Task 1- Create a GPO - Disable LLMNR

**Local Link Multicast Name Resolution (LLMNR)** is a vulnerability, so you will disable it on your Windows 10 machine (via the GC Computers OU).

A few notes about LLMNR:

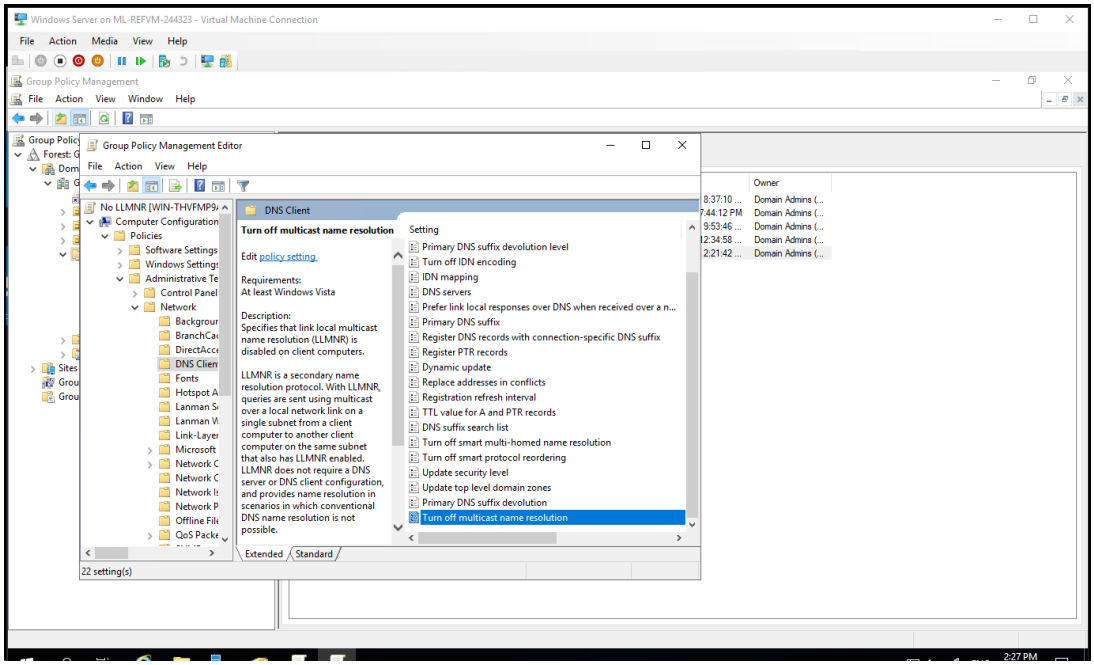
- LLMNR is a protocol used as a backup (not an alternative) for DNS in Windows.
- When Windows cannot find a local address (e.g., the location of a file server), it uses LLMNR to send out a broadcast across the network asking if any device knows the address.
- LLMNR's vulnerability is that it accepts any response as authentic, allowing attackers to poison or spoof LLMNR responses, forcing devices to authenticate to them.
- An LLMNR-enabled Windows machine may automatically trust responses from anyone in the network.

Turning off LLMNR for the GC Computers OU will prevent your Windows machine from trusting location responses from potential attackers.

