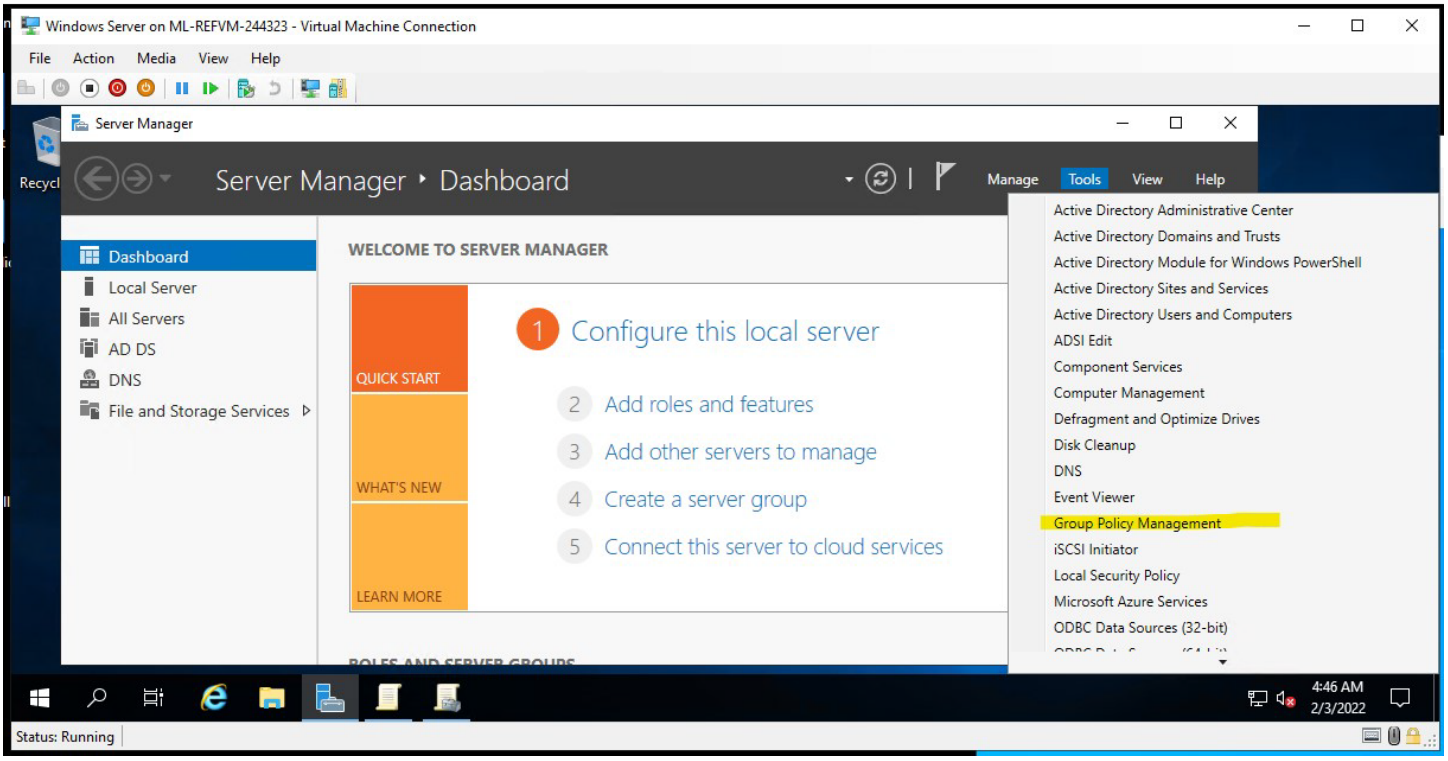
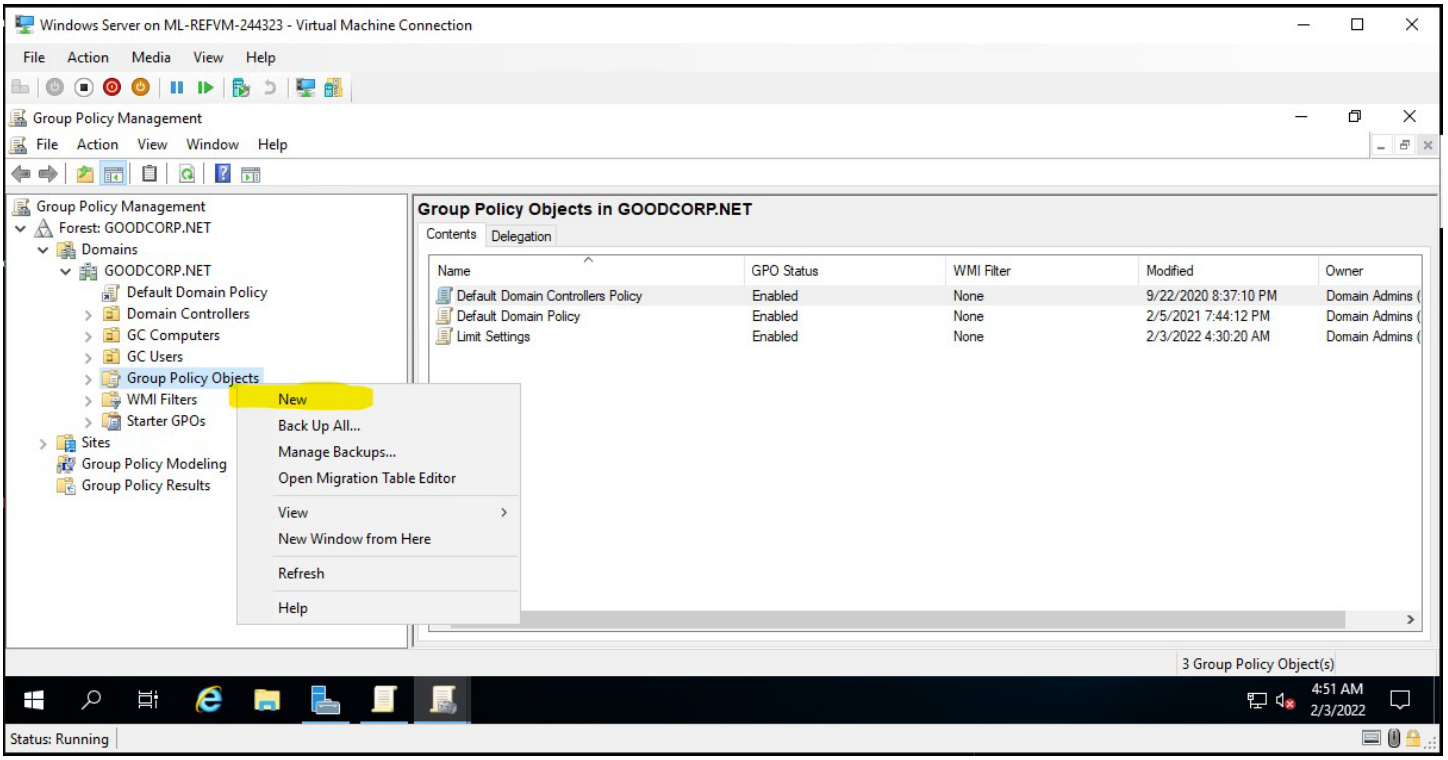
Task 1: Create a GPO: Disable Local Link Multicast Name Resolution

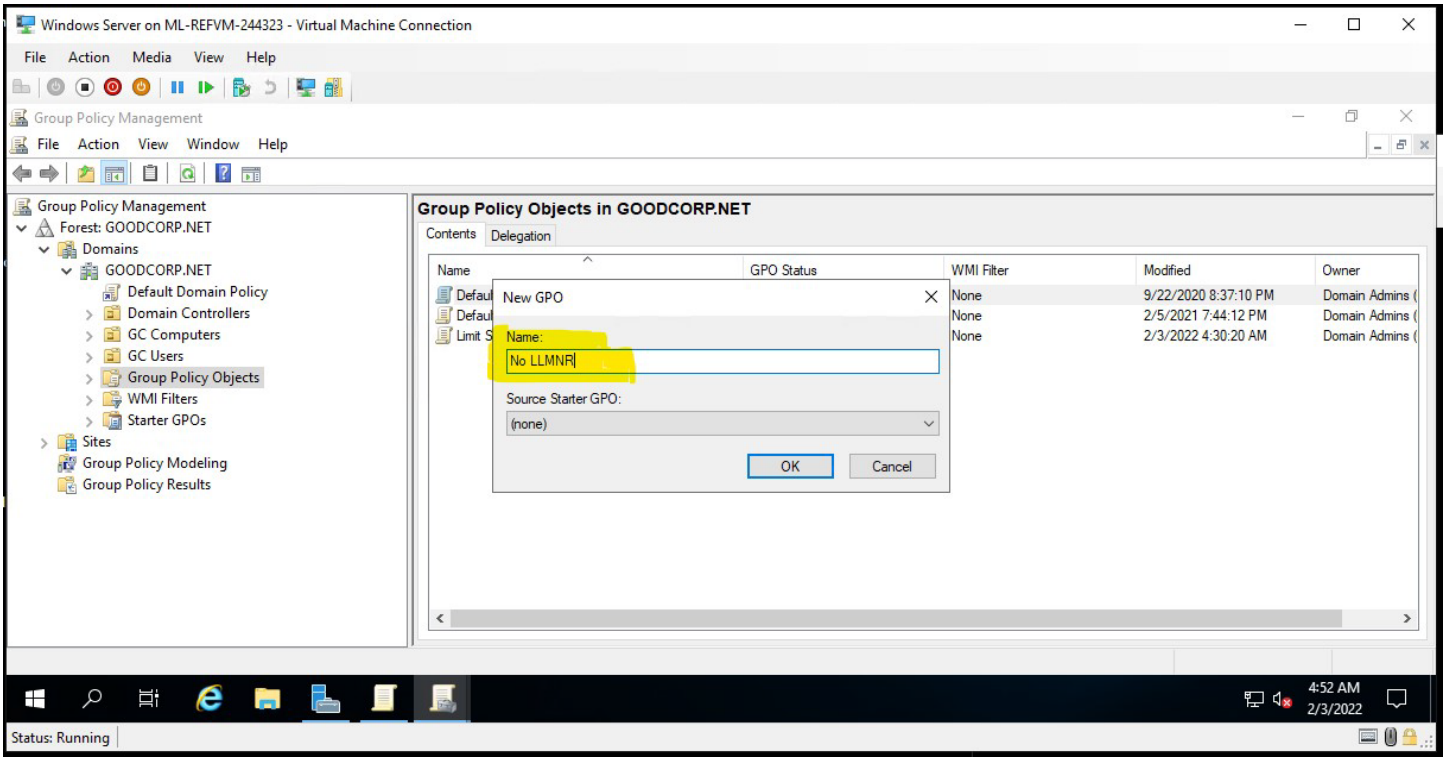
1. On the top-right of the Server Manager screen, open the **Group Policy Management** tool to create a new **GPO**.



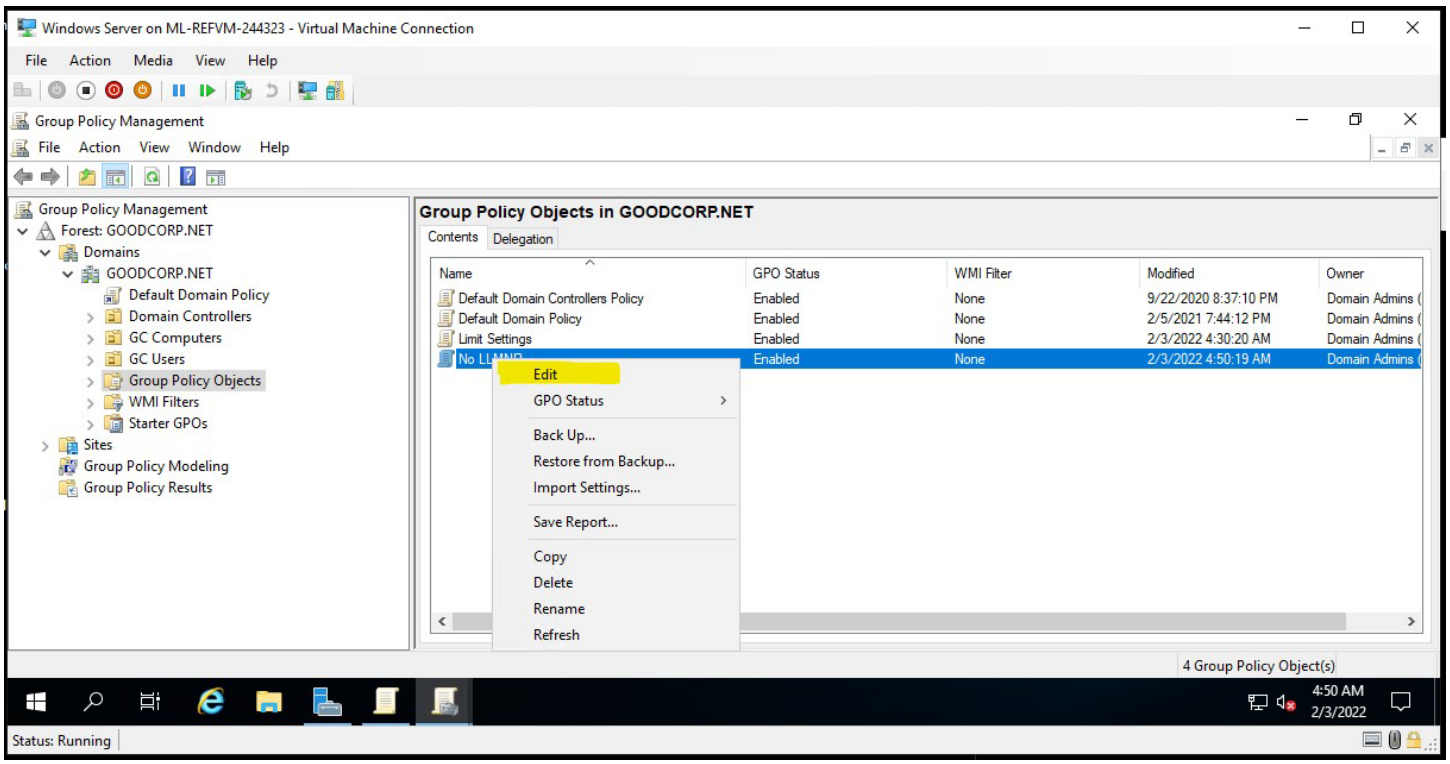
2. Right-click **Group Policy Objects** and select **New.**



