Implementing PowerShell Security in Windows

Scenario

Windows PowerShell is a very powerful administrative tool. As a security professional, you will need to control what PowerShell scripts run on your Windows servers. There are individual PowerShell cmdlets, and there are settings configured through Group Policy. Group Policy allows Active Directory administrators to centralize configuration control. You will create a Group Policy Object (GPO) to manage PowerShell logging. You will then create a script and explore the available execution policies. You will digitally sign your script to provide an additional layer of security. Next, you'll edit the GPO to centrally manage the execution policy. To match the Contoso company security policy, you will disable PS remoting. Finally, you will review the PowerShell log files that you enabled at the start of the activity.

Objectives

This activity is designed to test your understanding of and ability to apply content examples in the following CompTIA Security+ objective:

- 3.2 Given a scenario, implement host or application security features.

Lab

- DC1 VM

Task 1

Configure a GPO for PS Logging

Configure a Group Policy Object that enables logging and link it to the contoso.local domain.

1. Send **CTRL+ALT+DEL**, and then sign in to the **DC1** VM as **CONTOSO\Administrator** using **Pa$$w0rd** as the password.

2. From **Server Manager**, select **Tools > Group Policy Management**.

3. Right-click the **contoso.local** domain object and select **Create a new GPO in this domain, and Link it here**.

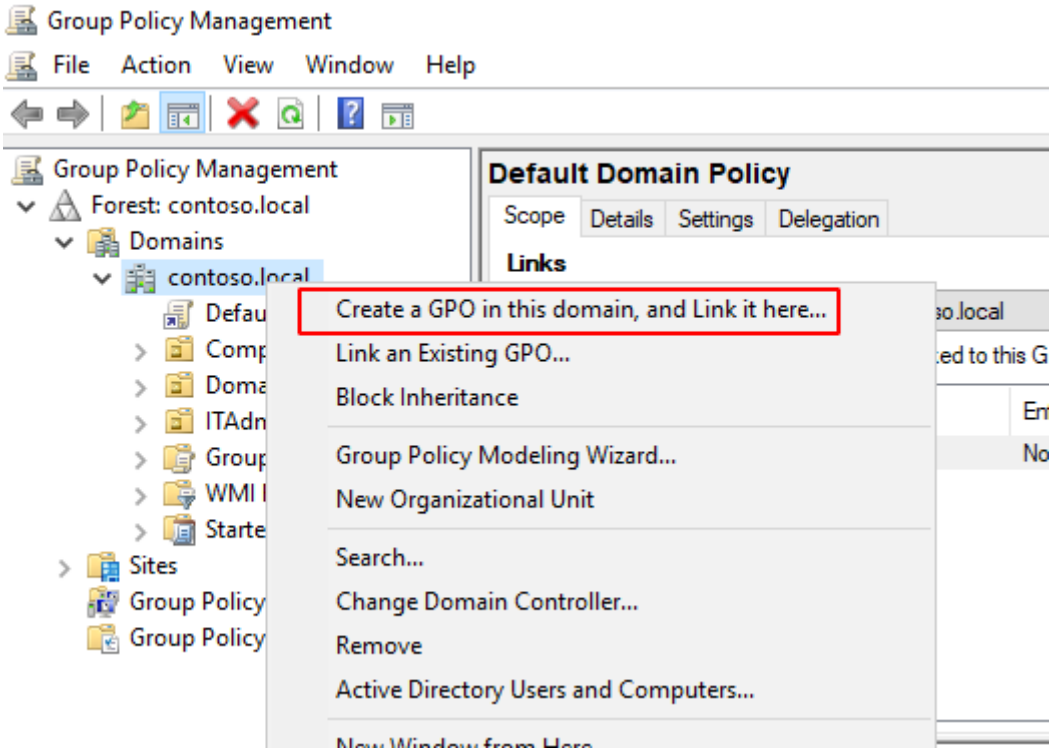

Figure 1.1 – Creating a GPO in domain contoso.local.

4. Name the new GPO **PowerShell Security GPO**, and then select **OK**.

5. Right-click the **PowerShell Security GPO**, and then select **Edit**.

6. In the GPO editor, select **Computer Configuration > Policies > Administrative Templates > Windows Components > Windows PowerShell**.

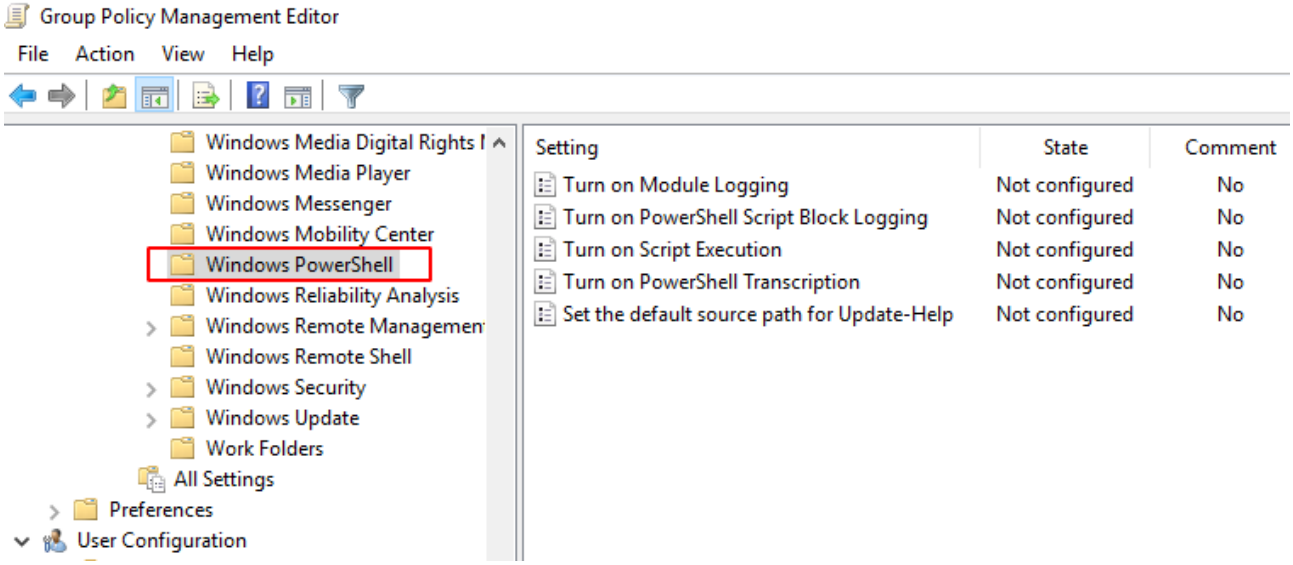

Figure 1.2 – Windows PowerShell Policies.