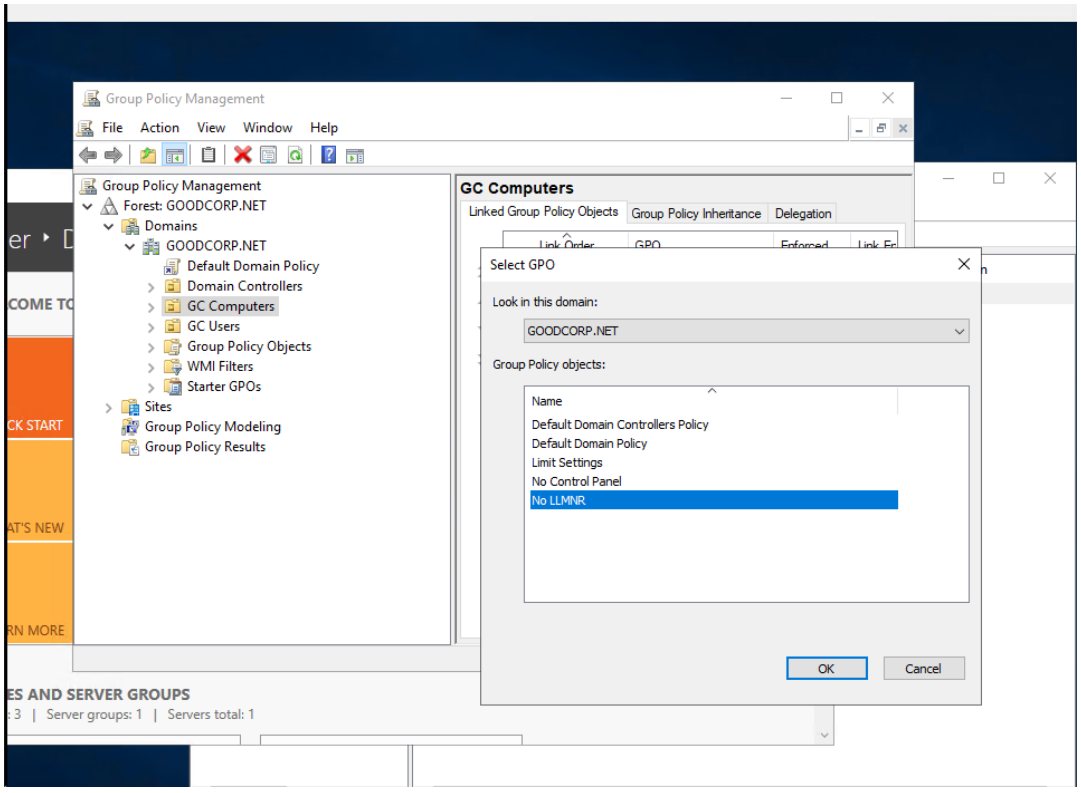
**Instructions**

Since this task deals with Active Directory Group Policy Objects, you'll work in your nested **Windows Server** machine.

- Create a Group Policy Object that prevents your domain-joined Windows machine from using LLMNR. To do so, complete the following steps:
- On the top-right of the Server Manager screen, open the Group Policy Management tool to create a new GPO.
- Right-click **Group Policy Objects** and select **New**.
- Name the Group Policy Object No LLMNR.
- Right-click the new **No LLMNR** GPO listing and select **Edit** to open the Group Policy Management Editor and find policies.
- In the Group Policy Management Editor, find the policy at the following path: Computer Configuration\Policies\Administrative Templates\Network\DNS Client.
    - Find the policy called Turn Off Multicast Name Resolution.
    - Enable this policy.