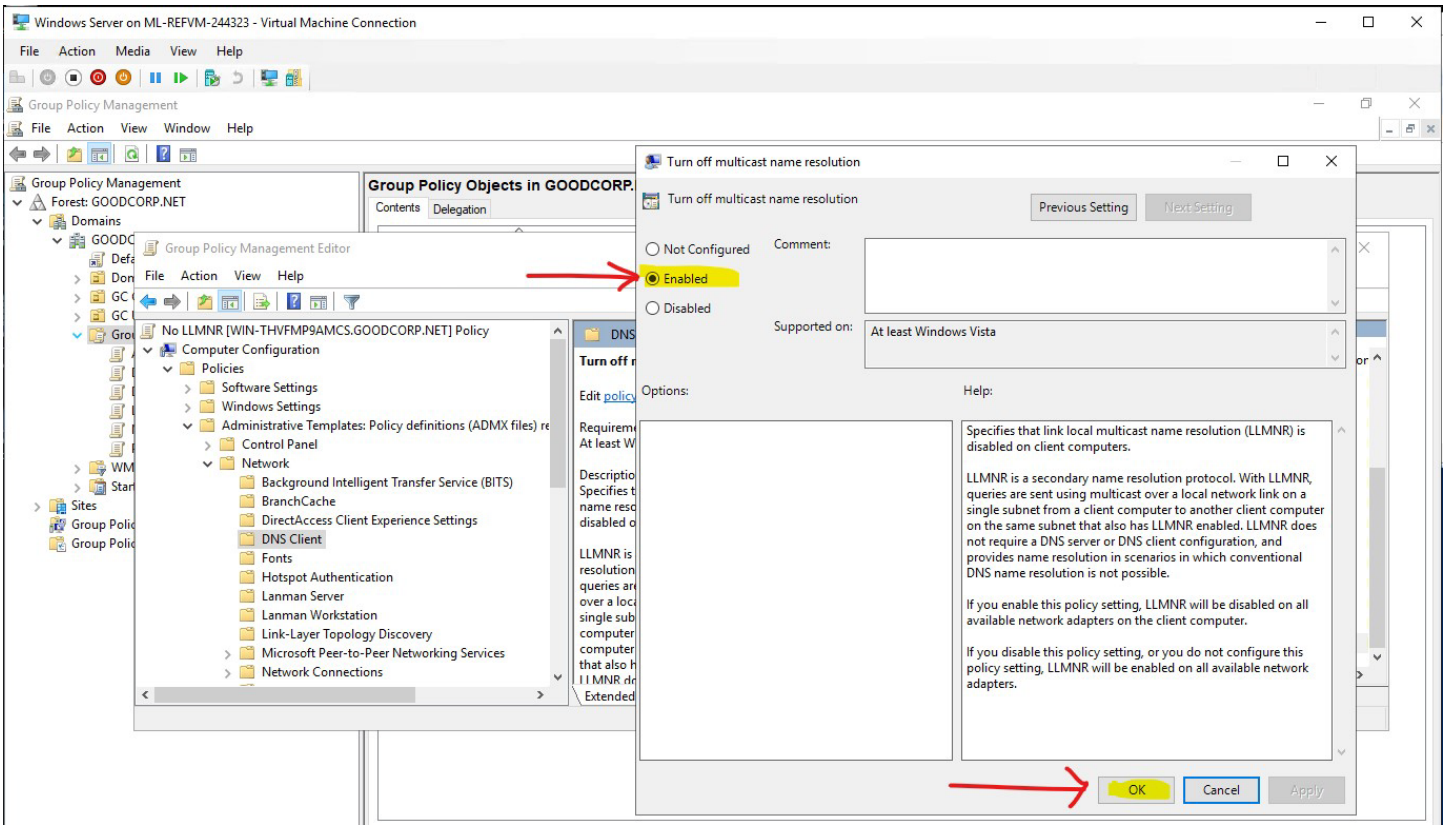
3. Name the Group Policy Object ***No LLMNR*.**

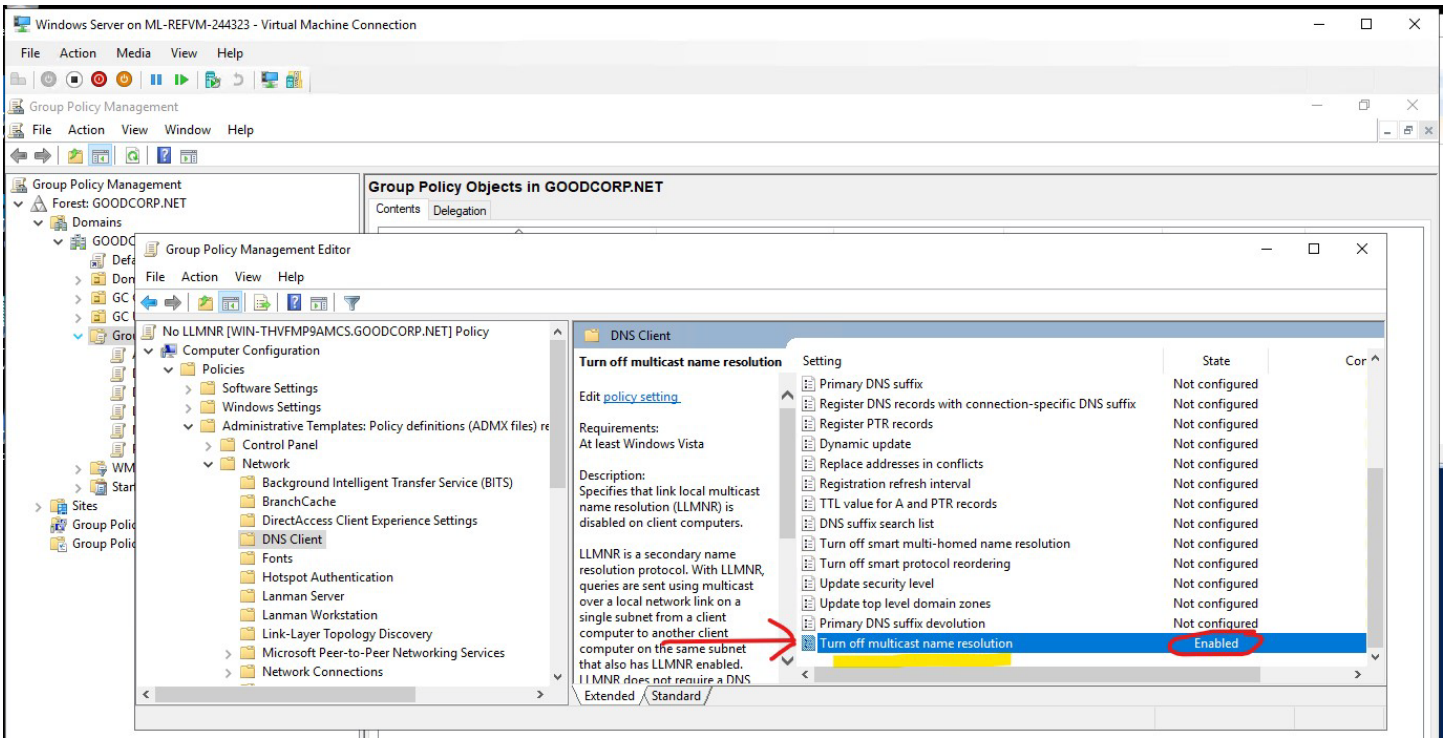


4. Right-click the new **No LLMNR** GPO listing and select **Edit** to open the **Group Policy Management Editor** and find policies.