7. Right-click the **Turn on PowerShell Script Block Logging** setting, and then select **Edit**.

8. Select the **Enabled** radio button, and then select **OK**.

9. Close the **Group Policy Management Editor**.

10. Select the **Group Policy Objects** folder. Right-click the **PowerShell Security GPO**, and select **Save Report**.

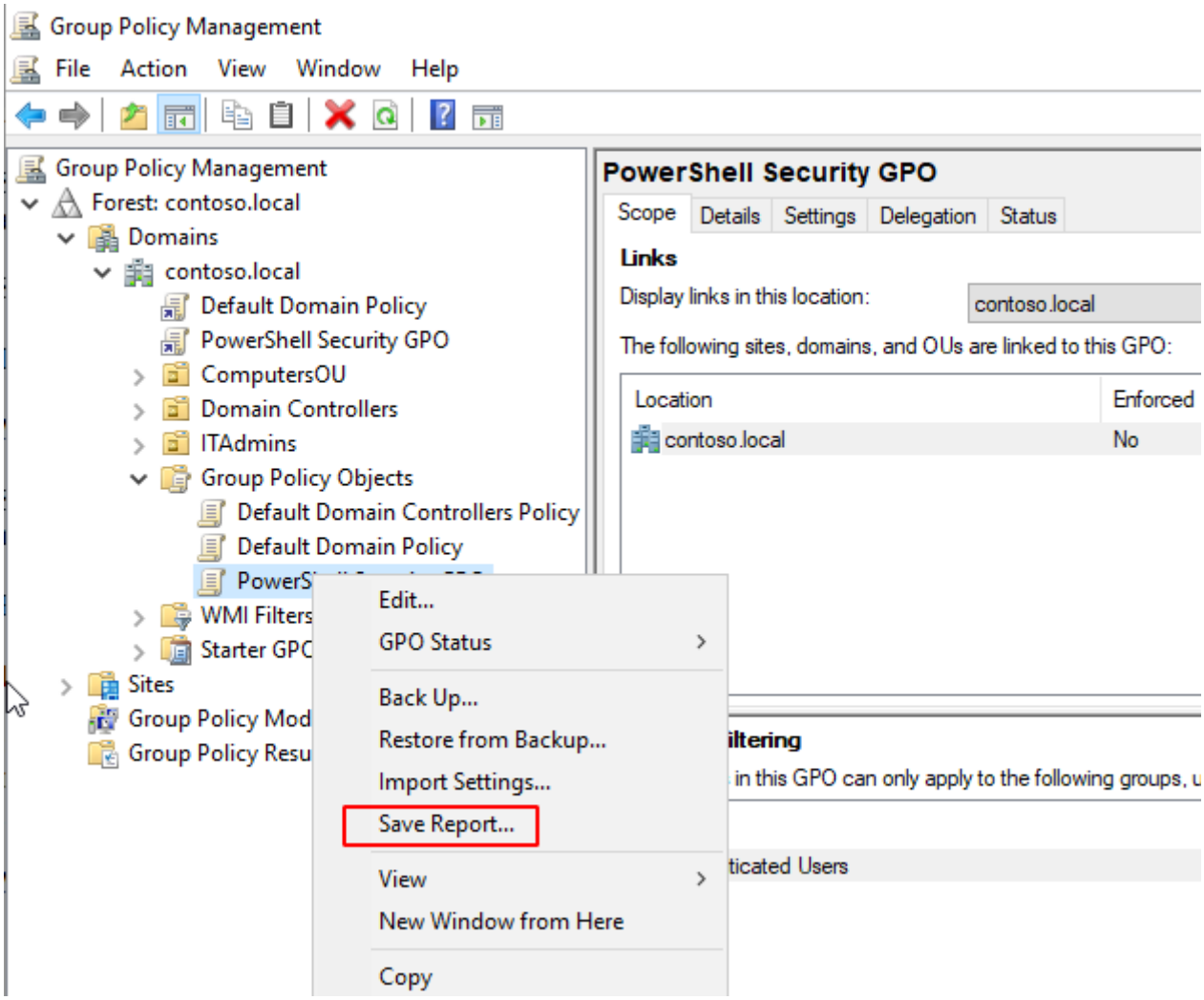

Figure 1.3 – Saving a report of the PowerShell Security GPO.

11. Select the **Desktop** folder, and then select the **Save** button.

12. **Double-click** the file to check the report.