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

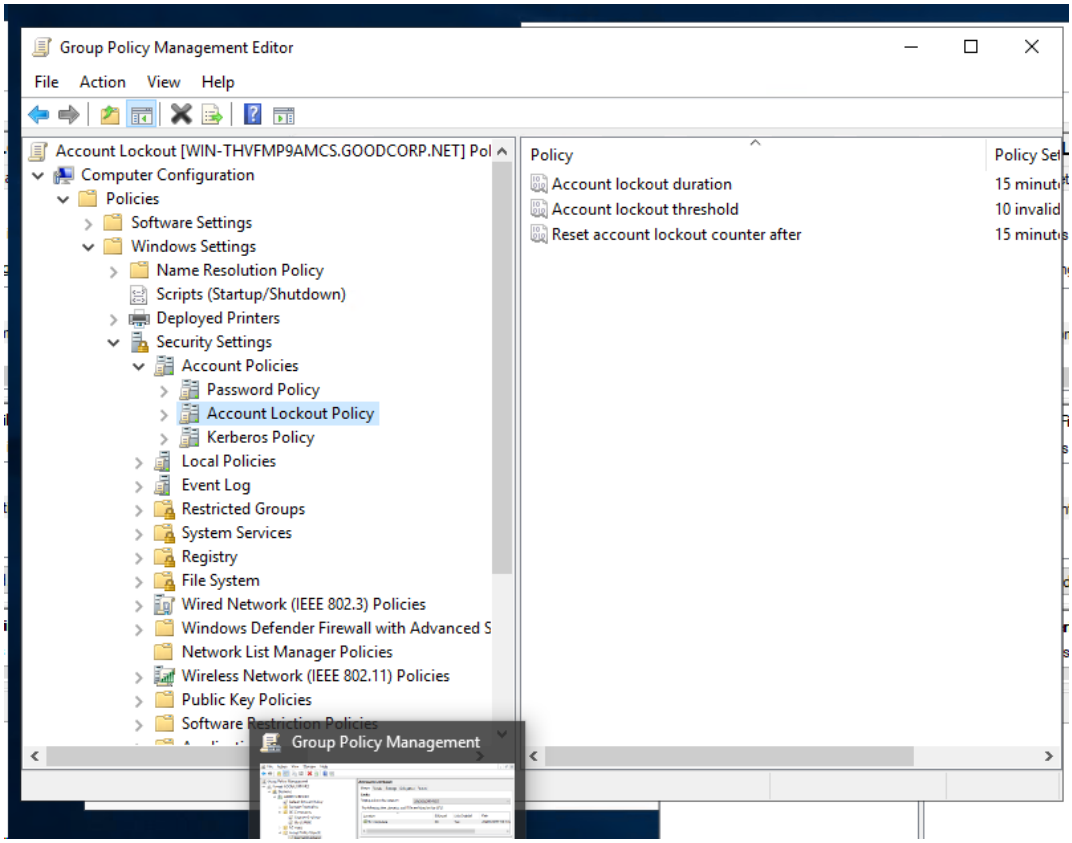


Task 2 - Create a GPO- Account Lockout

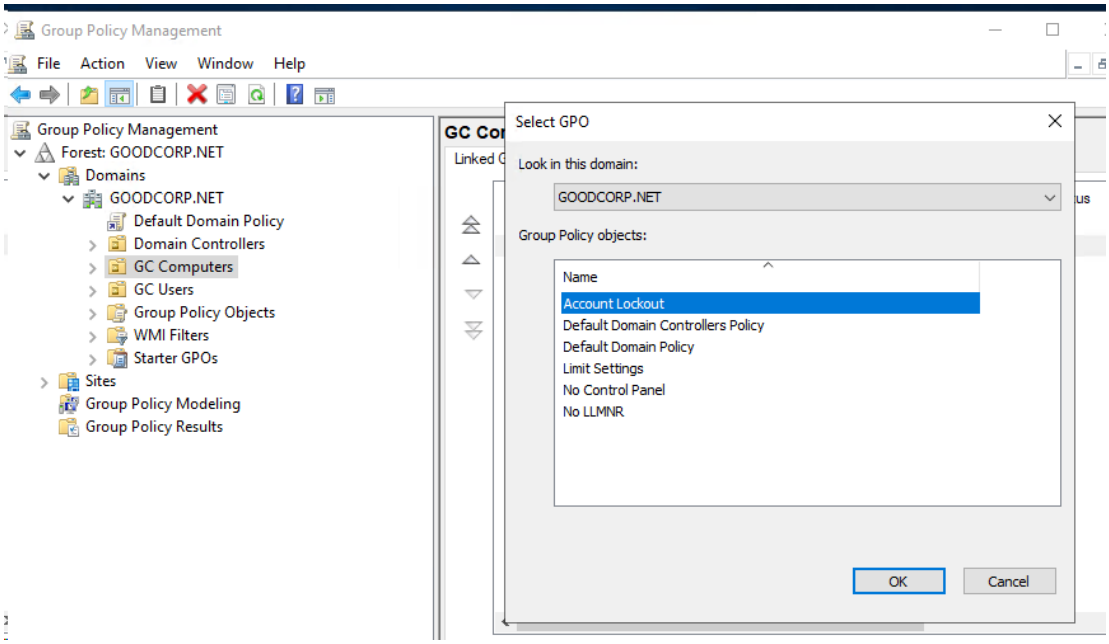
**Instructions**

Work within 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 machine.

- Name the Group Policy Object Account Lockout.
- You can use Microsoft's 10/15/15 recommendation if you'd like.
- When editing policies for this new GPO, keep in mind that you want *computer configuration* policies to apply to your GC Computers OU. Also, these policies involve Windows *security settings* and *accounts*.
- Don't forget to link the GPO to your GC Computers organizational unit.




Link the GPO to GC computers.

