

How to List Local Administrators Using PowerShell

Task. List local Administrators using PowerShell.

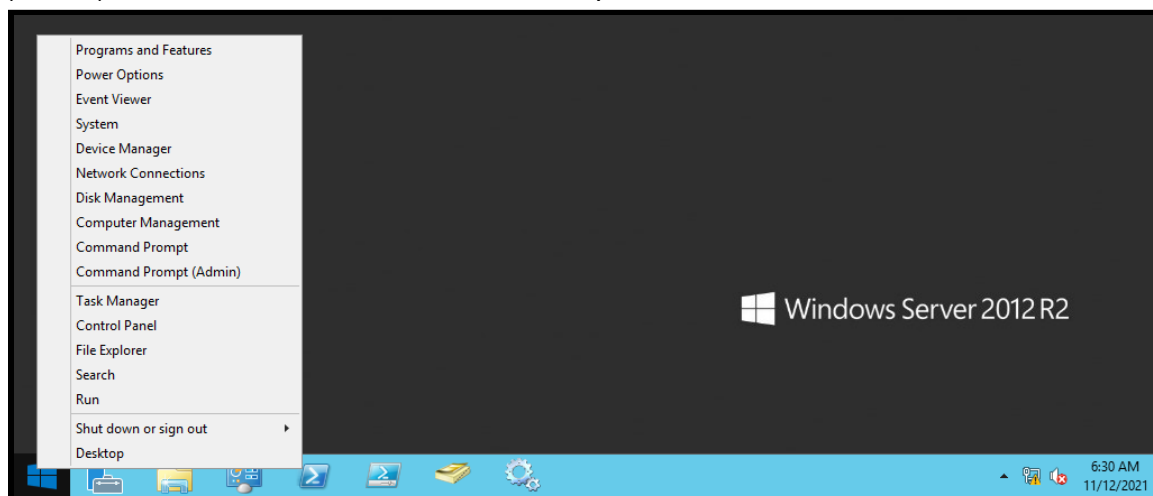
Purpose. The local Administrators group must be monitored for unauthorized changes. Local Administrator accounts have full control of the computers they were created on. Adversaries may seek to obtain and abuse these credentials as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion.

Conditions. You have domain administrator privileges, access to Windows PowerShell, access to the Windows PowerShell Integrated Scripting Environment (ISE), access to either a domain controller or a workstation with Remote Server Administration Tools (RSAT) installed, and Windows Remote Management (WinRM) is enabled across the network.

Standard. You were able to list the members of the local Administrators group using PowerShell.

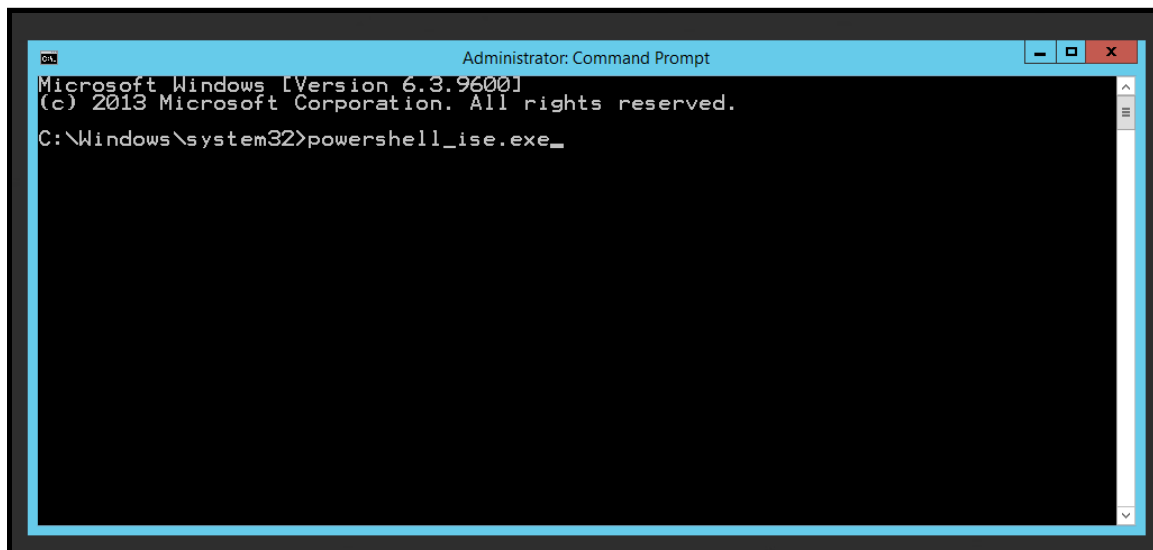
Step 1. Login to your domain administrator account on either a domain controller or a workstation with RSAT installed.