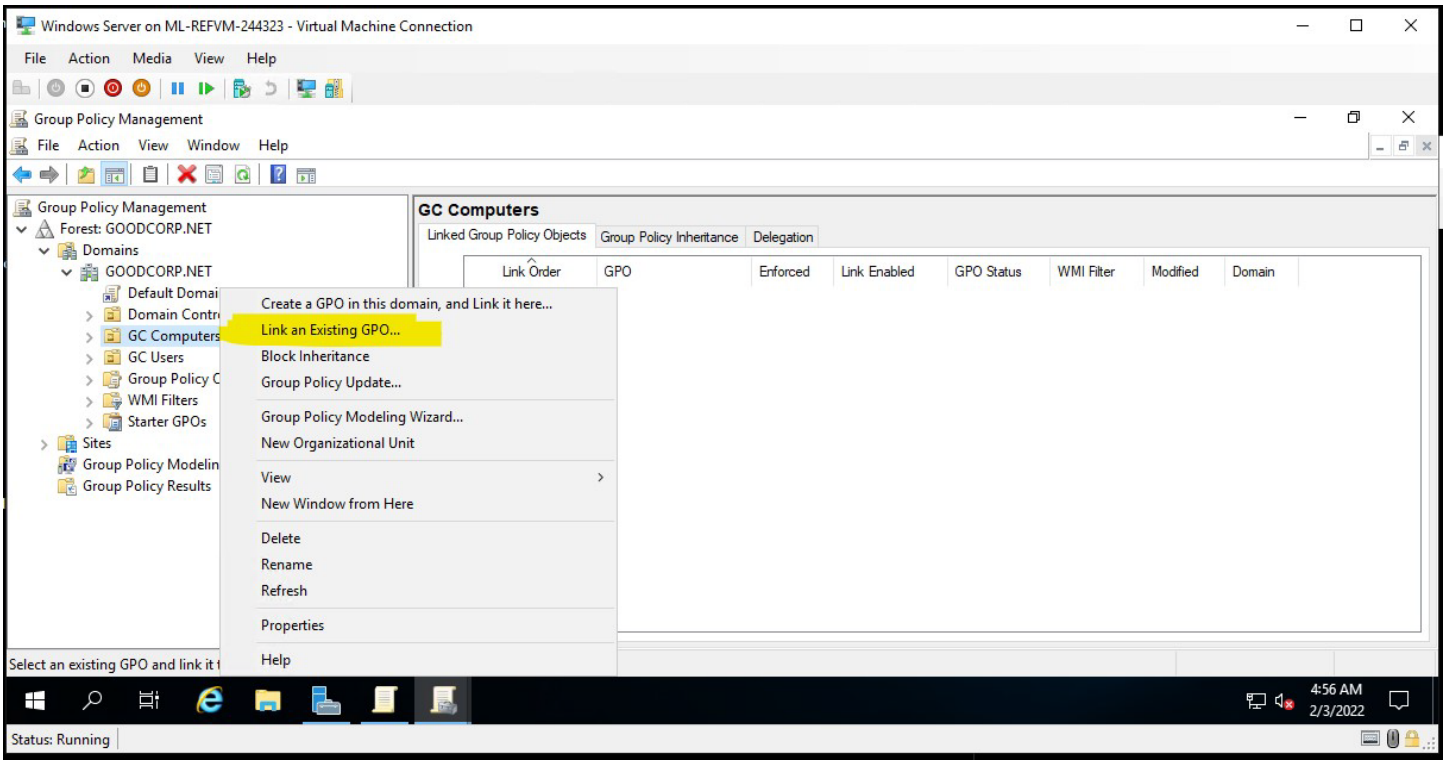


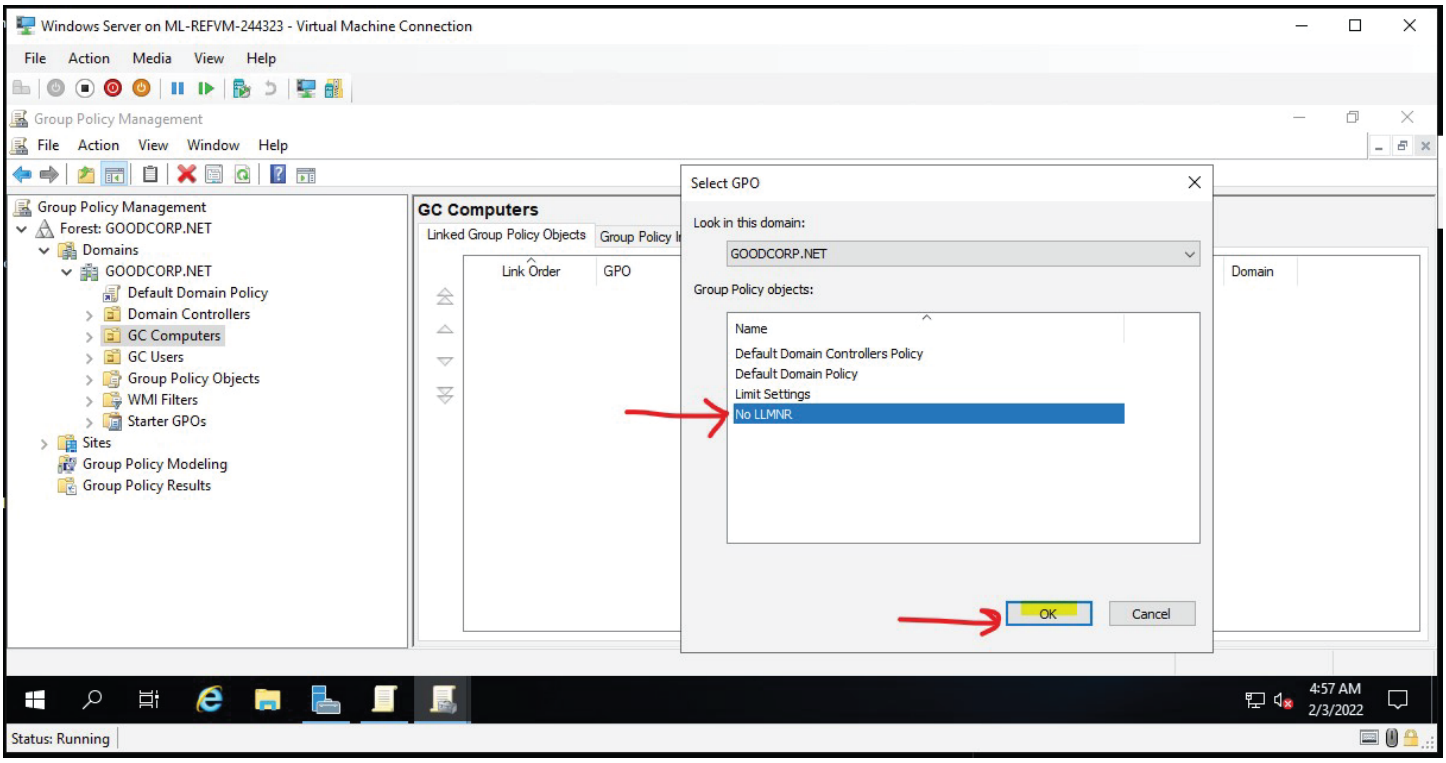
5. In the Group Policy Management Editor, the policy you are looking for is at the following path:
Computer Configuration\Policies\Administrative Templates\Network\DNS Client .
-Find the policy called ***Turn Off Multicast Name Resolution*** .
-Enable this policy.

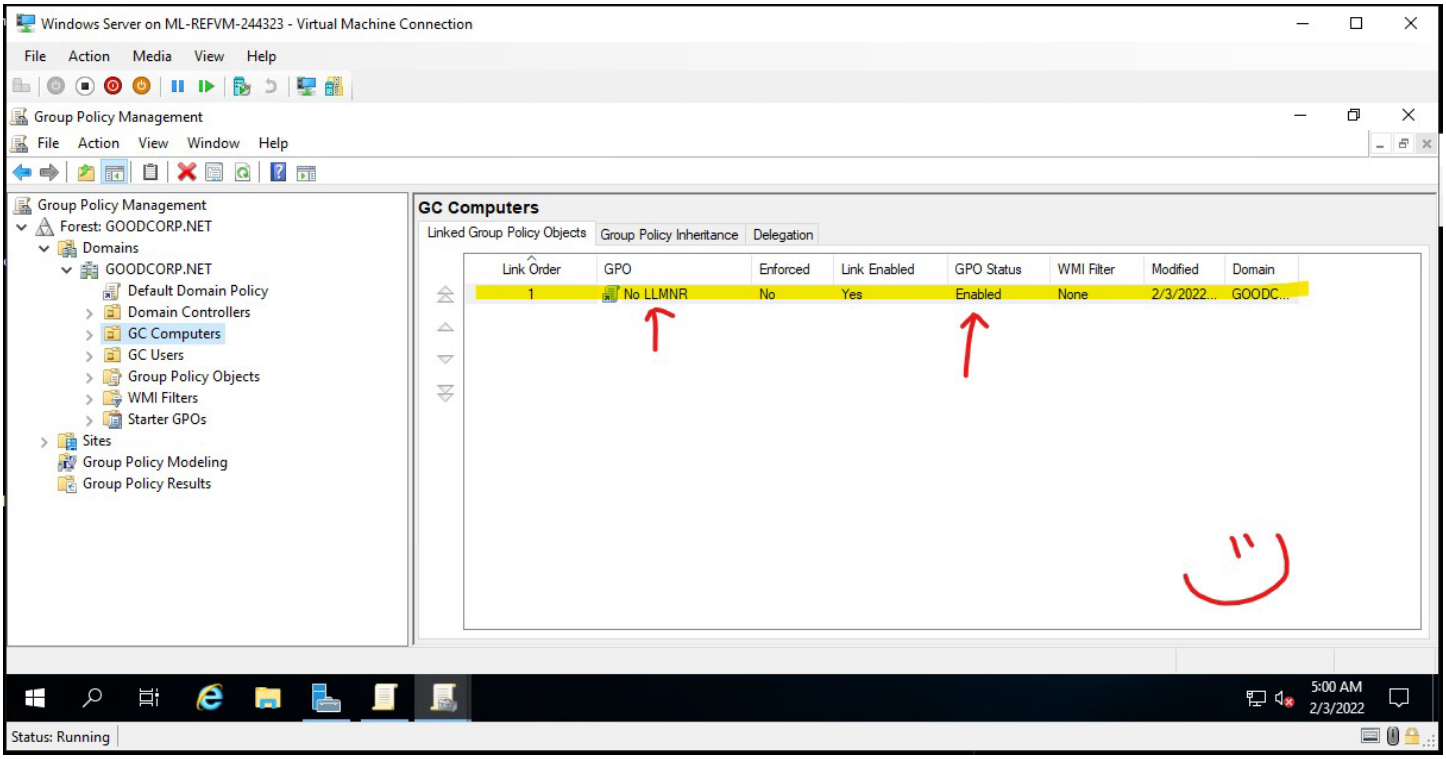




6. Exit the **Group Policy Management Editor** and link the GPO to the ***GC Computers*** organizational unit you previously created.