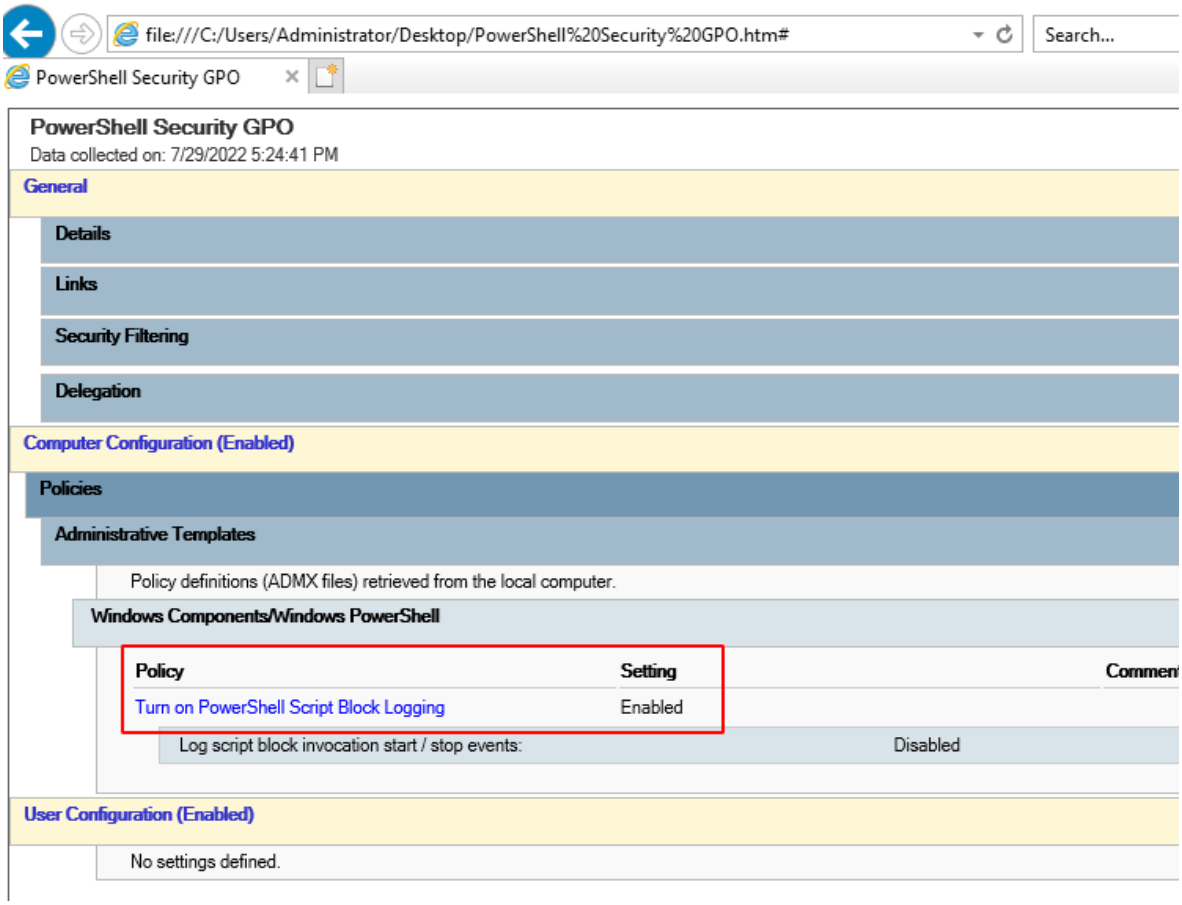

Figure 1.4 – Analyzing the exported GPO report.

13. Minimize the **Group Policy Management Console**.

14. Select the **Start** menu, right-click **Windows PowerShell**, and then select **Run as Administrator**.

TIP: Select **Yes** to confirm the UAC prompt if displayed on the screen.

15. Run the following cmdlet:

```
gpupdate /force
```

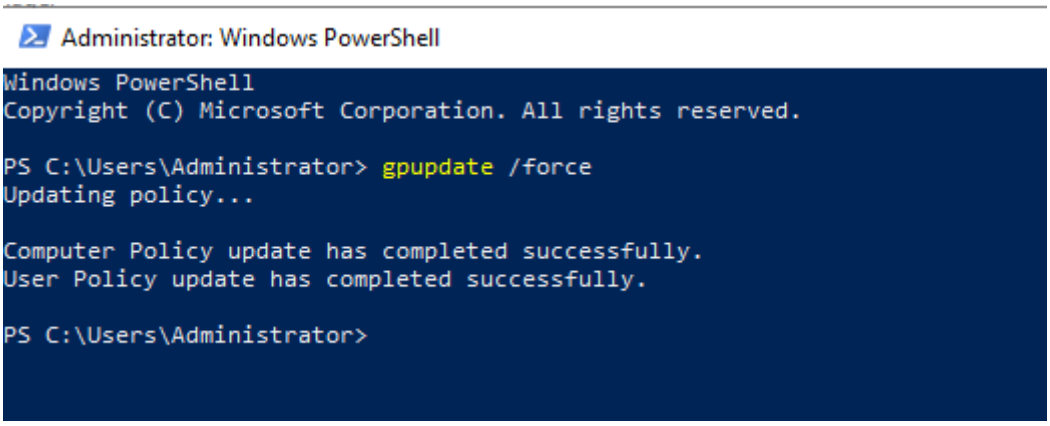

Figure 1.5 – Executing gpupdate cmdlet.

TIP: You might need to repeat the cmdlet. Use the **UP ARROW** key to repeat the most recent command or PowerShell cmdlet. For example, type **gpupdate /force** the first time you need it, and then select the **UP ARROW** key to repeat it.

16. Minimize the **PowerShell** console.

Task 2

Manage PowerShell execution control

You will configure PowerShell execution control on the DC1 VM. You would not be likely to create and test scripts on a domain controller. We are just using this VM to simplify the lab environment requirements.

Setting	Description
Unrestricted	No requirements; all scripts allowed
RemoteSigned	Local scripts allowed; remote scripts must be signed
AllSigned	Local and remote scripts must be signed
Restricted	No scripts allowed

Table 2.1 - PowerShell execution policy reference table.

1. In **Server Manager**, select **Tools > Windows PowerShell ISE**.

2. In PowerShell ISE, add the following script to the **Untitled.ps1** script pane (in the text editor, not at the command prompt at the bottom).

```
# Display the following text
Write-Host “Hello, $env:Username!”

# Display the date
$today = Get-Date
Write-Host “Today is $today”
```

3. From the PowerShell ISE menu bar, select the green triangle **Run Script (f5)** button to execute the script.

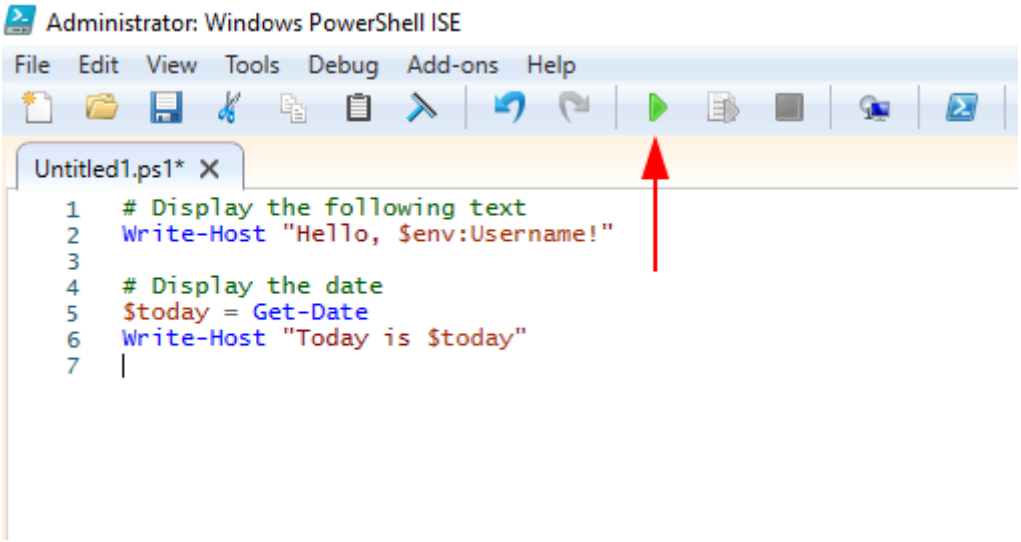

Figure 2.1 – Windows PowerShell ISE

Note that the script runs and displays the script contents and the formatted script data.

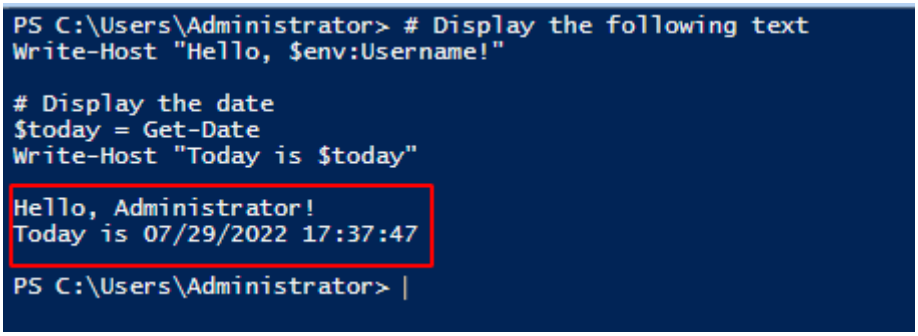

Figure 2.2 – Script execution on the bottom pane.

