:
  - $BadLocalUsers = @("ancano","tolfdir","Azula","combustionman") # example list
- For each:
  - if (Get-LocalUser -Name $u -ErrorAction SilentlyContinue) { Remove-LocalUser -Name $u }

1.2 Fix local group membership

GOAL:
- Only allowed members in Administrators, Remote Desktop Users, etc.

SCRIPT IDEAS:
- Define allowed local admins:
  - $AllowedAdmins = @("Administrator","SomeAdminFromREADME")
- Get current Administrators group members:
  - $admins = Get-LocalGroupMember "Administrators"
- Remove anything not in $AllowedAdmins (be careful not to remove default built-ins like "Administrator" or "Domain Admins"):
  - For each member:

- if ($AllowedAdmins -notcontains $member.Name -and $member.ObjectClass -eq "User") { Remove-LocalGroupMember "Administrators" $member }

1.3 Domain users and groups (if Domain Controller)
GOAL:
- Remove unauthorized domain users, adjust DnsAdmins, etc. [oai_citation:0‡CP 13 Unofficial r3 Answer Key.pdf](sediment://file_000000002d1071f8b8e44f8529cea83d)
SCRIPT IDEAS:
- Import ActiveDirectory module:
- Import-Module ActiveDirectory
- Example: remove Domain Users from DnsAdmins:
- $dnsAdmins = Get-ADGroup "DnsAdmins"
- Remove-ADGroupMember -Identity $dnsAdmins -Members "Domain Users" -Confirm:$false -ErrorAction SilentlyContinue
- Remove unauthorized domain users from admin groups (Domain Admins / Enterprise Admins) based on allowed lists.

1.4 Enforce password expiration for normal users
GOAL:
- Disable "password never expires" for non-service accounts.
SCRIPT IDEAS:
- If member server with local users:
- Get-LocalUser | Where-Object { $_.PasswordExpires -eq $false -and $_.Name -notin $ServiceAccounts } | ForEach-Object {
# Convert to AD/local change using net user if needed
}
- Domain (AD):
- Get-ADUser -Filter * -Properties PasswordNeverExpires | Where-Object { $_.PasswordNeverExpires -eq $true -and $_.Name -notin $ServiceAccounts } |
Set-ADUser -PasswordNeverExpires $false

==========
PHASE 2 – PASSWORD & ACCOUNT LOCKOUT POLICY (SCRIPTABLE)
==========

2.1 Configure password policy via net accounts
GOAL:
- Match CyberPatriot-style secure password settings. [oai_citation:1‡CP18-Server2019-Training2-Answer-Key.pdf](sediment://file_00000000c21871f8adae8fe44d9cc480)
SCRIPT IDEAS:
- net accounts /minpwlen:14
- net accounts /maxpwage:60
- net accounts /minpwage:1

- net accounts /uniquepw:24

2.2 Configure account lockout policy via net accounts
GOAL:
- Lockout after a reasonable number of failures, with secure reset.
SCRIPT IDEAS:
- net accounts /lockoutthreshold:5
- net accounts /lockoutduration:15
- net accounts /lockoutwindow:15
- Log success.

2.3 Ensure password complexity is enabled (Local Security Policy)
GOAL:
- Enforce complexity requirement.
SCRIPT IDEAS:
- You can use secedit INF editing or registry:
- Key: HKLM\SYSTEM\CurrentControlSet\Control\Lsa
- Value: "PasswordComplexity" (DWORD) = 1
- reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v "PasswordComplexity" /t REG_DWORD /d 1 /f

==========
PHASE 3 – LOCAL SECURITY OPTIONS (SCRIPTABLE VIA REGEDIT & SECEDIT)
==========

3.1 Accounts: limit blank password usage
GOAL:
- “Limit local account use of blank passwords to console logon only” = Enabled.
SCRIPT IDEAS:
- Registry:
- reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v "LimitBlankPasswordUse" /t REG_DWORD /d 1 /f