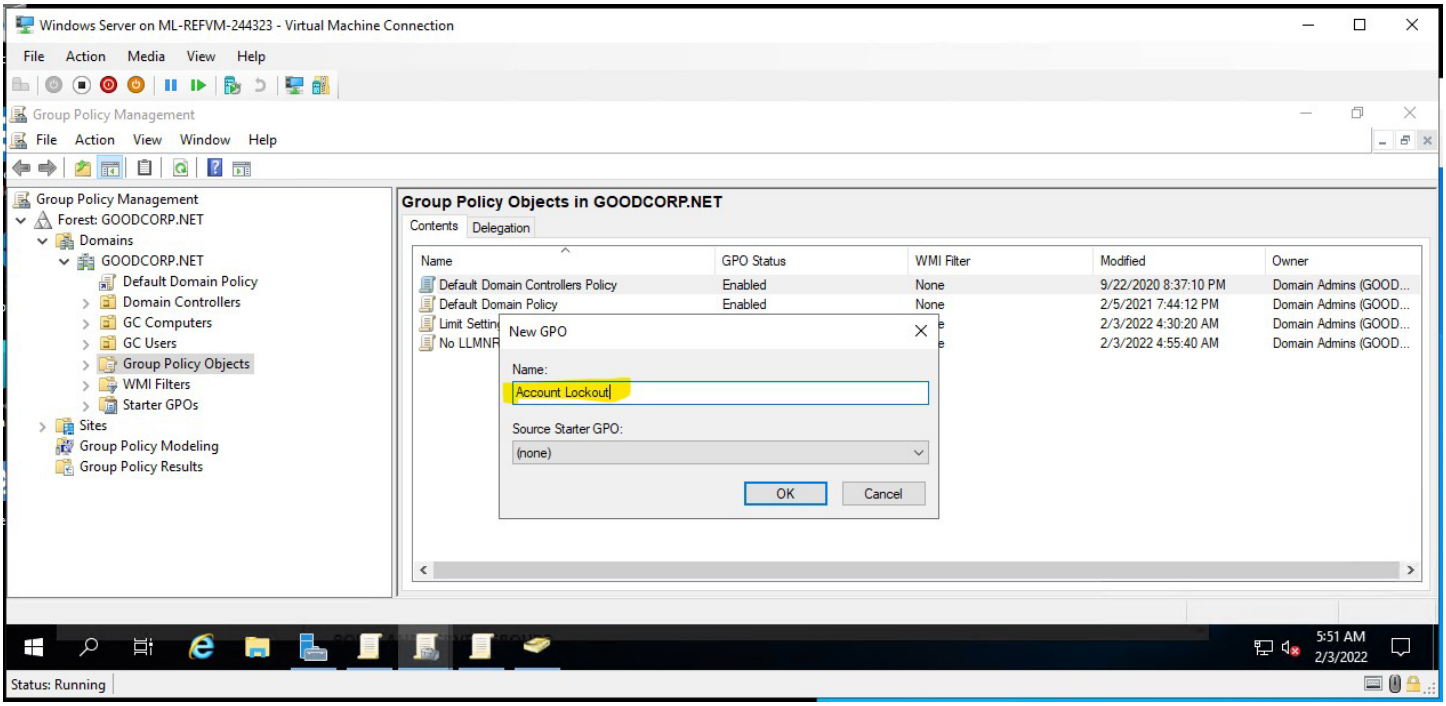




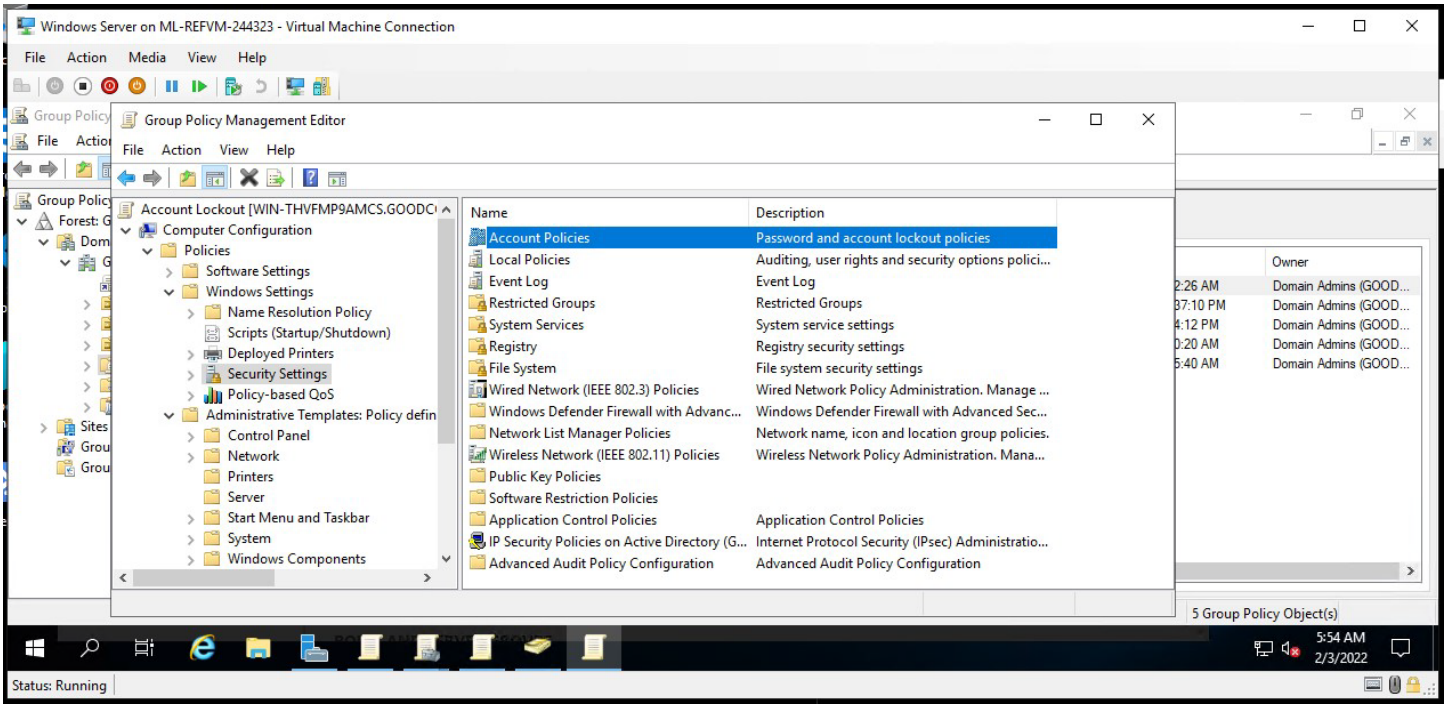
Task 2: Create a GPO: Account Lockout

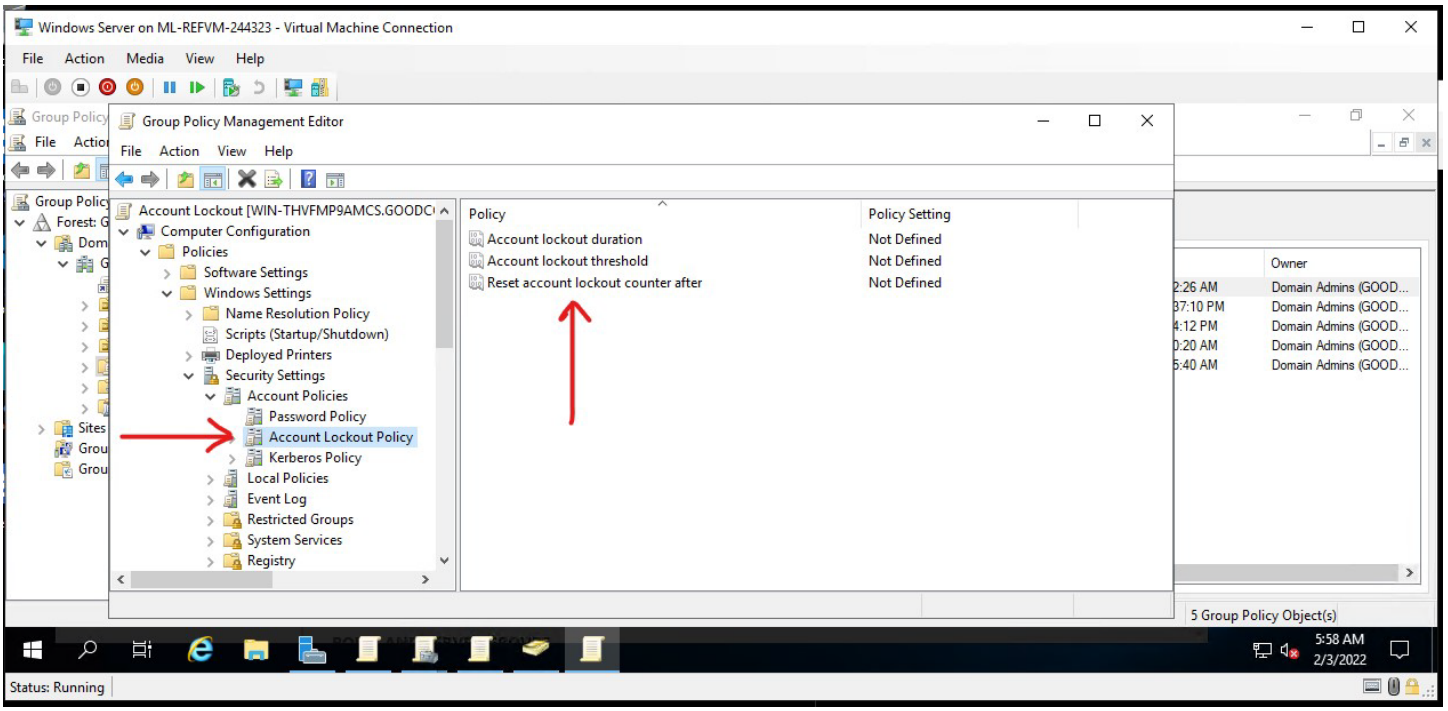
You'll be working within in your nested Windows Server machine again to create another **Group Policy Object**.
Create what you believe to be a reasonable account lockout Group Policy for the Windows 10

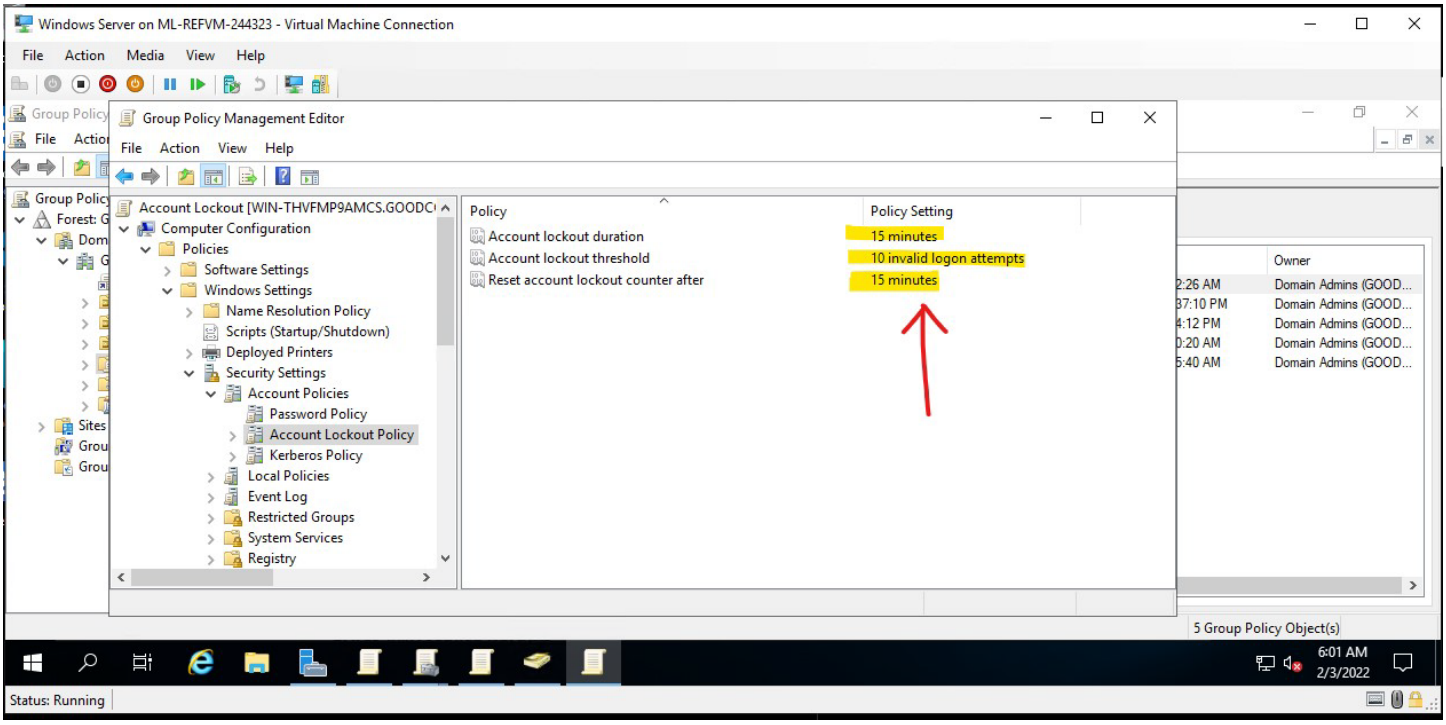
1. Name the Group Policy Object ***Account Lockout*** .