4. From the PowerShell ISE menu bar, save the script to the Desktop as a file named **PSHello.ps1**.

5. **Close** the PowerShell ISE.

6. Maximize the PowerShell console, or from the **Server Manager**, select **Tools > Windows PowerShell**.

7. Change to the Desktop folder by entering:

```
cd C:\Users\Administrator\Desktop
```

8. To take advantage of PowerShell's tab completion feature, call the script by typing **PSHe** , and then pressing the **TAB** key, and then press **ENTER**.

9. Run the following PowerShell cmdlet to display the current execution policy on the system:

```
Get-Executionpolicy
```

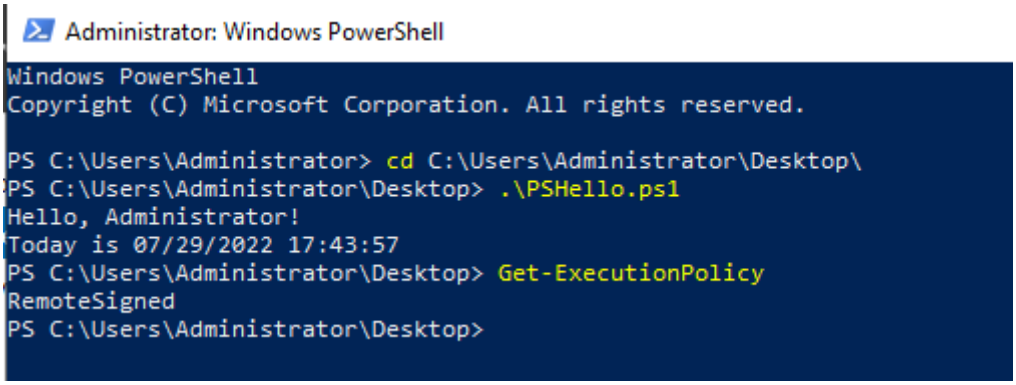

Figure 2.3 – Executing PowerShell scripts and viewing the Execution Policy.

10. Review the following snippet of the organization's written security policy as it pertains to scripts and security:

** Scripts created in-house must be digitally signed. Such scripts may be signed with a self-signed certificate. Scripts downloaded from the internet must be digitally signed by a trusted source. The PowerShell execution policy will be set as* ***AllSigned****.*

11. Configure the PowerShell Execution Policy to match the requirements.

```
Set-ExecutionPolicy -ExecutionPolicy AllSigned -Scope LocalMachine
```

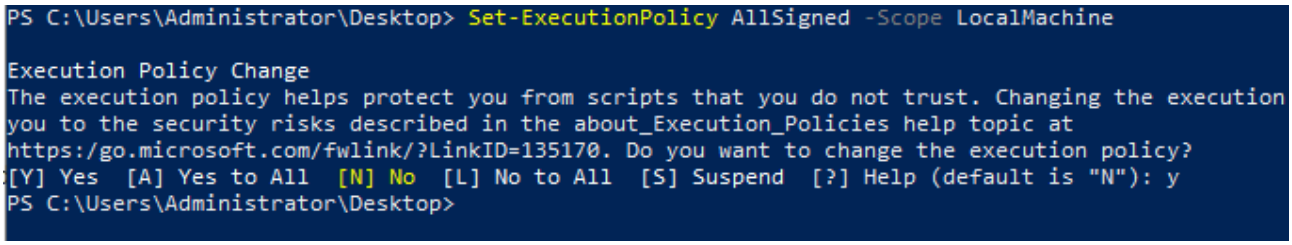

Figure 2.4 – Changing the Execution Policy

12. When prompted, enter y to respond "**yes**" to the confirmation message.

NOTE: This type of configuration is usually enforced through Group Policy. You will configure the Group Policy settings below.

13. Run the **PSHello.ps1** test script again from PowerShell.

TIP: Use the **UP ARROW** key to cycle through recent commands. For example, you could easily return the Get-ExecutionPolicy cmdlet by cycling upward through the PowerShell history.

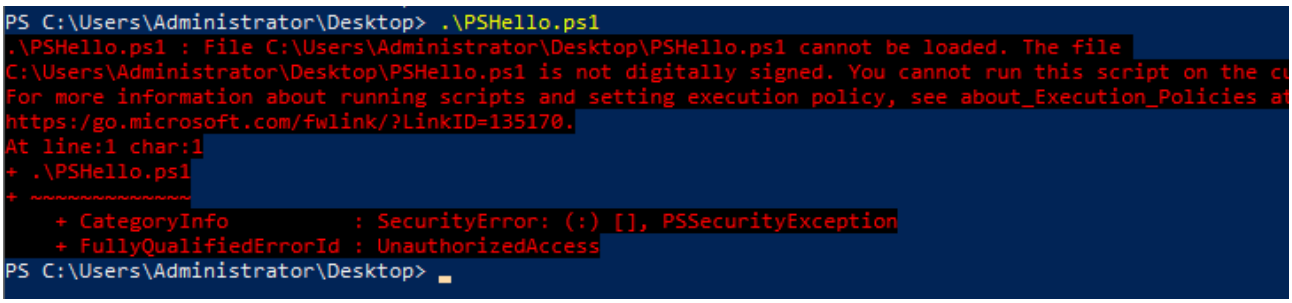

Figure 2.5 – Executing saved PowerShell script.

The script should fail this time, because it is not digitally signed. PowerShell errors are displayed in red.

14. Run the **Get-ExecutionPolicy** cmdlet again to confirm the policy is set to **AllSigned**.

15. Run the **Get-AuthenticodeSignature -FilePath C:\Users\Administrator\Desktop\Pshello.ps1** cmdlet.

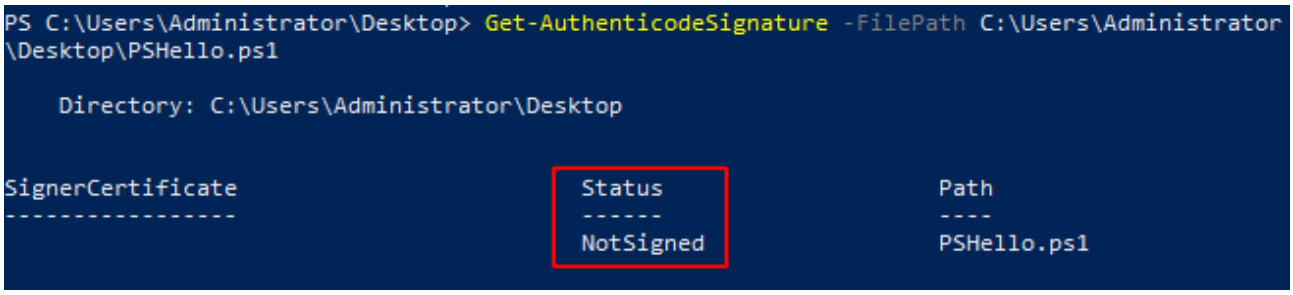

Figure 2.6 – The status of the PSHello.ps1.