3.2 Disable Guest account and ensure renamed admin (local)
SCRIPT IDEAS:
- net user Guest /active:no
- To rename Administrator (if allowed and not pre-renamed):
- wmic useraccount where name="Administrator" rename "NewAdminName"

3.3 Network access: anonymous restrictions
GOAL:
- Disallow anonymous enumeration and Everyone->anonymous permission mapping.
SCRIPT IDEAS:

- reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v "RestrictAnonymousSAM" /t REG_DWORD /d 1 /f
- reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v "RestrictAnonymous" /t REG_DWORD /d 1 /f
- reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v "EveryoneIncludesAnonymous" /t REG_DWORD /d 0 /f

3.4 Network security: NTLM / LM hardening
SCRIPT IDEAS:
- reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v "LmCompatibilityLevel" /t REG_DWORD /d 5 /f
- # (5 = Send NTLMv2 response only; refuse LM & NTLM)

3.5 Interactive logon: require CTRL+ALT+DEL
SCRIPT IDEAS:
- reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System" /v "DisableCAD" /t REG_DWORD /d 0 /f

3.6 Interactive logon: inactivity timeout
SCRIPT IDEAS:
- reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System" /v "InactivityTimeoutSecs" /t REG_DWORD /d 900 /f

3.7 User Account Control (UAC) tightening
GOAL:
- Run all admins in Admin Approval Mode, secure desktop, etc.
SCRIPT IDEAS:
- $uacKey = "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System"
- reg add $uacKey /v "EnableLUA" /t REG_DWORD /d 1 /f
- reg add $uacKey /v "ConsentPromptBehaviorAdmin" /t REG_DWORD /d 2 /f
- reg add $uacKey /v "PromptOnSecureDesktop" /t REG_DWORD /d 1 /f

3.8 Recovery console: deny auto admin logon
SCRIPT IDEAS:
- reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Setup\RecoveryConsole" /v "SecurityLevel" /t REG_DWORD /d 0 /f

==========
PHASE 4 – USER RIGHTS ASSIGNMENT (SCRIPTABLE VIA SECEDIT INF)
==========

4.1 Export current security policy
SCRIPT IDEAS:
- secedit /export /cfg C:\CP_Script_Logs\secpol_current.inf

4.2 Edit INF file programmatically
GOAL:
- Modify user rights like “Access this computer from the network”, “Deny log on through RDP”, etc.
SCRIPT IDEAS:
- Read INF as text in PowerShell, use regex to update lines like:
  - SeNetworkLogonRight = *S-1-5-32-544,*S-1-5-11
    (Administrators + Authenticated Users)
  - SeDenyNetworkLogonRight = *S-1-5-32-546,*S-1-5-113
    (Guests + Local account)
- Save modified INF to C:\CP_Script_Logs\secpol_hardened.inf
- Apply:
  - secedit /configure /db C:\Windows\Security\Database\secedit.sdb /cfg C:\CP_Script_Logs\secpol_hardened.inf /areas USER_RIGHTS

4.3 Key rights to enforce in INF
EXAMPLES (map SIDs yourself in INF):
- SeNetworkLogonRight:
  - Administrators, Authenticated Users (+ Domain Controllers group if DC).
- SeRemoteInteractiveLogonRight:
  - Administrators, Remote Desktop Users.
- SeDenyNetworkLogonRight:
  - Guests, Local account.
- SeDenyRemoteInteractiveLogonRight:
  - Guests, Local account.
- SeBackupPrivilege, SeRestorePrivilege:
  - Administrators only.
- SeDebugPrivilege:
  - Administrators only.
- SeEnableDelegationPrivilege (Enable computer and user accounts to be trusted for delegation):
  - For DC: Administrators; for member server: No one.