2. You can use **Microsoft's 10/15/15 recommendation** if you'd like.



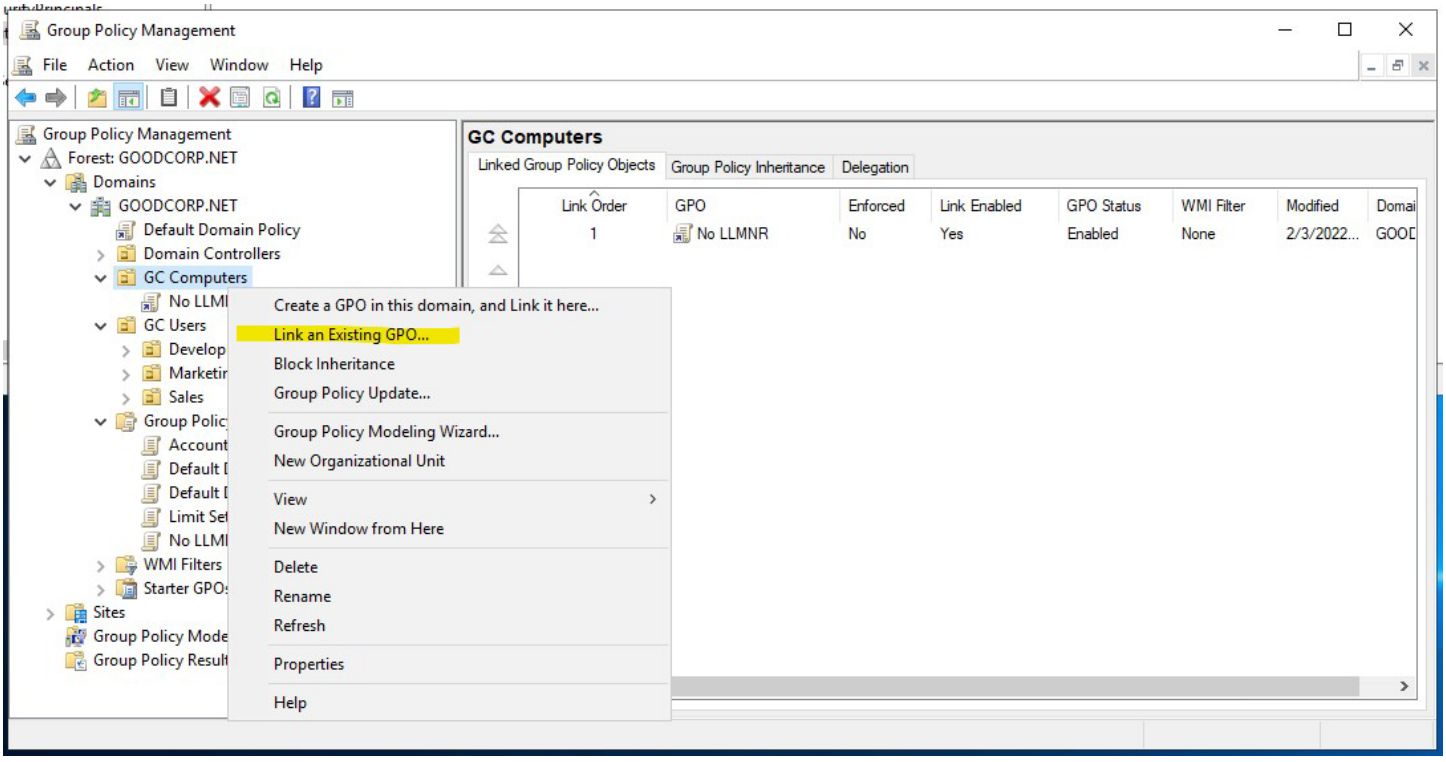


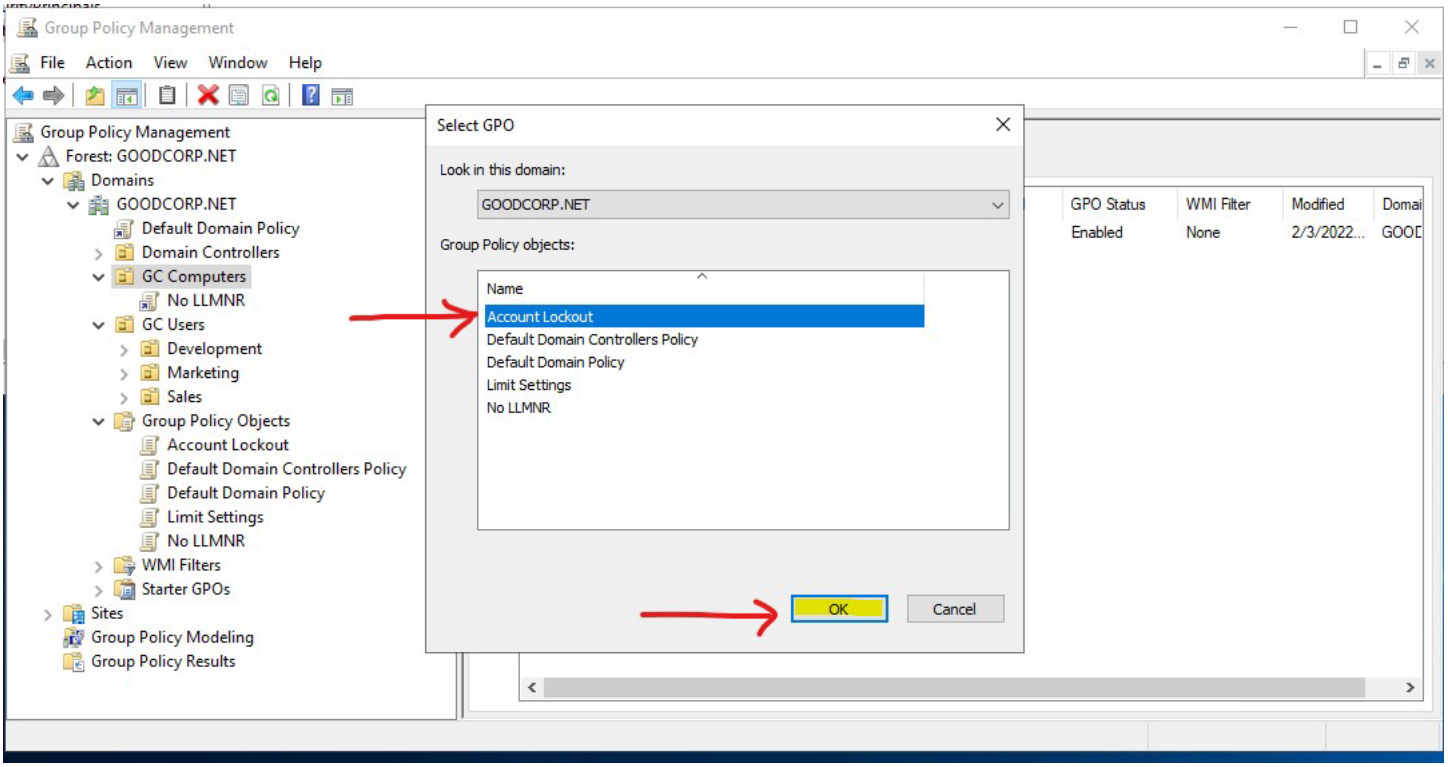


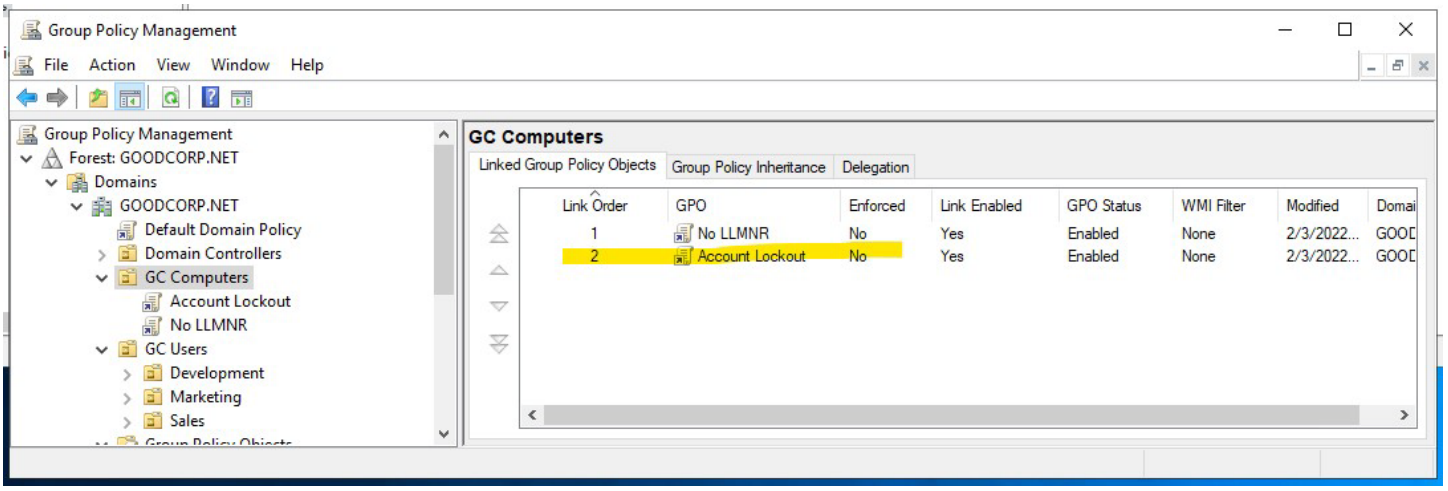
3. When editing policies for this new GPO, keep in mind that you're looking for ***computer configuration*** policies to apply to your ***GC Computers*** OU. Also, these policies involve ***Windows security settings*** and ***accounts***.

Yep, should have told me in #2 but I found it. :)

4. Don't forget to link the GPO to your ***GC Computers*** organizational unit.

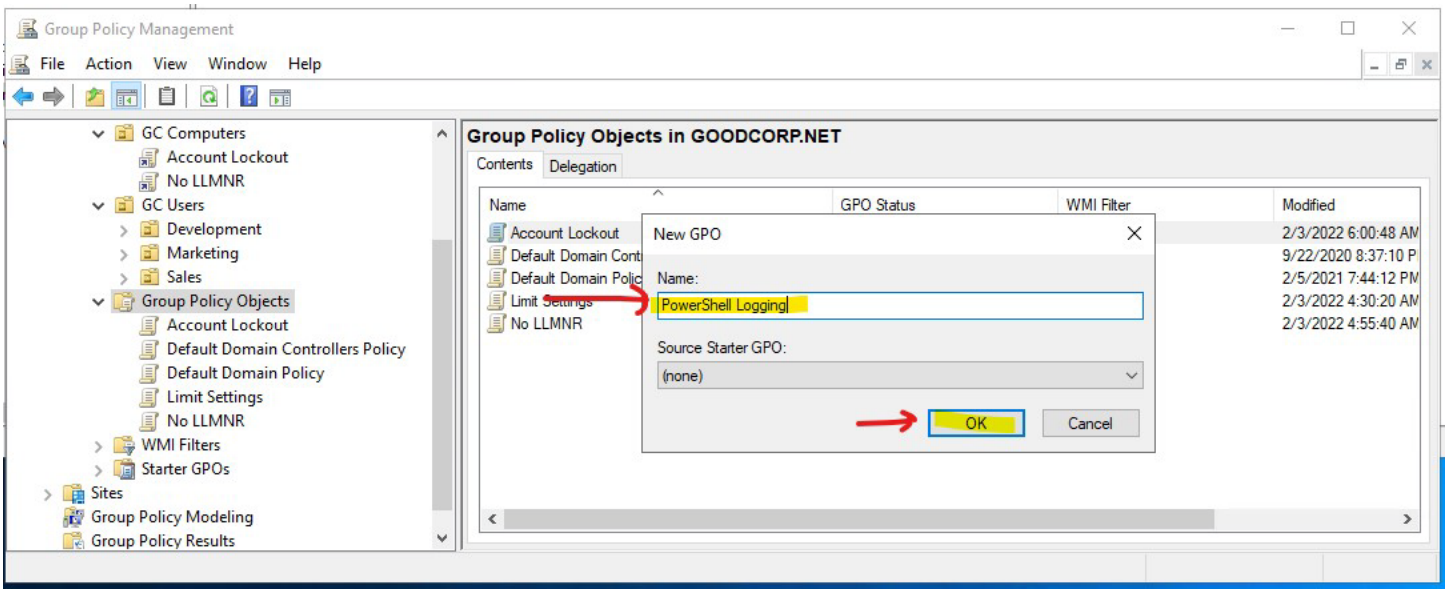


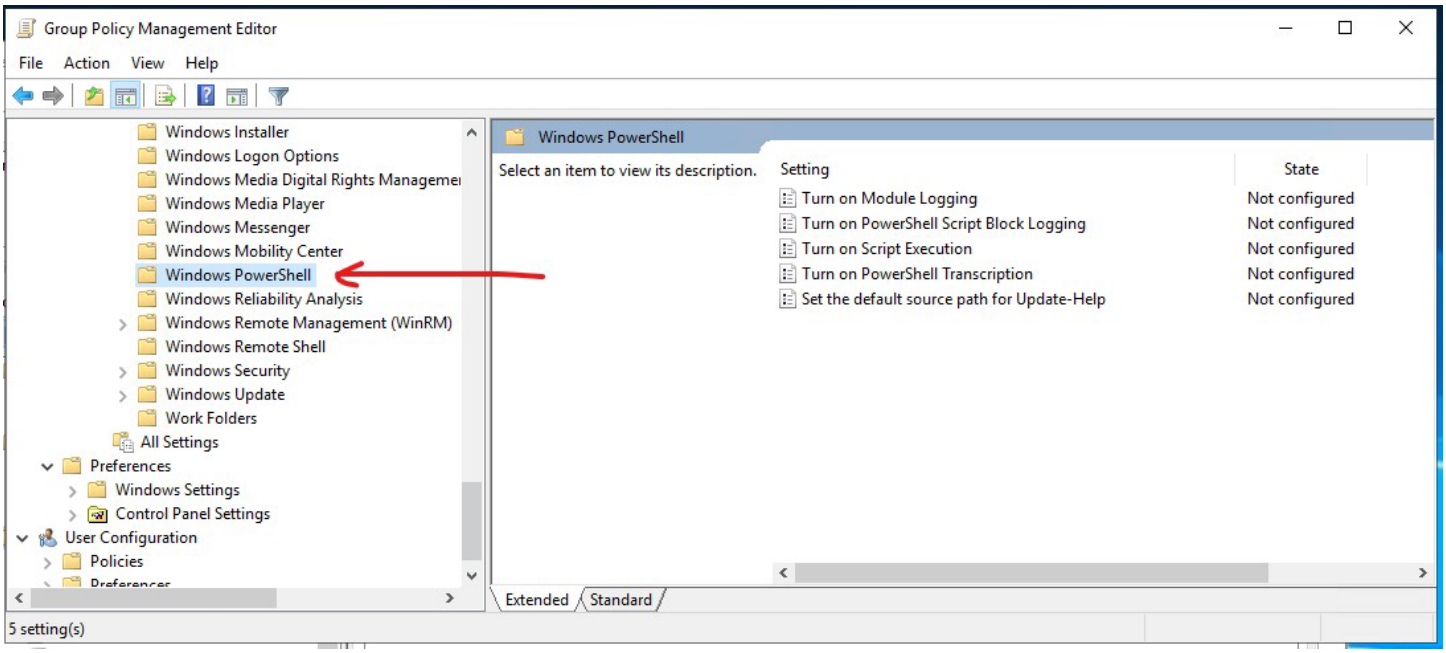

Group Policy Management
File Action View Window Help
Group Policy Management
Forest: GOODCORP.NET
Domains
GOODCORP.NET
Default Domain Policy
Domain Controllers
GC Computers
No LLMNR
GC Users
Development
Marketing
Sales
Group Policy Objects
Account Lockout
Default Domain Controllers Policy
Default Domain Policy
Limit Settings
No LLMNR
WMI Filters
Starter GPOs
Sites
Group Policy Modeling
Group Policy Results
Select GPO
Look in this domain:
GOODCORP.NET
Group Policy objects:
Name
Account Lockout
Default Domain Controllers Policy
Default Domain Policy
Limit Settings
No LLMNR
OK
Cancel
GPO Status
WMI Filter
Modified
Enabled
None
2/3/2022...


Group Policy Management
File Action View Window Help
Group Policy Management
Forest: GOODCORP.NET
Domains
GOODCORP.NET
Default Domain Policy
Domain Controllers
GC Computers
Account Lockout
No LLMNR
GC Users
Development
Marketing
Sales
GC Computers
Linked Group Policy Objects
Group Policy Inheritance
Delegation
Link Order
GPO
Enforced
Link Enabled
GPO Status
WMI Filter
Modified
1
No LLMNR
No
Yes
Enabled
None
2/3/2022...
2
Account Lockout
No
Yes
Enabled
None
2/3/2022...

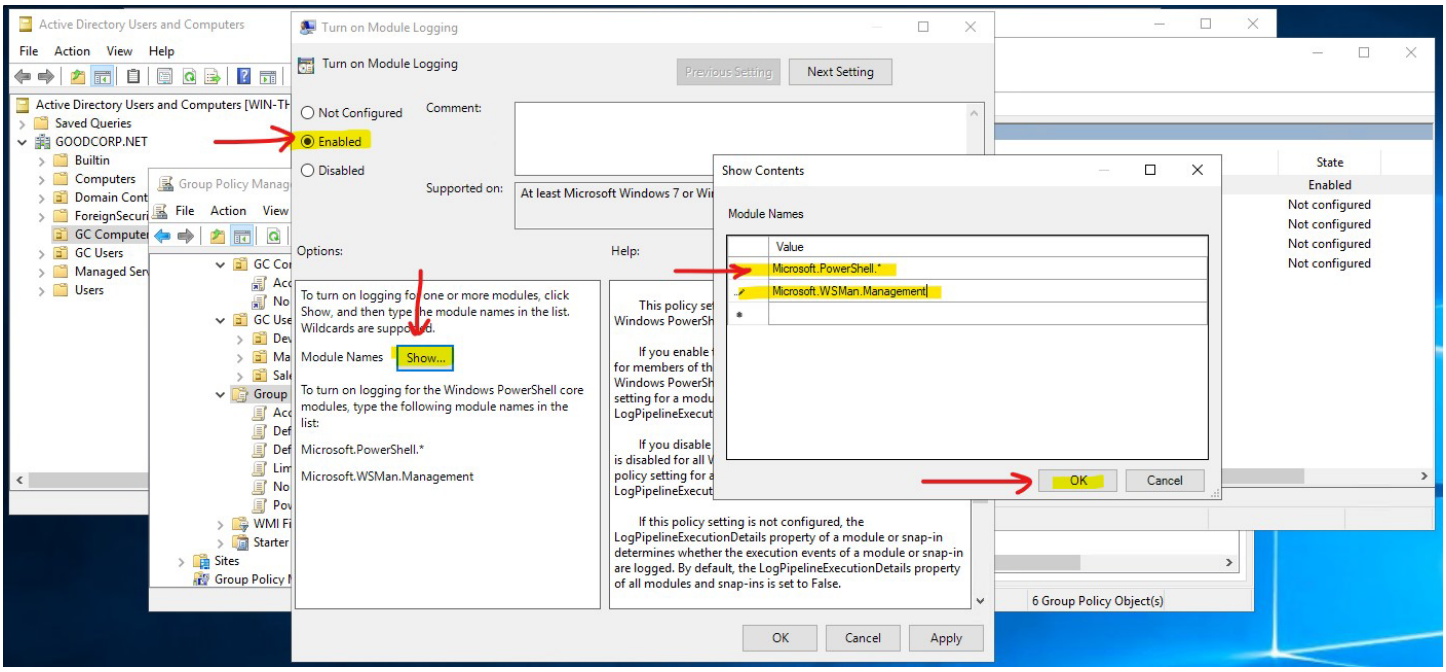
Task 3: Create a GPO: Enabling Verbose PowerShell Logging and Transcription

1. Name the Group Policy Object ***PowerShell Logging.***
   -Find the proper Windows Powershell policy in **Group Policy Management Editor**.
   -Hint: Check out the ***computer configuration, administrative templates, and Windows component directories.***