Task 3

Create a code signing certificate

According to the company policy, you need to digitally sign your PowerShell scripts. You will generate a code signing certificate and link it to the test script.

1. In the PowerShell window, run the following cmdlet to generate a new self-signed certificate.

```
New-SelfSignedCertificate -DnsName administrator@contoso.local -
CertStoreLocation Cert:\CurrentUser\My\ -Type CodeSigning
```

NOTE: The certificate may take up to one minute to be generated.

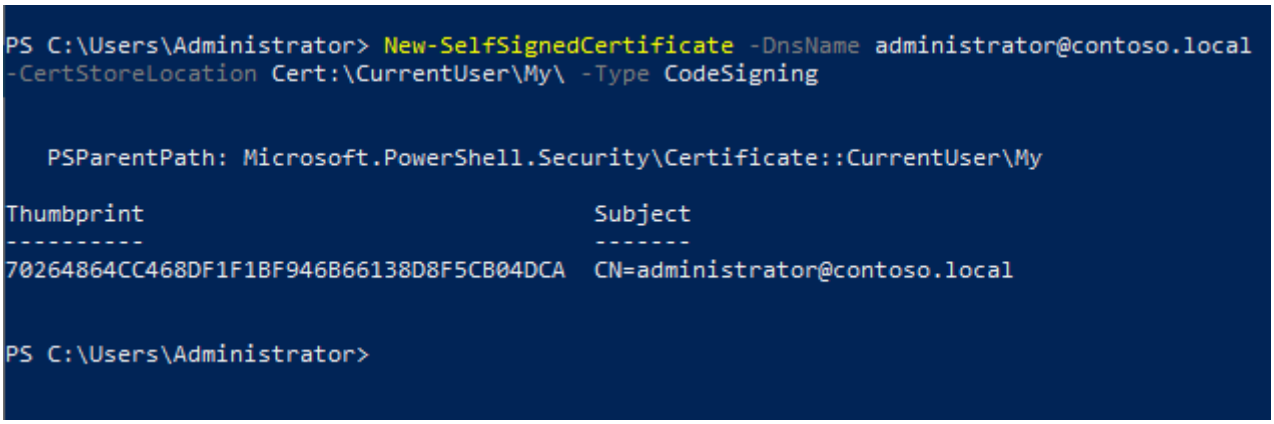

Figure 3.1 – PowerShell New-SelfSignedCertificate cmdlet.

2. Open the Certificates MMC by running the **certmgr.msc** command in PowerShell.

3. Expand the **Personal** node, and then select the **Certificates** node.

Two certificates are displayed. Including the new Code Signing certificate.

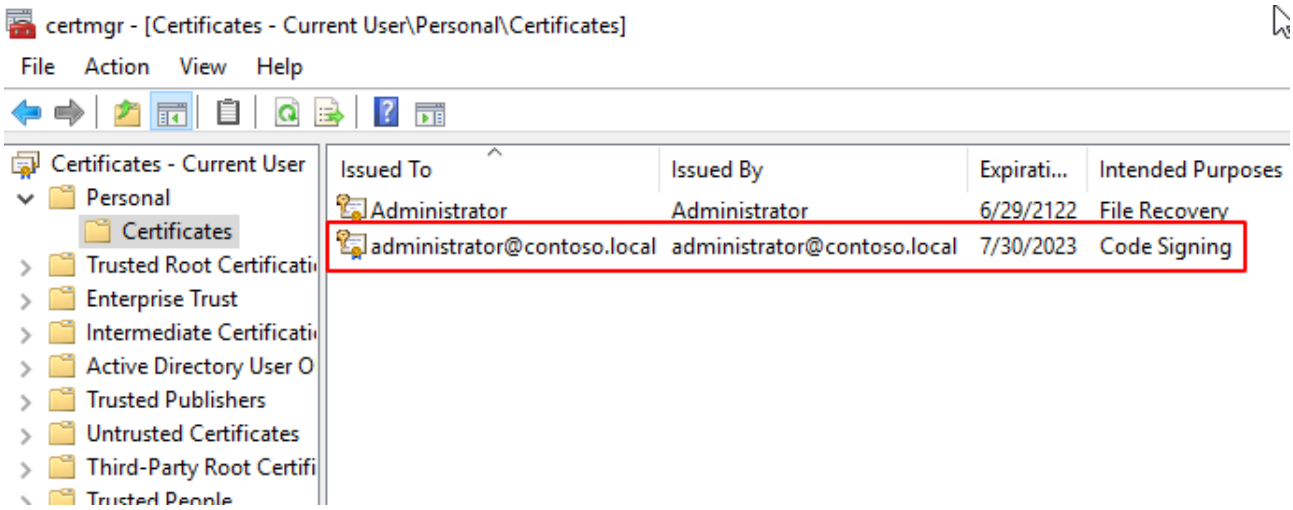

Figure 3.2 – The certmgr.msc console displaying the code signing certificate.

4. Double-click the new **Code Signing** certificate, select the **Details** tab, and then select **Copy to File**.

5. Complete the **Certificate Export Wizard** by making the following selections:

- Select **No, do not export the private key**
- Select **DER encode binary X.509 (.CER)** format
- Browse to the **Desktop** and **save** the file with the name **code-cert**
- Click **Next** and **Finish.**

6. A message indicates the export was successful. Select **OK**.

7. **Close** the certificate properties dialog box.

8. In the **certmgr** console, right-click **Trusted Root Certification Authorities**, and then select **All Tasks > Import**.