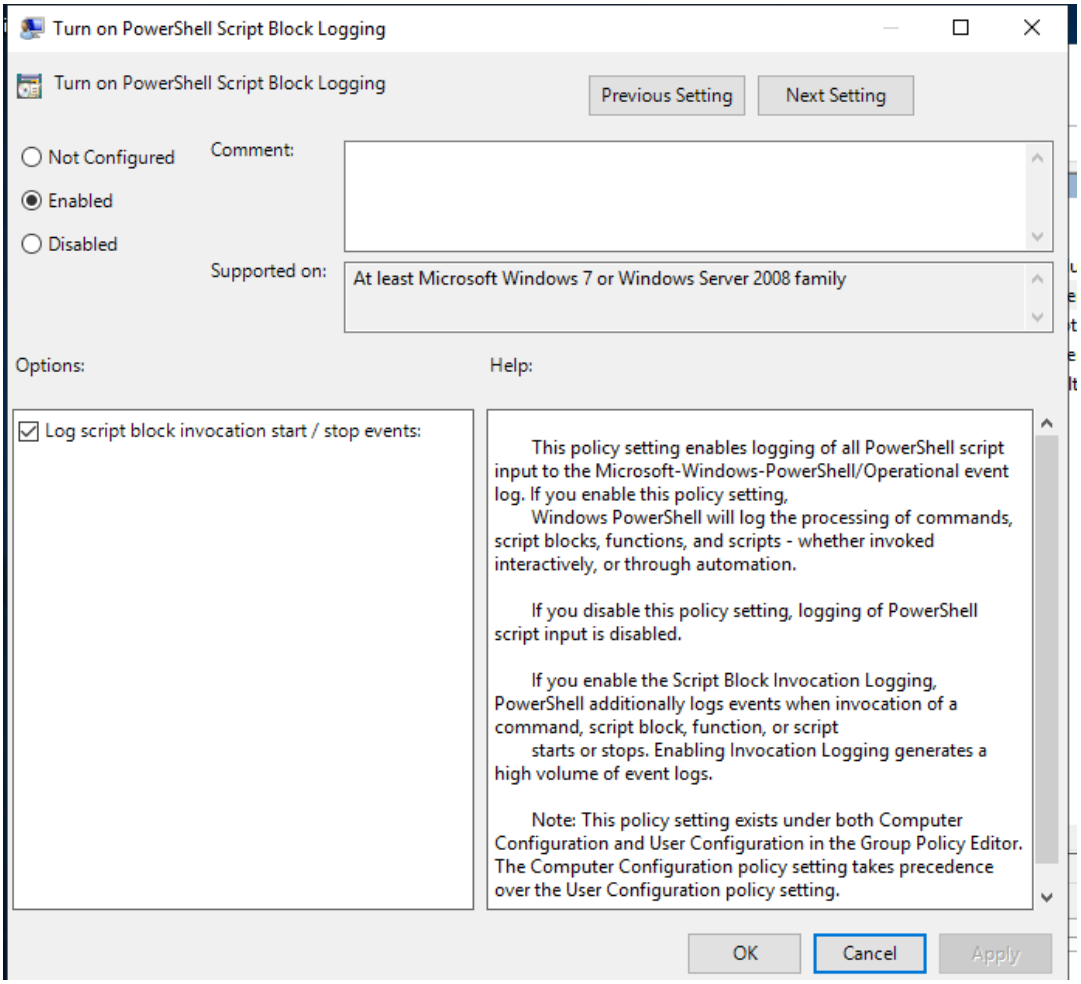



Task 3: Create a GPO - Enabling Verbose PowerShell Logging and Transcription

**Instructions**

For this task, work in your **Windows Server** machine. Create a Group Policy Object to enable PowerShell logging and transcription. This GPO will combine multiple policies into one, although they are all under the same policy collection.

1. Name the Group Policy Object PowerShell Logging.
    - Find the proper Windows PowerShell policy in Group Policy Management Editor.
2. Enable Turn on Module Logging and do the following:
    - Click **Show** next to **Module Names**.
    - Since you 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.