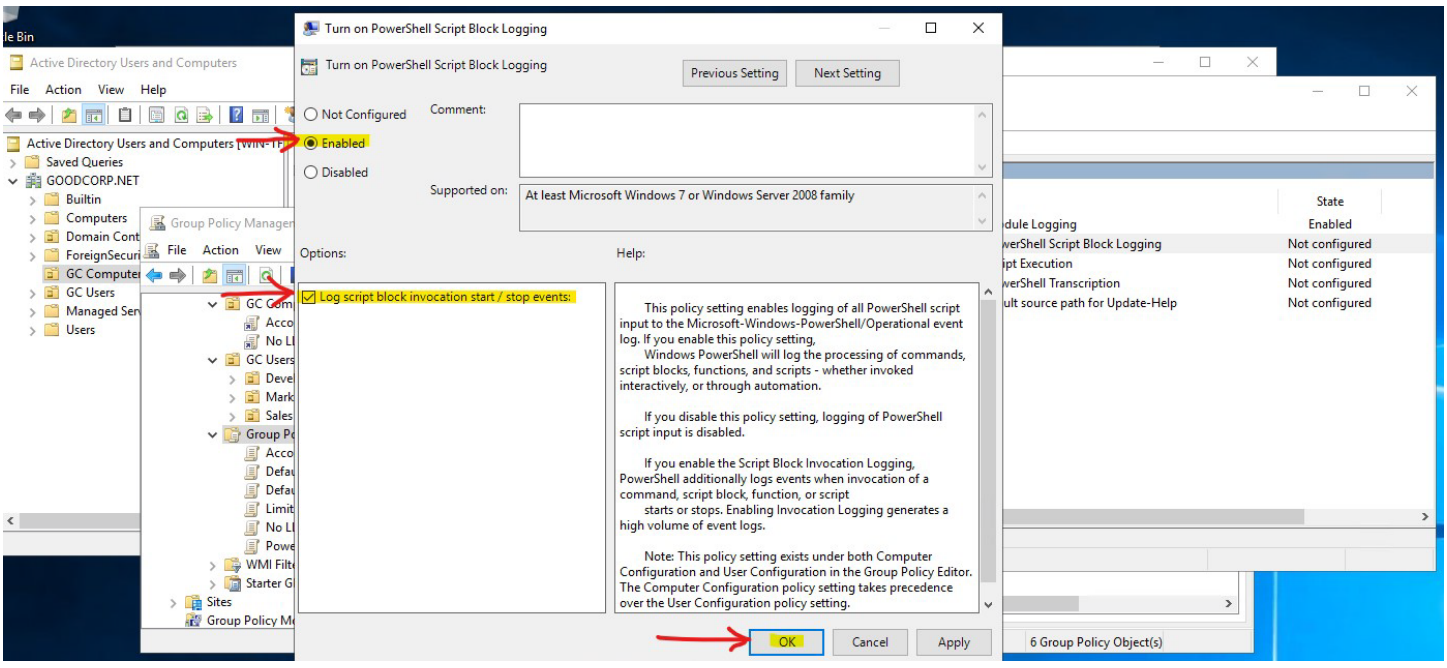


2. Enable the ***Turn on Module Logging*** and do the following:
   -Click ***Show*** next to ***Module Names***.
   -Since we want to log ***all*** PowerShell modules, enter an asterisk * (wildcard) for the ***Module Name,*** then click ***OK***.



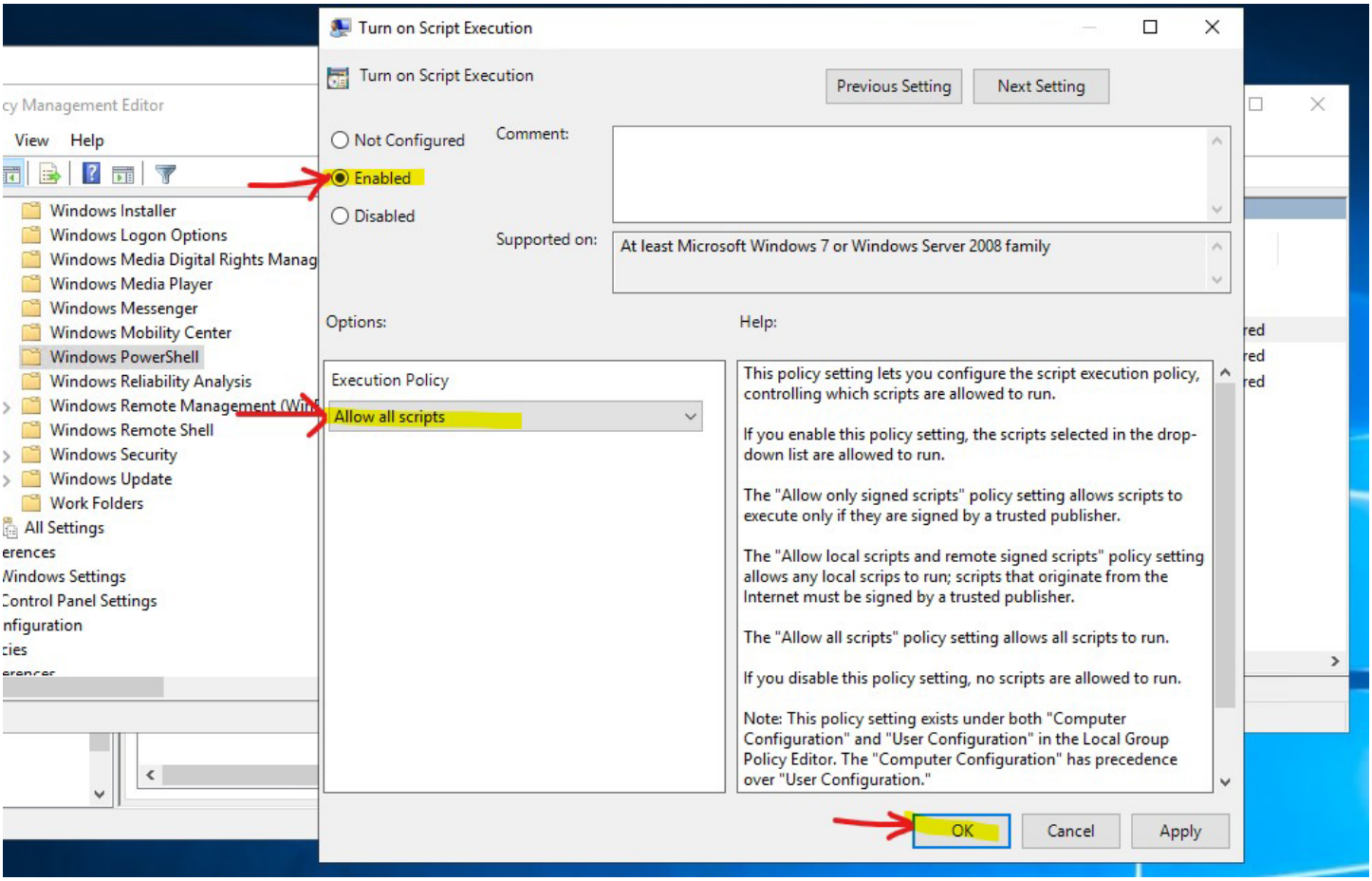
3. Enable the ***Turn on PowerShell Script Block Logging*** policy.

   -Make sure to check the ***Log script block invocation start/stop events:*** setting.



4. Enable the ***Turn on Script Execution*** policy and do the following:

Set **Execution Policy to Allow all scripts.**



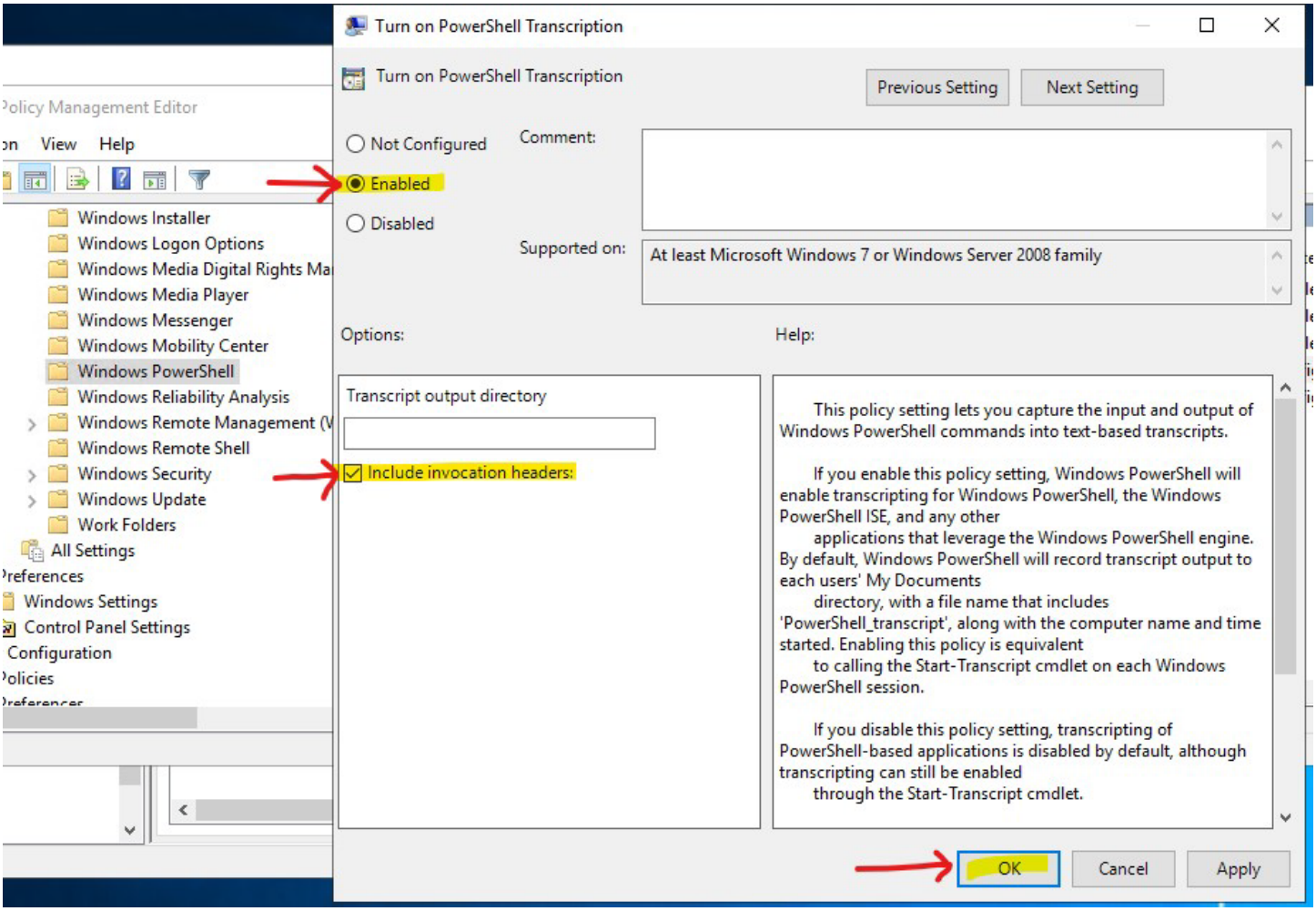
Note: Do you remember the Set-ExecutionPolicy cmdlet we ran during the PowerShell exercises? This policy can enforce those settings as part of a GPO.

5. Enable the ***Turn on PowerShell Transcription*** policy and do the following:

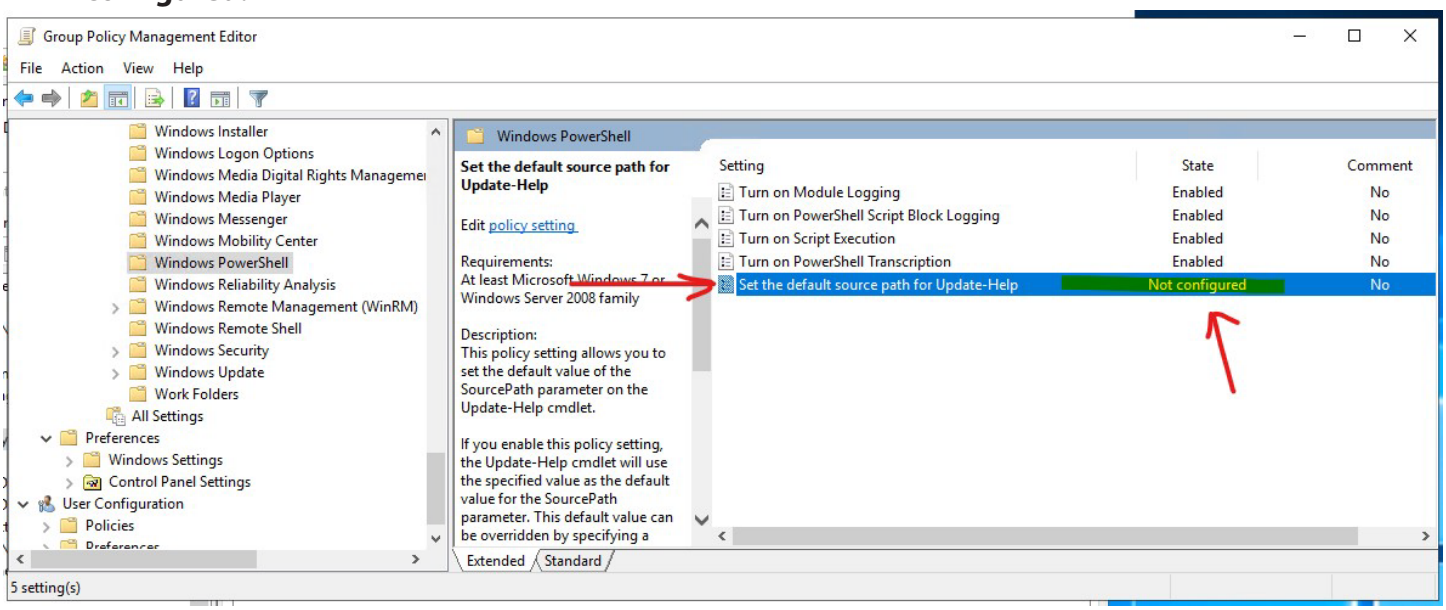
-Leave the **Transcript output directory** blank (this defaults to the user's ~\Documents directory).