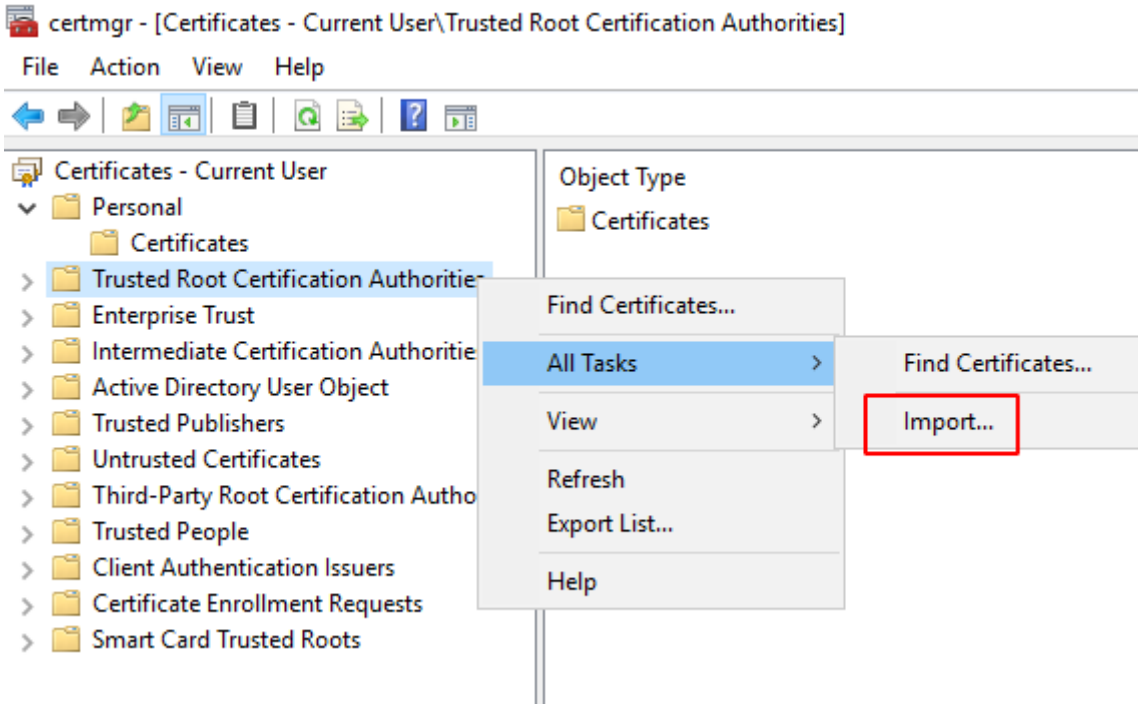

Figure 3.3 – Importing a certificate to the Trusted Root Certification Authorities folder.

9. In the **Certificate Import Wizard**, specify the **code-cert.cer** file from the **Desktop**, accept all other defaults, and then select **Finish**.

10. Select **Yes** and then select **OK** when prompted to install the certificate.

TIP: The import process may take approximately 30 seconds.

11. In the **certmgr** console, right-click **Trusted Publishers**, and then repeat the process above to import the **code-cert.cer** certificate.

12. **Confirm** the prompts to **install** the certificate.

13. Close the **certmgr** console.

14. From the **PowerShell** console, run the following cmdlet to digitally sign the Psversion.ps1 script:

```
Set-AuthenticodeSignature -FilePath C:\Users\Administrator\Desktop\
PSHello.ps1 -Certificate (Get-ChildItem -Path Cert:\CurrentUser\My\ -
CodeSigningCert)
```

15. Run the **.\PSHello.ps1** test script again.

You should be successful because the script is now digitally signed.

```
PS C:\Users\Administrator\Desktop> Set-AuthenticodeSignature -FilePath C:\Users\Administra
tor\Desktop\PSHello.ps1 -Certificate (Get-ChildItem -Path Cert:\CurrentUser\My\ -CodeSigni
ngCert)


    Directory: C:\Users\Administrator\Desktop


SignerCertificate                         Status                 Path
-----------------                         ------                 ----
70264864CC468DF1F1BF946B66138D8F5CB04DCA  Valid                  PSHello.ps1


PS C:\Users\Administrator\Desktop> .\PSHello.ps1
Hello, Administrator!
Today is 07/30/2022 09:26:53
PS C:\Users\Administrator\Desktop>
```

Figure 3.4 – Signing the script and executing it.

16. Run the following cmdlet again to confirm the status of the script.

```
Get-AuthenticodeSignature -FilePath C:\Users\Administrator\Desktop\
PSversion.ps1
```

```
PS C:\Users\Administrator\Desktop> Get-AuthenticodeSignature -FilePath C:\Users\Administra
tor\Desktop\PSHello.ps1


    Directory: C:\Users\Administrator\Desktop


SignerCertificate                         Status                 Path
-----------------                         ------                 ----
70264864CC468DF1F1BF946B66138D8F5CB04DCA  Valid                  PSHello.ps1
```

Figure 3.5 – Confirm the status of the PSHello.ps1 script.

17. Close the **PowerShell** window.

Task 4

Configure a GPO that enforces the execution policy

Edit an existing Group Policy Object (GPO) to set the execution policy as AllSigned, and link it to the contoso.local domain.