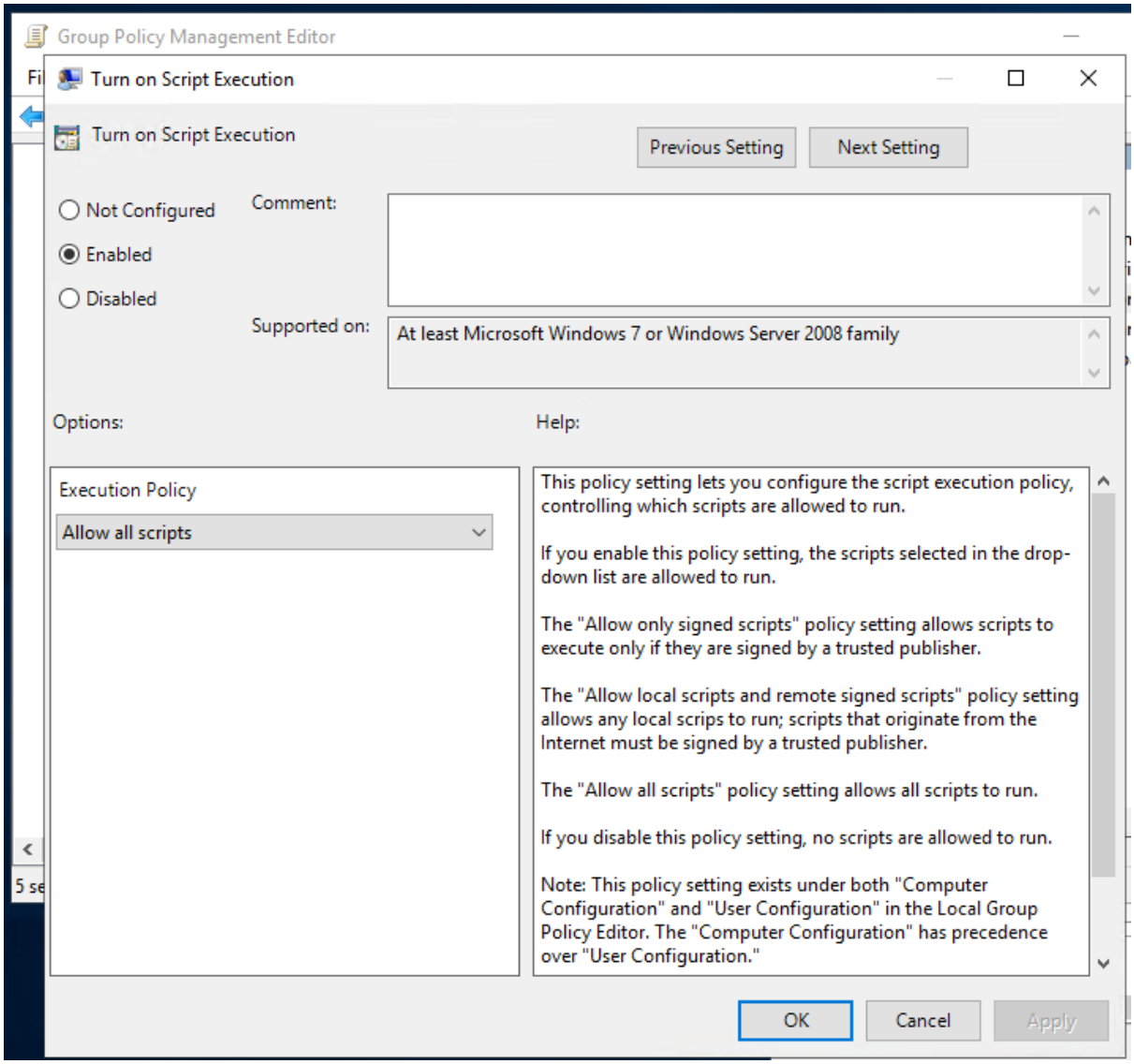



- This policy uses the following template to log what is executed in the script block:

```
$collection =
foreach ($item in $collection) {
  <Everything here will get logged by this policy>
}
```

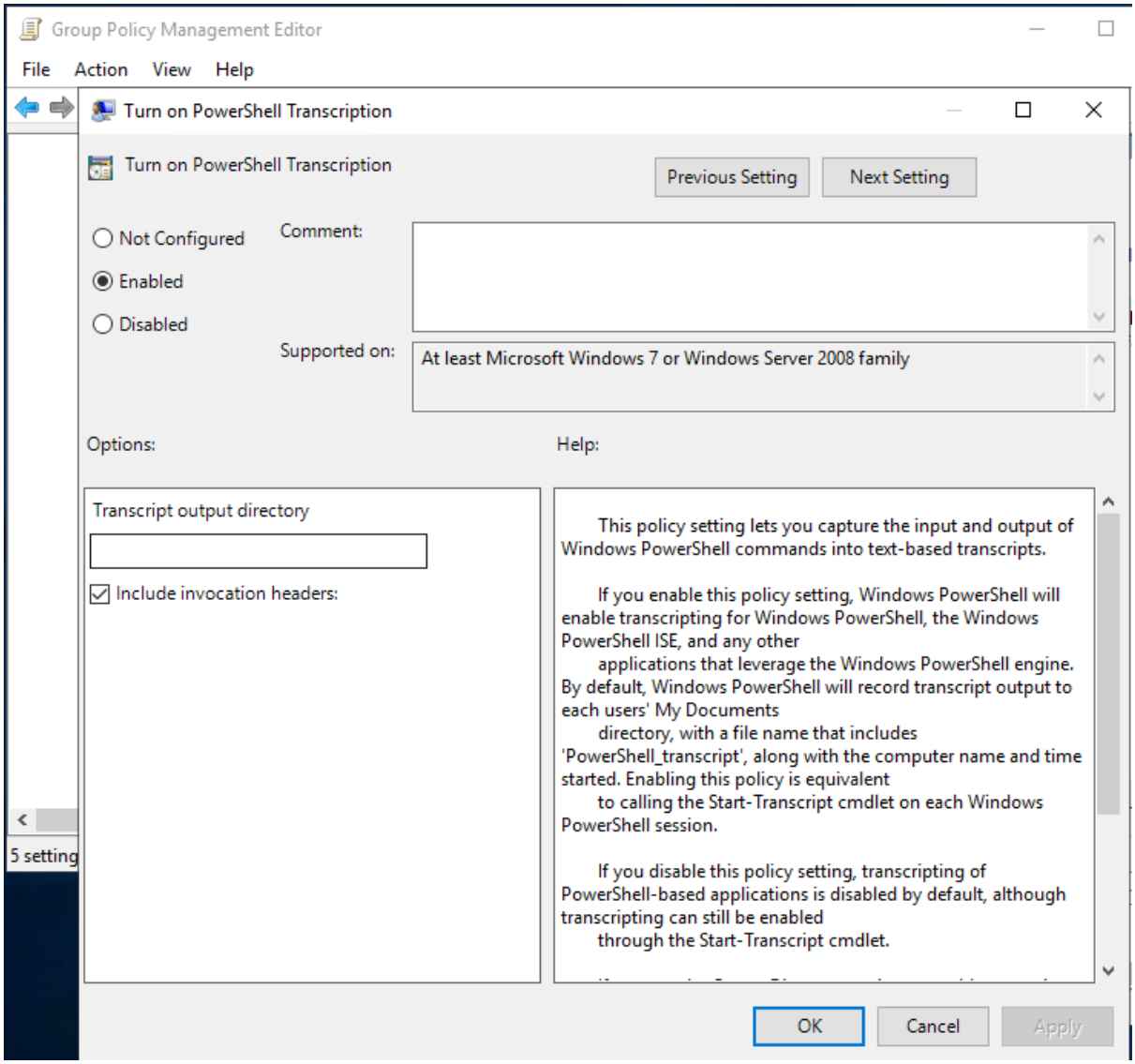
- Check the Log script block invocation start/stop events: setting.