Note: "Transcription" means that an exact copy of the the commands are created in an output directory.

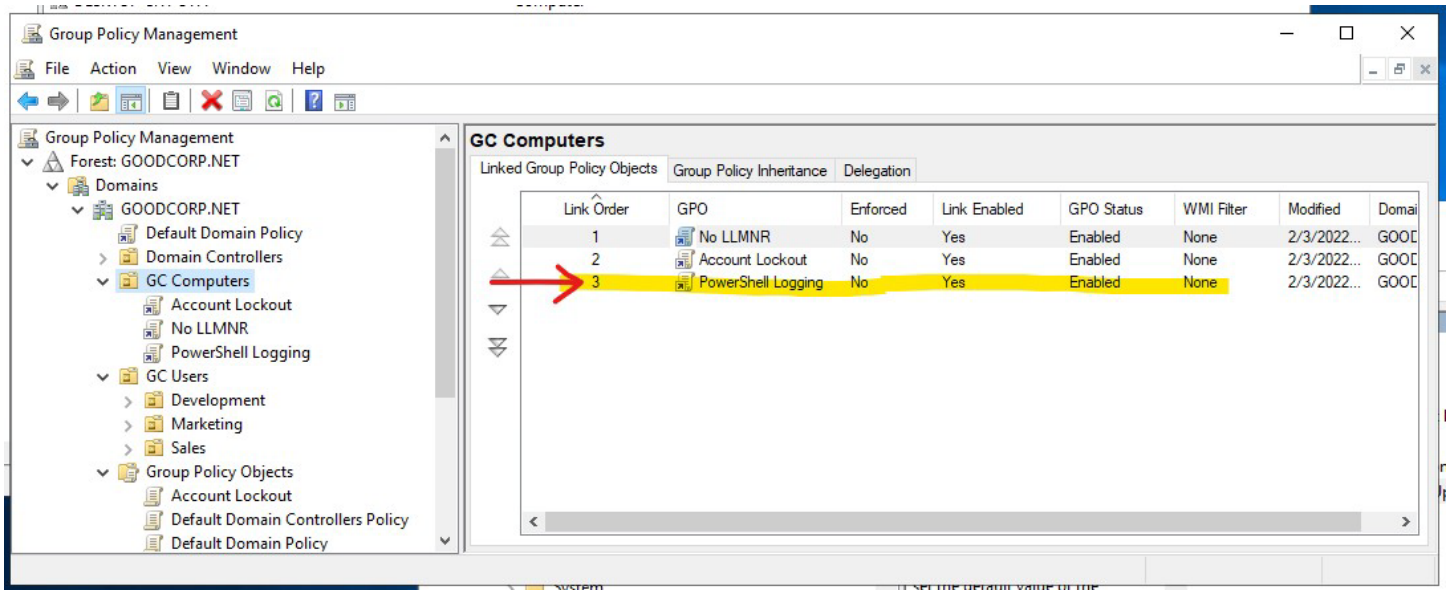
-Check the **Include invocation headers** option. This will add timestamps to the command transcriptions.

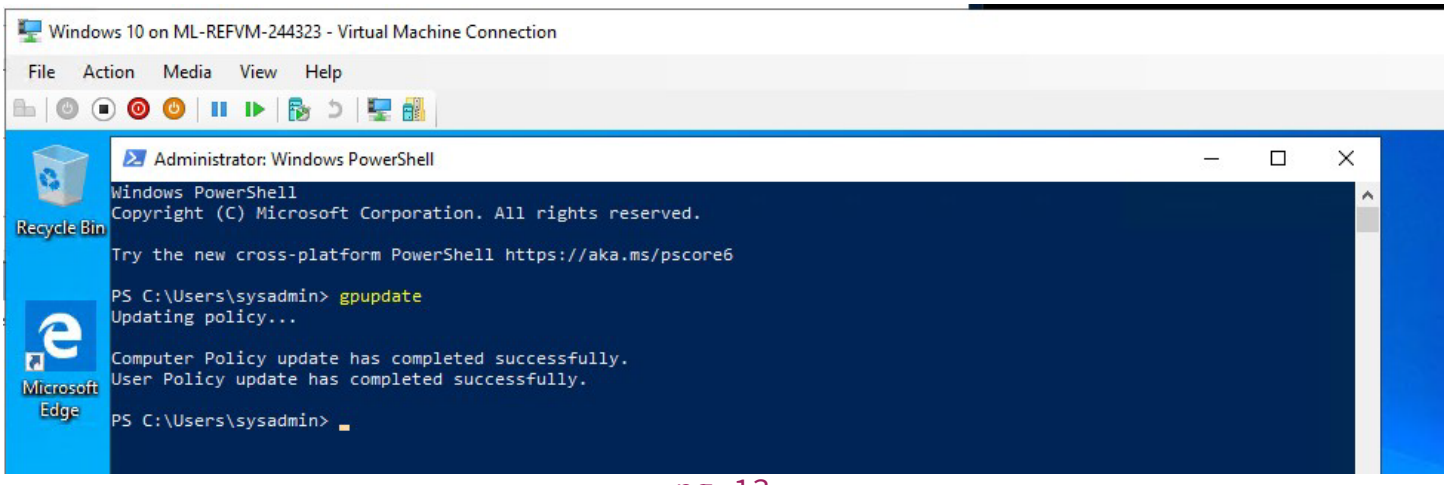


6. Leave the ***Set the default source path for Update-Help*** policy as **Not configured.**



7. Link this new ***PowerShell Logging*** GPO to the ***GC Computers*** OU.





Task 4: Create a Script: Enumerate Access Control Lists

Familiarize yourself with the basics of ***Get-Acls*** :

-***Get-Acl*** without any parameters or arguments will return the security descriptors of the directory you're currently in.
-***Get-Acl <filename>*** will return the specific file's ACL. We'll need to use this for our task.

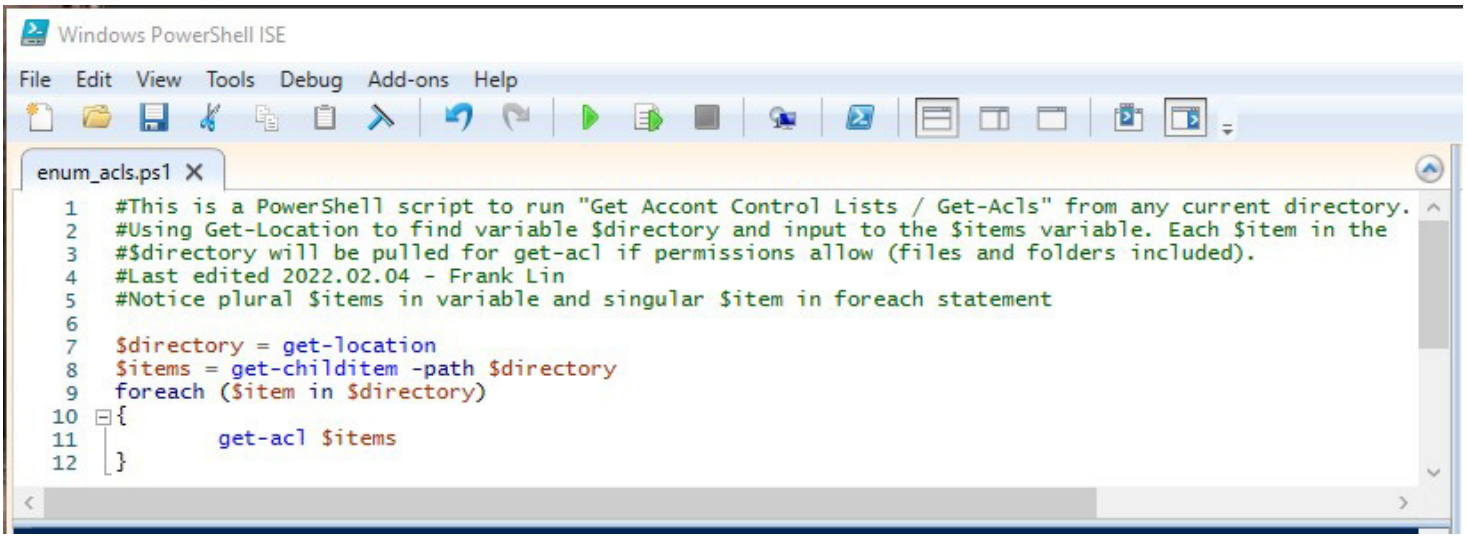
Create a PowerShell script that will enumerate the Access Control List of each file or subdirectory within the current working directory.

1. Create a ***foreach*** loop. You can use the following template:

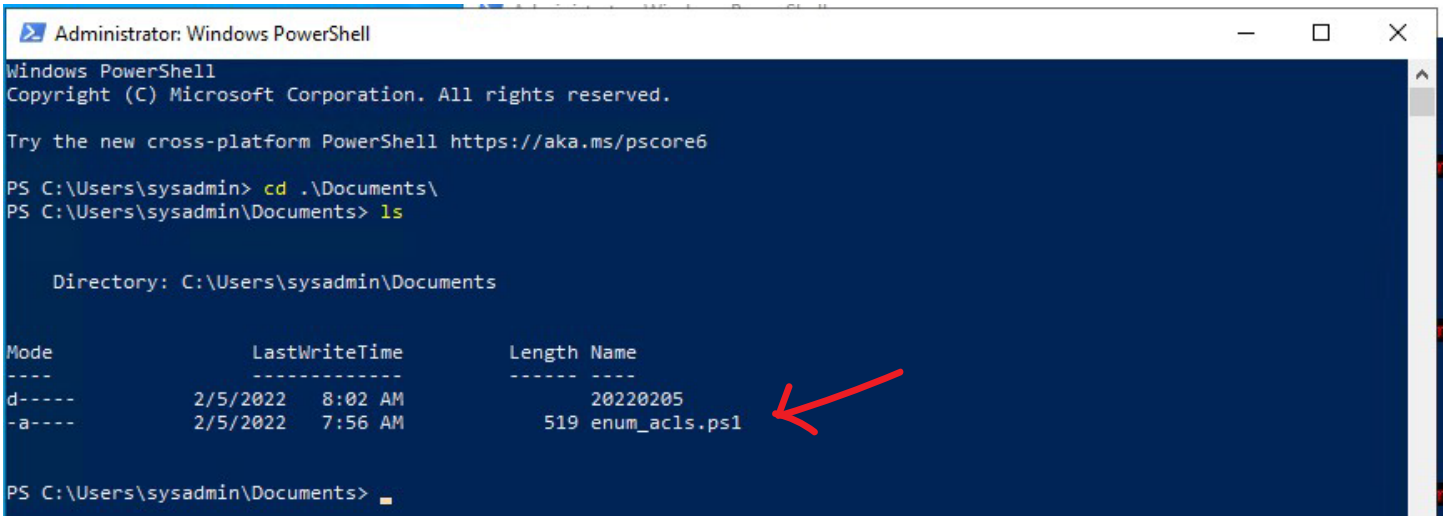
foreach ($item in $directory) {
<Script block>
}

2. Above the ***foreach*** condition, set a variable, ***$directory*** , to the contents of the current directory.

3. Replace the script block placeholder with the command to enumerate the ACL of a file, using the ***$item*** variable in place of the file name.
You'll need to use the following cmdlets:
-***Get-ChildItem*** (or any alias of ***Get-ChildItem*** , such as ***ls*** or ***dir***)
-***Get-Acl***