Step 2. Right-click on the Windows icon in the bottom-left corner and select “Command Prompt (Admin)” to start an elevated Command Prompt session.

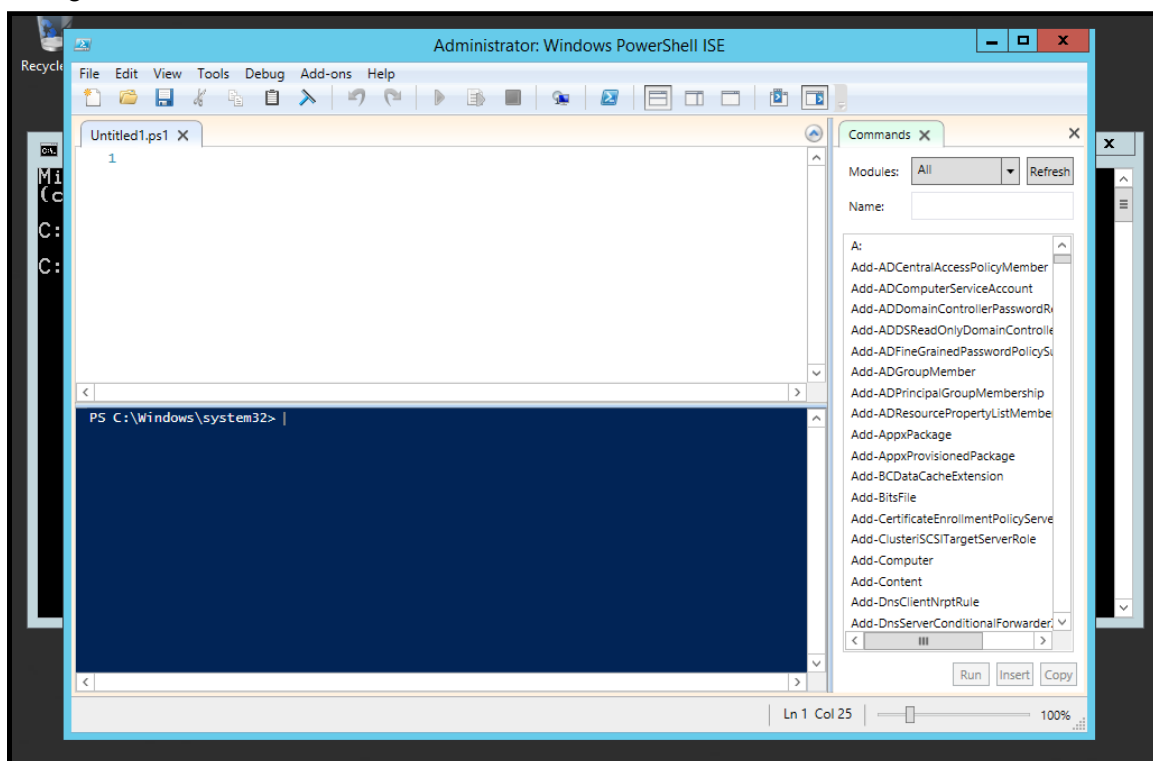


How to List Local Administrators Using PowerShell

Step 3. Type “powershell_ise.exe” to invoke Windows PowerShell ISE.