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.

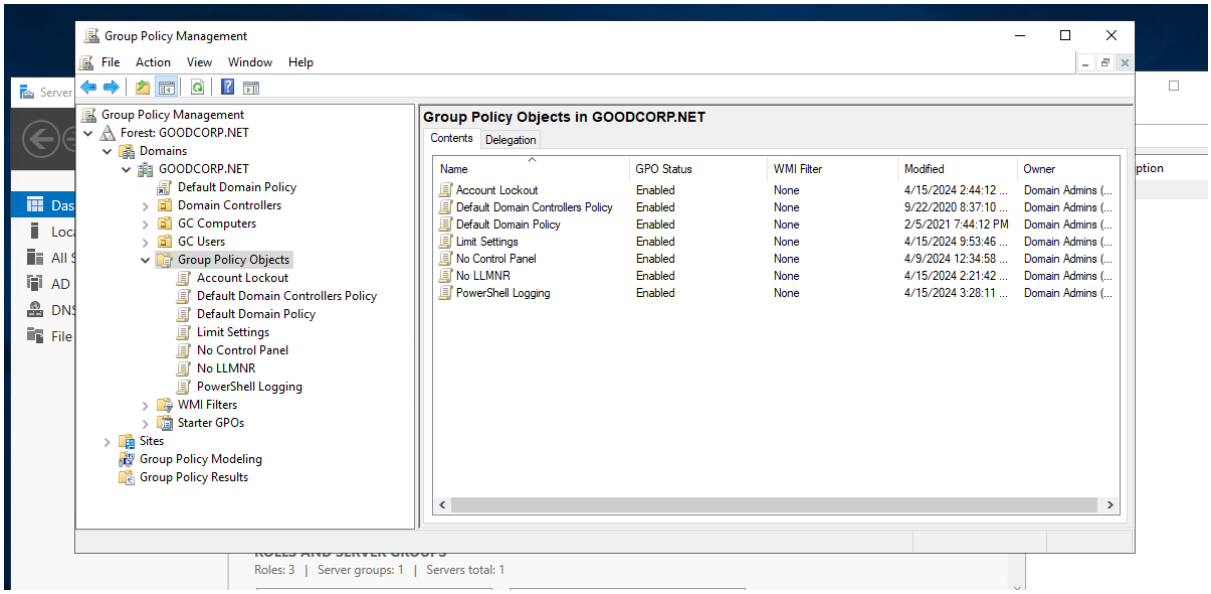
5. Enable the Turn on PowerShell Transcription policy and do the following:
    - Leave the **Transcript output directory** blank (this defaults to the user's ~\Documents directory).




**Note:** "Transcription" means that an exact copy of the 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.




The next time you log in to your Windows 10 machine, run gpupdate. Then launch a new PowerShell window and run a script. You'll see verbose PowerShell logs created in the Windows 10 machine directory for the user that ran the script: C:\Users<user>\Documents.

Speaking of scripts, your next task is to create a script.

### Task 4: Create a Script—Enumerate Access Control Lists

Before we create a script, let's review Access Control Lists.