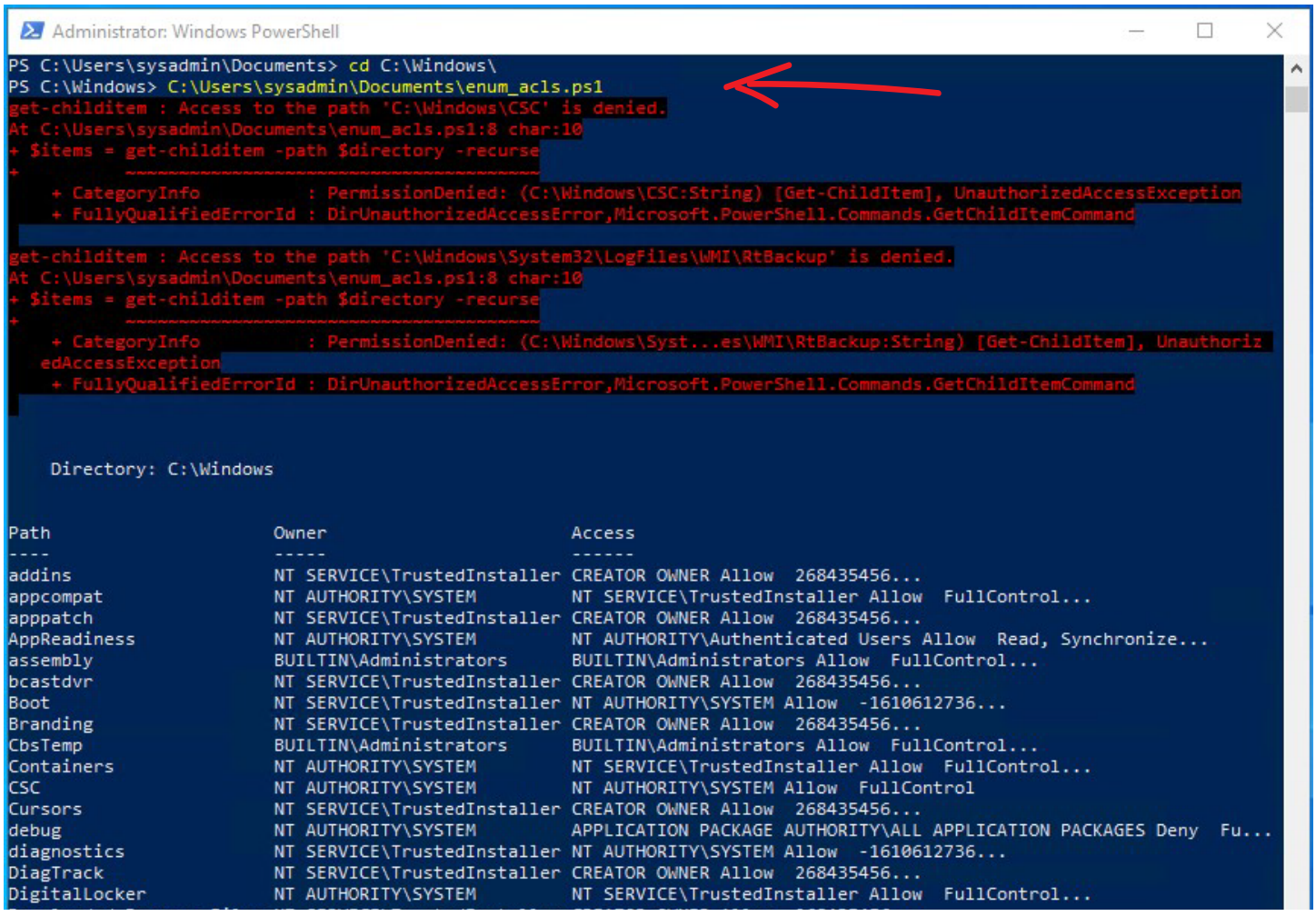


4. Save this script in ***C:\Users\sysadmin\Documents*** as ***enum_acls.ps1*** .



5. Test this script by moving to any directory (***cd C:\Windows***), and running ***C:\Users\sysadmin\Documents\enum_acls.ps1*** (enter the full path and file name).

You should see the ACL output of each file or subdirectory where you ran the script from.

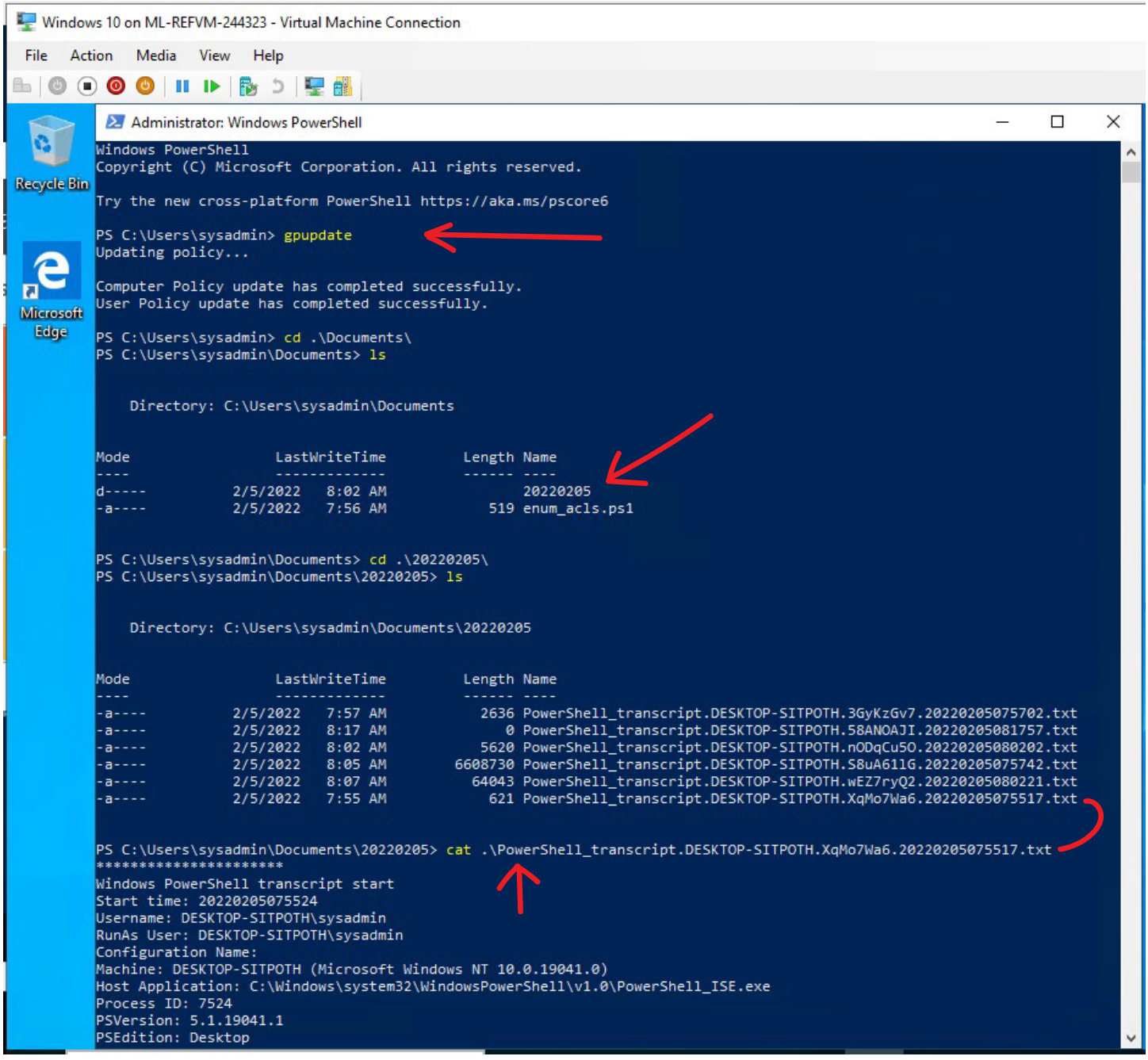


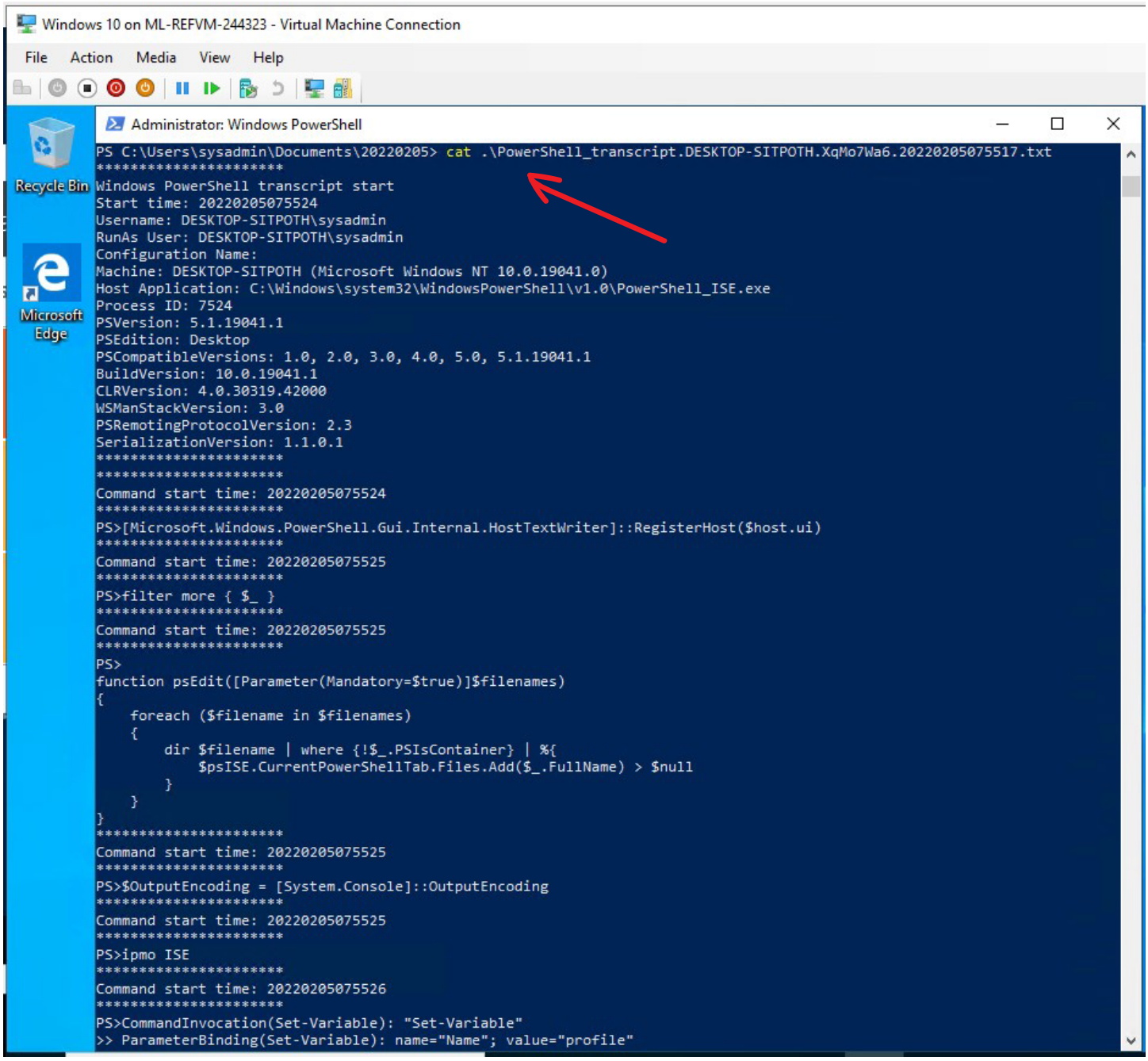
Bonus Task 5: Verify Your PowerShell Logging GPO

For this task we'll want to test and verify that our PowerShell logging GPO is working properly.

Instructions

-Ensure you're logged into the Windows 10 machine as ***sysadmin | cybersecurity*** .
-Run ***gpupdate*** in an administrative PowerShell window to pull the latest Active Directory changes.
-Close and relaunch PowerShell into an administrative session.
-Navigate to a directory you want to see the ACLs in. You can go to C:\Windows, as you did in Task 4.
-Run the ***enum_acls.ps1*** script using the full file path and name such as the one in Task 4.
-Check the ***C:\Users\sysadmin\Documents*** for your new logs.
*You should see a directory with the current date (for example, ***20200908***) as the directory name. Your new transcribed PowerShell logs should be inside.




Windows 10 on ML-REFVM-244323 - Virtual Machine Connection
File Action Media View Help
Recycle Bin
Microsoft Edge
Administrator: Windows PowerShell
PS C:\Users\sysadmin\Documents\20220205> cat .\PowerShell_transcript.DESKTOP-SITPOTH.XqMo7Wa6.20220205075517.txt
****
Windows PowerShell transcript start
Start time: 20220205075524
Username: DESKTOP-SITPOTH\sysadmin
RunAs User: DESKTOP-SITPOTH\sysadmin
Configuration Name:
Machine: DESKTOP-SITPOTH (Microsoft Windows NT 10.0.19041.0)
Host Application: C:\Windows\system32\WindowsPowerShell\v1.0\PowerShell_ISE.exe
Process ID: 7524
PSVersion: 5.1.19041.1
PSEdition: Desktop
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.19041.1
BuildVersion: 10.0.19041.1
CLRVersion: 4.0.30319.42000
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
SerializationVersion: 1.1.0.1
****
****
Command start time: 20220205075524
****
PS>[Microsoft.Windows.PowerShell.Gui.Internal.HostTextWriter]::RegisterHost($host.ui)
****
Command start time: 20220205075525
****
PS>filter more { $_ }
****
Command start time: 20220205075525
****
PS>
function psEdit([Parameter(Mandatory=$true)]$filenames)
{
foreach ($filename in $filenames)
{
dir $filename | where {!$_.PSIsContainer} | %{
$psISE.CurrentPowerShellTab.Files.Add($_.FullName) > $null
}
}
}
****
Command start time: 20220205075525
****
PS>$OutputEncoding = [System.Console]::OutputEncoding
****
Command start time: 20220205075525
****
PS>ipmo ISE
****
Command start time: 20220205075526
****
PS>CommandInvocation(Set-Variable): "Set-Variable"
>> ParameterBinding(Set-Variable): name="Name"; value="profile"