1. In the **Group Policy Management c**onsole, expand the **contoso.local** domain object.

2. Right-click the **PowerShell Security GPO** (GPO), and then select **Edit**.

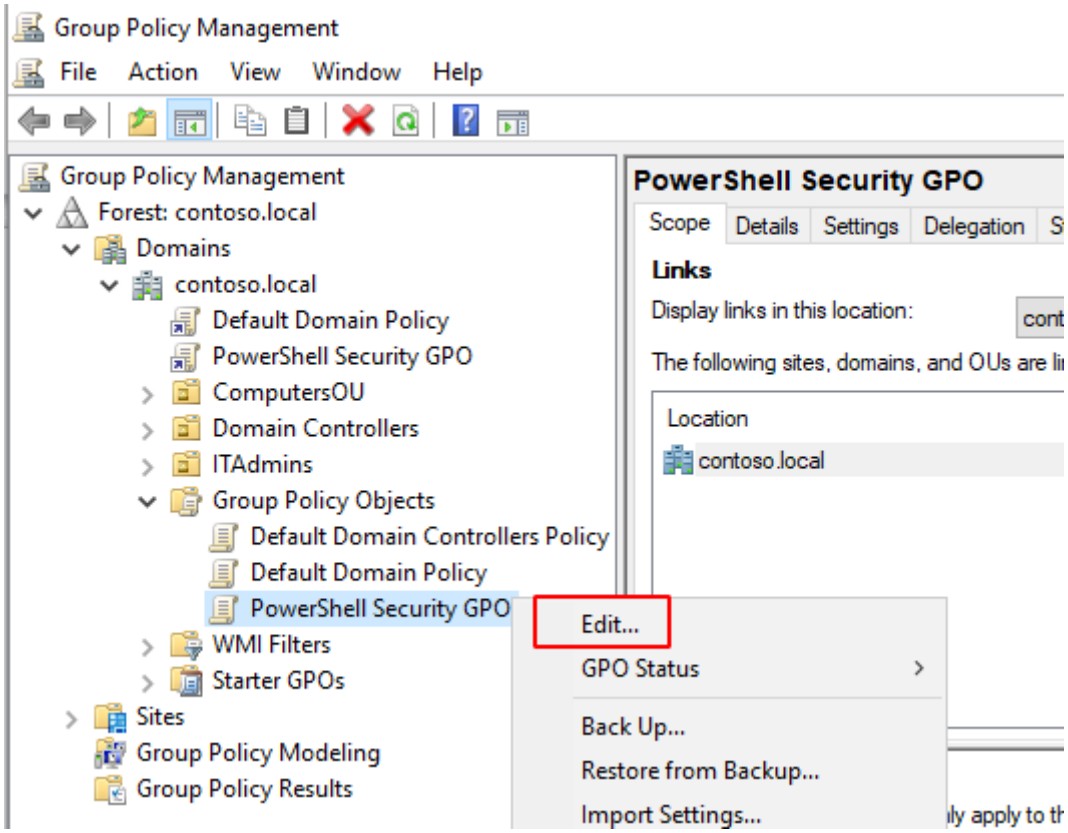

Figure 4.1 – Group Policy Management console.

3. In the GPO editor, select **Computer Configuration > Policies > Administrative Templates > Windows Components > Windows PowerShell**.

4. Right-click **Turn on Script Execution**, and then select **Edit**.

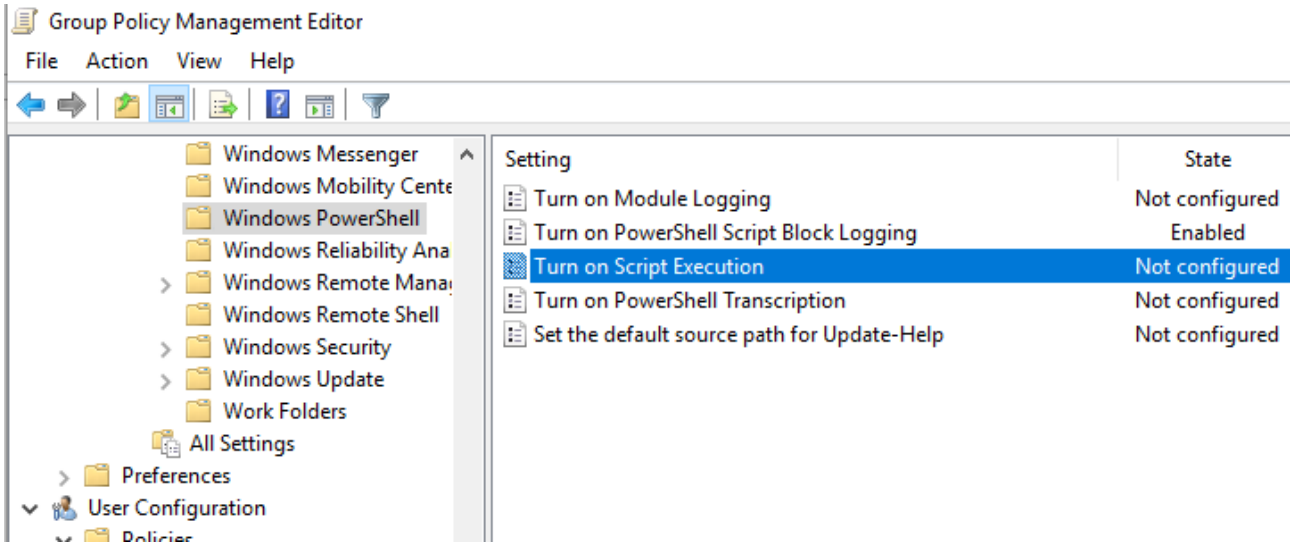

Figure 4.2 – Editing the Turn on Screen Execution policy.

5. Select the Enabled radio button.

6. In the Execution Policy drop-down menu, select **Allow only signed scripts**, and then select **OK**.

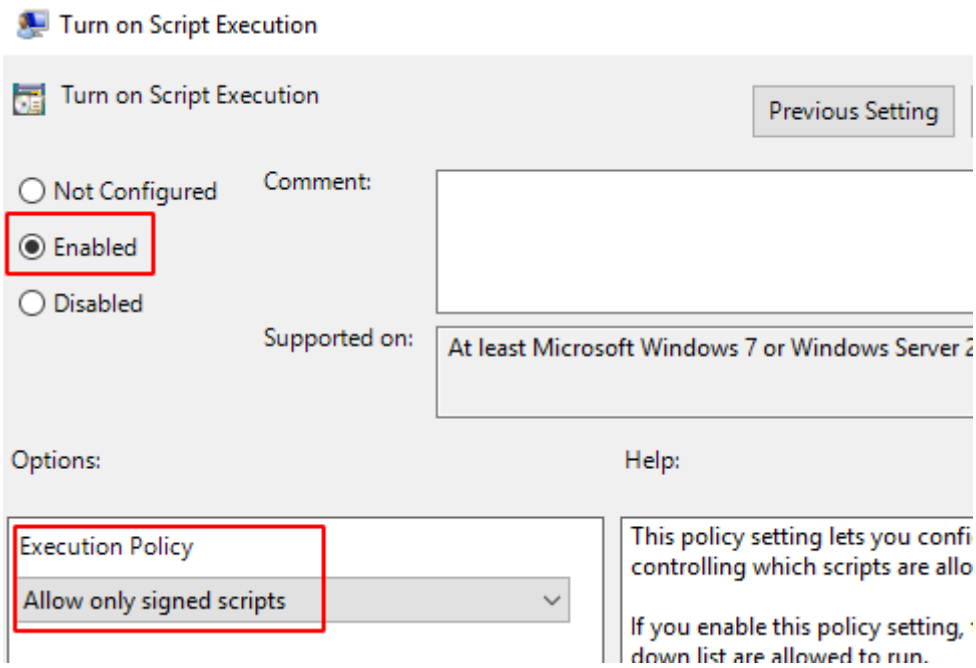

Figure 4.3 – Enabling and configuring the Execution Policy.

7. Close the **Group Policy Management Editor**.

8. Close the **Group Policy Management Console**.

Task 5

Disable Remote PowerShell

The Contoso security policy states that PowerShell cmdlets must only be run locally on Domain Controllers. You must disable PowerShell Remoting on DC1 to prevent users from establishing remote PowerShell sessions. Such sessions would permit the remote execution of cmdlets such as Restart-Computer or Get-Service on the DC.