- In Windows, access to files and directories are managed by Access Control Lists (ACLs). These identify which entities (known as security principals), such as users and groups, can access which resources. ACLs use security identifiers to manage which principals can access which resources.
- While you don't need to know the specific components within ACLs for this task, you do need to know how to use the Get-Acl PowerShell cmdlet to retrieve them. View Get-Acl documentation here.

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.

**Instructions**

For this task, you'll work in your nested **Windows 10** machine with the following credentials: sysadmin | cybersecurity.

Create a PowerShell script that will enumerate the Access Control List of each file or subdirectory within the current working directory. To do so, complete the following steps:

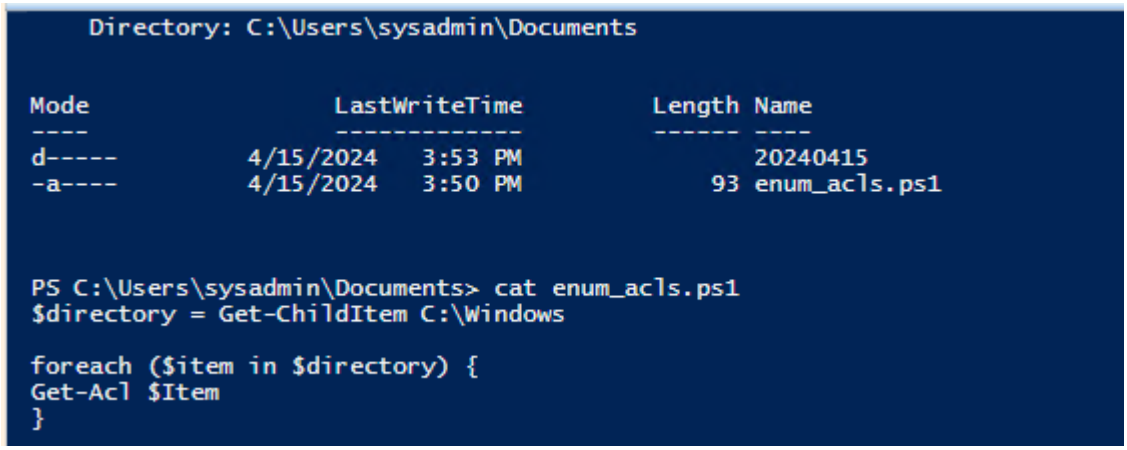
1. Create a foreach loop. You can use the following template:

foreach ($item in $directory) {

<Script block>

}

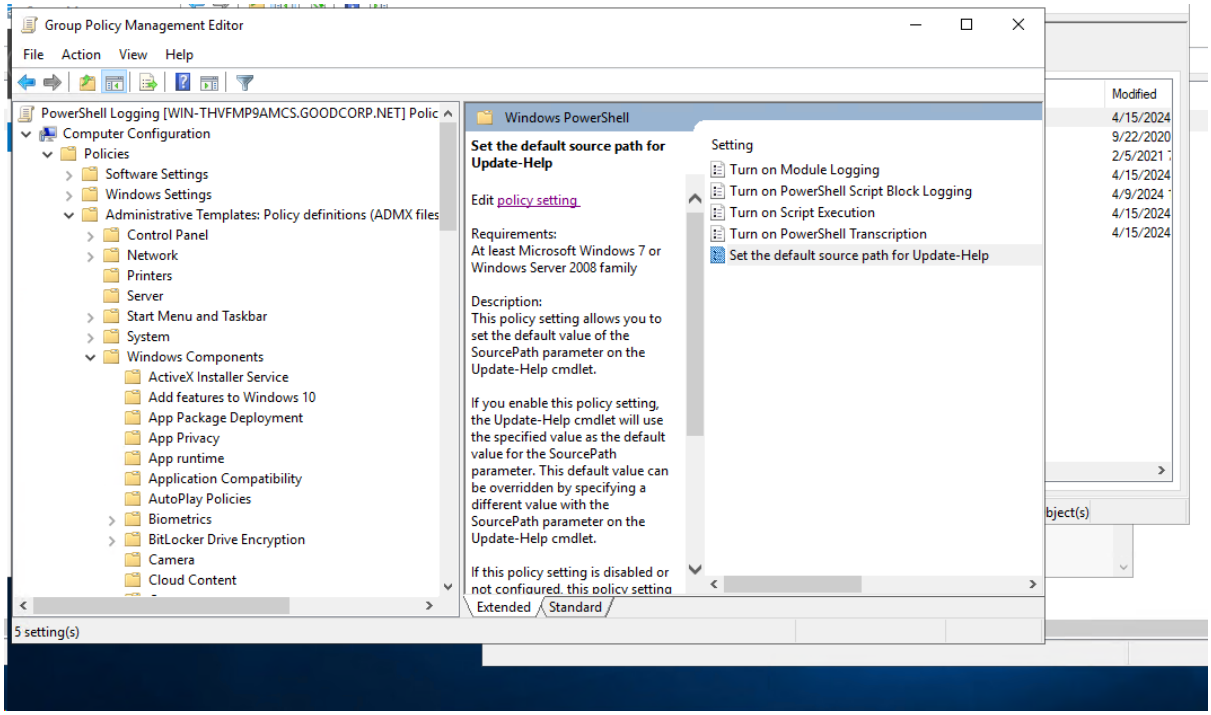
2. Above the foreach condition, set a variable, $directory, to the contents of the current directory.
3. Replace the script block placeholder with the command to enumerate the ACL of a file, using the $item variable in place of the file name.
    - You'll need to use the following cmdlets:
        - 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.