==========
PHASE 5 – AUDIT POLICY (CLASSIC + ADVANCED)
==========

5.1 Basic audit policy using auditpol
SCRIPT IDEAS:
- Enable logon/logoff audits:
  - auditpol /set /category:"Logon/Logoff" /success:enable /failure:enable
- Account management:
  - auditpol /set /subcategory:"User Account Management" /success:enable /failure:enable

- System integrity:
  - auditpol /set /subcategory:"System Integrity" /success:enable /failure:enable
- Policy change:
  - auditpol /set /category:"Policy Change" /success:enable /failure:enable
- Object access (File Share):
  - auditpol /set /subcategory:"File Share" /success:enable

5.2 Ensure “Force audit policy subcategory settings” is enabled
SCRIPT IDEAS:
- reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v "SCENoApplyLegacyAuditPolicy" /t REG_DWORD /d 1 /f

==========
PHASE 6 – NETWORK ADAPTER, FIREWALL, RDP, REMOTE ASSISTANCE
==========

6.1 Disable unnecessary NetBIOS / DNS registration on NIC
(Optional; careful on DC)
SCRIPT IDEAS:
- Use WMI/NetAdapterAdvancedProperty or registry per NIC:
  - netsh interface ip set global dhcpmediasense=enabled (as needed)
- For NetBIOS:
  - Use PowerShell to set NetbiosOptions=2 in adapter’s TCP/IP parameters for non-critical members.

6.2 Enable Windows Defender Firewall for all profiles
SCRIPT IDEAS:
- netsh advfirewall set allprofiles state on

6.3 Tighten inbound firewall rules with PowerShell
GOAL:
- Only allow required services (DNS, AD, File sharing, RDP if allowed).
SCRIPT IDEAS:
- Disable all inbound rules for consumer / non-server apps:
  - Get-NetFirewallRule | Where-Object { $_.DisplayGroup -match "Xbox|MSN|Something" } | Disable-NetFirewallRule
- Ensure RDP rule is enabled only if README requires:
  - If ($AllowRDP) { Enable-NetFirewallRule -DisplayGroup "Remote Desktop" } else { Disable-NetFirewallRule -DisplayGroup "Remote Desktop" }
- Ensure DNS rules enabled on DC:
  - Enable-NetFirewallRule -DisplayGroup "DNS Server"

6.4 Remote Assistance off (script)
SCRIPT IDEAS:

- reg add "HKLM\SYSTEM\CurrentControlSet\Control\Remote Assistance" /v "fAllowToGetHelp" /t REG_DWORD /d 0 /f

6.5 WinRM: disallow unencrypted traffic
SCRIPT IDEAS:
- reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows\WinRM\Service" /v "AllowUnencryptedTraffic" /t REG_DWORD /d 0 /f

==========
PHASE 7 – SHARES, SMB, SYSVOL, NTDS (SERVER 2019-SPECIFIC)
==========

7.1 Remove unauthorized shares
SCRIPT IDEAS:
- Use PowerShell:
  - Get-SmbShare | Where-Object { $_.Name -notin @("C$","ADMIN$","IPC$","SYSVOL","NETLOGON") } | ForEach-Object { Remove-SmbShare -Name $_.Name -Force }

7.2 Fix SYSVOL / NETLOGON share permissions
GOAL:
- Remove “Everyone” from full access; keep Domain Admins, SYSTEM, etc.
SCRIPT IDEAS:
- Use icacls:
  - icacls C:\Windows\SYSVOL /remove:g Everyone
  - icacls C:\Windows\SYSVOL /grant "SYSTEM:(F)" "Domain Admins:(F)"
- Similar for NETLOGON path.

7.3 Disable NTDS dump share and remove dump files [oai_citation:2‡CP 13 Unofficial r3 Answer Key.pdf](sediment://file_000000002d1071f8b8e44f8529cea83d)
SCRIPT IDEAS:
- Remove any share pointing to NTDS.dit dumps:
  - Get-SmbShare | Where-Object { $_.Path -like "*NTDS*" } | Remove-SmbShare -Force
- Remove dump directory or .dit files if they are copies (NOT the live NTDS.dit):
  - Remove-Item "C:\Some\NTDS\DumpPath*" -Force

7.4 Disable SMBv1
SCRIPT IDEAS:
- Set-SmbServerConfiguration -EnableSMB1Protocol $false -Force

7.5 Require encryption on sensitive SMB shares
SCRIPT IDEAS:
- Set-SmbServerConfiguration -EncryptData $true -Force
- For specific shares:

- Set-SmbShare -Name "ExecShareName" -EncryptData $true

==========
PHASE 8 – DNS & ACTIVE DIRECTORY SECURITY (SERVER 2019 ROUND ITEMS)
==========

8.1 DNS plugin / malicious DLL removal [oai_citation:3‡CP 13 Unofficial r3 Answer Key.pdf](sediment://file_000000002d1071f8b8e44f8529cea83d)
GOAL:
- Remove malicious DNS plugin DLL and its registry reference.
SCRIPT IDEAS:
- Stop DNS service:
- sc.exe stop dns
- Remove malicious DLL example:
- $dll = "C:\Windows\System32\DNS\ipv6_dnsapi.dll"
- if (Test-Path $dll) { Remove-Item $dll -Force }
- Remove registry plugin:
- reg delete "HKLM\SYSTEM\CurrentControlSet\Services\DNS\Parameters" /v "ServerLevelPluginDll" /f
- Start DNS:
- sc.exe start dns

8.2 Configure DNS service restart-on-failure [oai_citation:4‡CP 13 Unofficial r3 Answer Key.pdf](sediment://file_000000002d1071f8b8e44f8529cea83d)
SCRIPT IDEAS:
- sc.exe failure DNS reset= 10 actions= restart/10000/restart/10000/restart/10000

8.3 DNS auditing level
SCRIPT IDEAS:
- Import-Module DnsServer
- Set-DnsServerDiagnostics -EventLogLevel 3

8.4 SIGRed workaround (if required)
GOAL:
- Depending on scenario, apply registry key / patch described in docs.
SCRIPT IDEAS:
- (Generic example, adjust to real requirement)
- reg add "HKLM\SYSTEM\CurrentControlSet\Services\DNS\Parameters" /v "TcpReceivePacketSize" /t REG_DWORD /d 0xFF00 /f

==========
PHASE 9 – SERVICES & FEATURES (SCRIPTABLE)
==========

9.1 Disable insecure or unnecessary services
[oai_citation:5‡CP18-Server2019-Training2-Answer-Key.pdf](sediment://file_00000000c21871f8adae8fe44d9cc480)
SCRIPT IDEAS:
- $servicesToDisable = @(
"RemoteRegistry",
"SMTPSVC","SimpleMailTransferProtocol",
"FTPSVC","Microsoft FTP Service",
"Telnet",
"TeamSpeak","TightVNC"
)
- ForEach ($svc in $servicesToDisable) {
Get-Service -Name $svc -ErrorAction SilentlyContinue | ForEach-Object {
Set-Service -Name $_.Name -StartupType Disabled
Stop-Service -Name $_.Name -Force -ErrorAction SilentlyContinue
}
}

9.2 Ensure required core services are running
SCRIPT IDEAS:
- $required = @("EventLog","DNS","NTDS","LanmanServer","LanmanWorkstation","W32Time")
- ForEach ($r in $required) {
$s = Get-Service -Name $r -ErrorAction SilentlyContinue
if ($s -and $s.Status -ne "Running") { Start-Service $r }
}

9.3 Optional: remove unused Windows Features
SCRIPT IDEAS:
- Get-WindowsFeature | Where-Object { $_.InstallState -eq "Installed" -and $_.Name -match "Telnet|SNMP" } |
ForEach-Object { Uninstall-WindowsFeature $_.Name -Restart:$false }

==========
PHASE 10 – SOFTWARE: UPDATES, UNINSTALL, BACKDOORS
==========

10.1 Trigger Windows Update scan (safe version)
SCRIPT IDEAS:
- Install-Module PSWindowsUpdate -Scope CurrentUser -Force (if allowed)
- Import-Module PSWindowsUpdate
- Get-WindowsUpdate -AcceptAll -Install -AutoReboot:$false
(If PSWindowsUpdate is not allowed or not present, at least set policy to auto-check.)