1. Review the following snippet of the organization's written security policy as it pertains to scripts and security:

** The Remote PowerShell functionality will be disabled on all Domain Controllers.*

2. If necessary, launch **Windows PowerShell** from the **Tools** menu in **Server Manager**.

3. Configure the PowerShell cmdlet that disables PowerShell Remoting:

```
Disable-PSRemoting -Force
```

```
PS C:\Users\Administrator> Disable-PSRemoting -Force
WARNING: Disabling the session configurations does not undo all the changes made by the Enable-PSRemoting or
Enable-PSSessionConfiguration cmdlet. You might have to manually undo the changes by following these steps:
    1. Stop and disable the WinRM service.
    2. Delete the listener that accepts requests on any IP address.
    3. Disable the firewall exceptions for WS-Management communications.
    4. Restore the value of the LocalAccountTokenFilterPolicy to 0, which restricts remote access to members of the
Administrators group on the computer.
PS C:\Users\Administrator>
```

Figure 5.1 – Disabling PSRemoting.

4. Verify the status of PowerShell Remoting:

```
Get-Pssessionconfiguration | Format-Table -Property Name, Permission
```

```
PS C:\Users\Administrator> Get-PSSessionConfiguration | Format-Table -Property Name, Permission

Name                                Permission
----                                ----------
microsoft.powershell                NT AUTHORITY\NETWORK AccessDenied, NT AUTHORITY\INTERACTIVE
microsoft.powershell.workflow       NT AUTHORITY\NETWORK AccessDenied, BUILTIN\Administrators Ac
microsoft.powershell32              NT AUTHORITY\NETWORK AccessDenied, NT AUTHORITY\INTERACTIVE
microsoft.windows.servermanagerworkflows NT AUTHORITY\NETWORK AccessDenied, NT AUTHORITY\INTERACTIVE


PS C:\Users\Administrator>
```

Figure 5.2 – The status of the NT AUTHORITY\NETWORK identity is set to AccessDenied.

5. Close the **PowerShell** console.

Task 6

Review PowerShell logging

Log files are critical component of server security. You will display and filter PowerShell log files for suspicious events.

1. In **Server Manager**, select **Tools > Event Viewer**.

2. Browse to **Applications and Services Logs > Microsoft > Windows > PowerShell**, and then select the **Operational** log.

3. From the **Actions** menu on the right, select **Filter Current Log**.

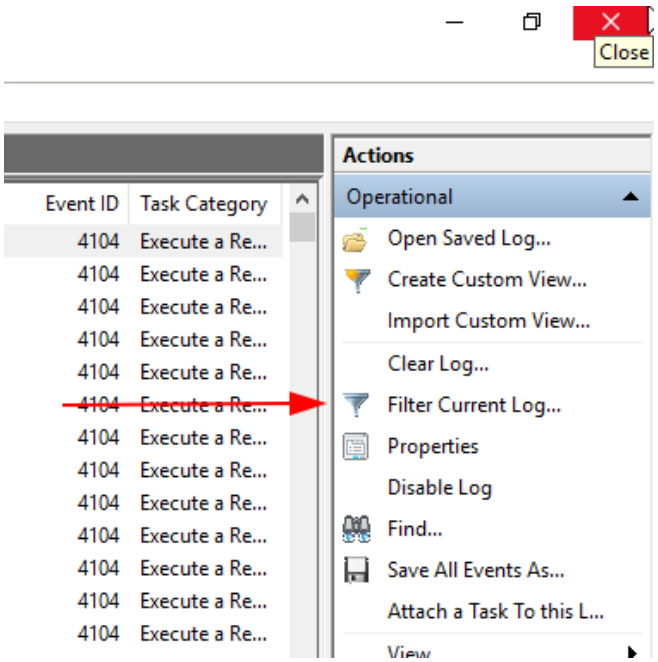

Figure 6.1 – Selecting Filter Current Log under Actions.

4. Replace the <**All Event IDs**> field with Event ID **4100**, and then select **OK**.

5. Browse the displayed entries. You should see text indicating that the **PSHello.ps1** script could not be run because it is unsigned.

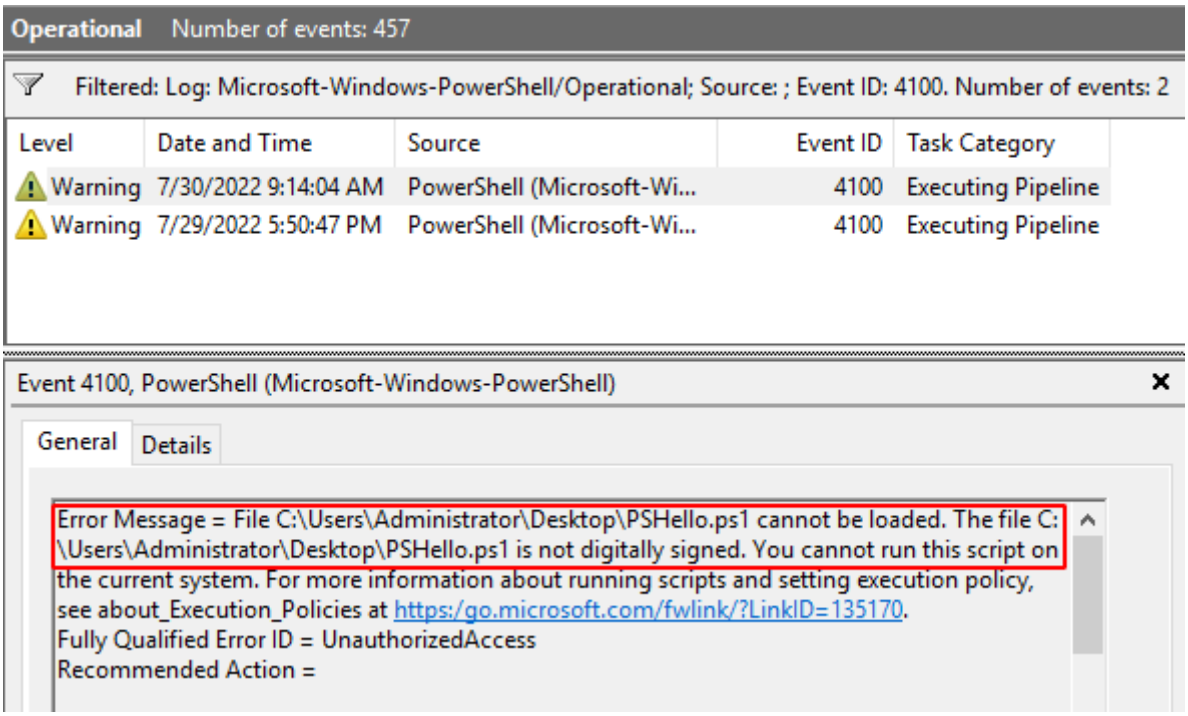

Figure 6.2 – Event ID 4100 showing an error when executing PSHello.ps1.

NOTE: Attempting to run an unsigned script was an earlier task in the activity, and is an example of a suspicious entry. Such entries indicate someone may be trying to run unauthorized scripts in your environment.