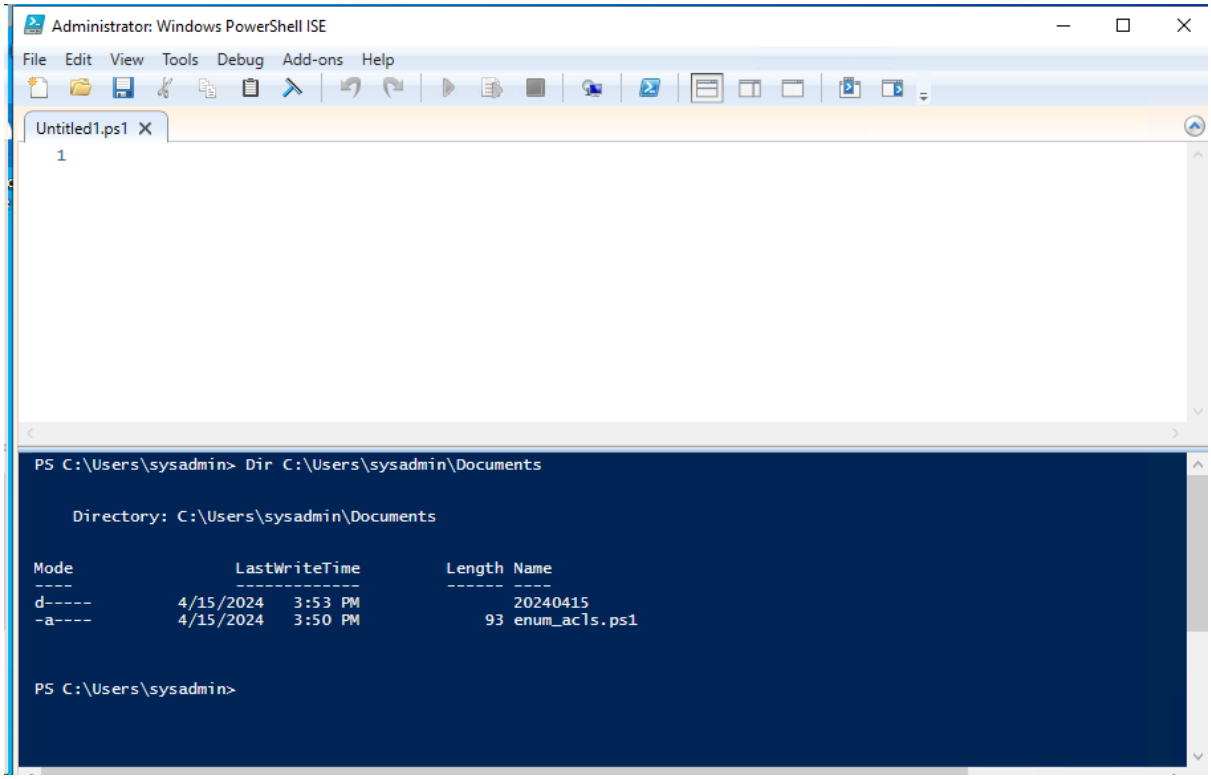

5. Test this script by moving to any directory (cd C:\Windows), and running C:\Users\sysadmin\Documents\enum_acls.ps1 (enter the full path and file name).
    - You should see the ACL output of each file or subdirectory where you ran the script from.

*A different pathway but same outcome:*

*Computer configuration --> Policies --> Administrative templates --> Windows components --> Windows PowerShell*




After the script running:



**Optional Additional Challenge 5: Verify Your PowerShell Logging GPO**

For this task, test and verify that your PowerShell logging GPO is working properly.

**Instructions**

- Ensure that you're logged in to 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 that 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.