10.2 Update Firefox (if installed & required)
SCRIPT IDEAS:
- If you can't reliably script the internal updater, document manual step OR
- Use winget/choco if allowed (often not in CP), so safer to leave as manual.

10.3 Uninstall unwanted programs (BitTorrent, Wireshark-as-unwanted, etc.)
[oai_citation:6‡CP18-Server2019-Training2-Answer-Key.pdf](sediment://file_00000000c21871f8adae8fe44d9cc480)
SCRIPT IDEAS:
- $BadPrograms = @("BitTorrent","Wireshark","Burp Suite","Jellyfin","Games","Teamspeak")
- Get-WmiObject Win32_Product | Where-Object { $BadPrograms -contains $_.Name } |
ForEach-Object { $_.Uninstall() }
NOTE:
- Win32_Product is slow and can trigger MSI repairs; if possible, target specific uninstall strings from registry instead.

10.4 Delete hacking tools archives/backdoors
SCRIPT IDEAS:
- $badPaths = @(
"C:\Users\Public\Downloads\brutus-aet2-darknet.zip",
"C:\Users*\Downloads\shellshock-exploit.py",
"C:\Users\Public\Downloads*.zip"
)
- foreach ($p in $badPaths) { Remove-Item $p -Force -ErrorAction SilentlyContinue }

10.5 Scheduled tasks – remove obvious backdoors
SCRIPT IDEAS:
- Get-ScheduledTask | Where-Object { $_.TaskName -match "reverse|backdoor|shell" } |
Unregister-ScheduledTask -Confirm:$false

==========
PHASE 11 – PROHIBITED FILES, PERMISSIONS, LOGGING
==========

11.1 Remove prohibited media and sensitive files
SCRIPT IDEAS:
- Get-ChildItem -Path C:\Users -Include *.mp3,*.mp4,*.avi -Recurse -ErrorAction SilentlyContinue |
Remove-Item -Force
- Search for plaintext password files:
- Get-ChildItem -Path C:\Users -Include *.txt,*.doc*,*.xlsx -Recurse -ErrorAction SilentlyContinue |
Select-String -Pattern "password","creds","p@ss" |
ForEach-Object { Remove-Item $_.Path -Force }

11.2 Tighten permissions on webroots and sensitive folders
SCRIPT IDEAS:
- $SensitiveFolders = @(
"C:\Windows\SYSVOL",
"C:\Windows\NTDS",
"C:\Windows\System32\DNS"
)
- foreach ($f in $SensitiveFolders) {
if (Test-Path $f) {
icacls $f /inheritance:e
icacls $f /grant:r "SYSTEM:(F)" "Administrators:(F)"
icacls $f /remove:g "Everyone"
}
}

11.3 Event log size & overwrite policy
SCRIPT IDEAS:
- Using wevtutil:
- wevtutil sl Security /ms:20971520 # 20 MB
- wevtutil sl Application /ms:10485760
- wevtutil sl System /ms:10485760

==========
PHASE 12 – FINAL CHECKS AND LOGGING
==========

12.1 Summarize changes in log
SCRIPT IDEAS:
- At the end of the script:
- Add-Content $Log "Hardening completed at $(Get-Date)"
- Optionally dump key info:
- Add-Content $Log "Firewall: $(netsh advfirewall show allprofiles state)"
- Add-Content $Log "SMB1: $(Get-SmbServerConfiguration | Select EnableSMB1Protocol | Out-String)"

12.2 Simple “sanity checks” your script can print
SCRIPT IDEAS:
- Write-Host to screen at the end:
- RDP status, Firewall status, lockout settings:
- net accounts
- netsh advfirewall show allprofiles state
- Get-SmbServerConfiguration | Select EnableSMB1Protocol
- On DC:

- Verify DNS service running:
  - Get-Service dns

12.3 Make sure script is safe to re-run
- All commands are set-based, not blindly additive (no duplicate group members, no repeated share creation).
- Deletion commands always scoped to known bad objects, not wildcards that might hit legitimate items.

==========
END OF SERVER 2019 SCRIPT-ORIENTED CHECKLIST
==========