Once your point-of-view looks like the screenshot below, click-on the “Maximize” button in the top-right corner of the Windows PowerShell ISE window. Then, close the “Command” pane on the right.



How to List Local Administrators Using PowerShell

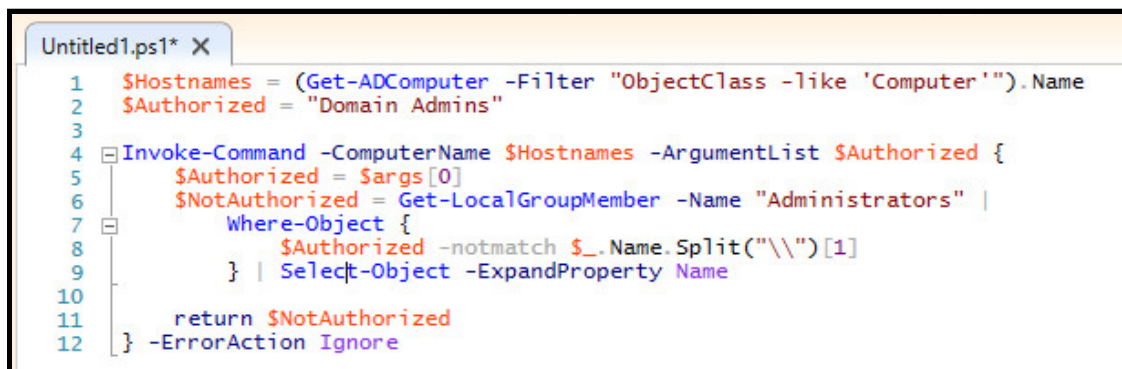
Step 5. Type the command sentences below in the Script pane of Windows PowerShell ISE to specify which accounts are authorized to be a member of the local Administrators group. This will help automate your analysis and ensure authorized accounts are excluded from the output generated.

```
$Hostnames = (Get-ADComputer -Filter "ObjectClass -like 'Computer']").Name
$Authorized = "Domain Admins"

Invoke-Command -ComputerName $Hostnames -ArgumentList $Authorized {
    $Authorized = $args[0]
    $NotAuthorized = Get-LocalGroupMember -Name "administrators" |
        Where-Object {
            $Authorized -notmatch $_.Name.split("\\")[1]
        } | Select-Object -ExpandProperty Name

    return $NotAuthorized
} -ErrorAction Ignore
```

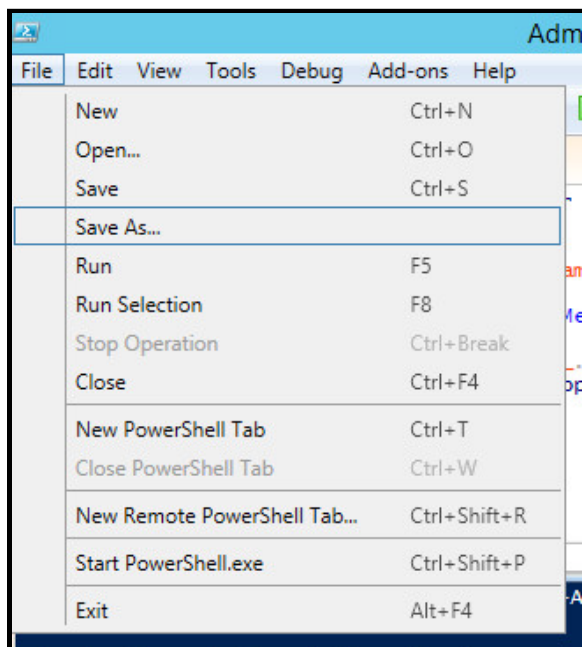
Your point-of-view should look like the screenshot below.

A screenshot of the Windows PowerShell ISE interface. The title bar shows 'Untitled1.ps1* X'. The script editor contains the following PowerShell code:

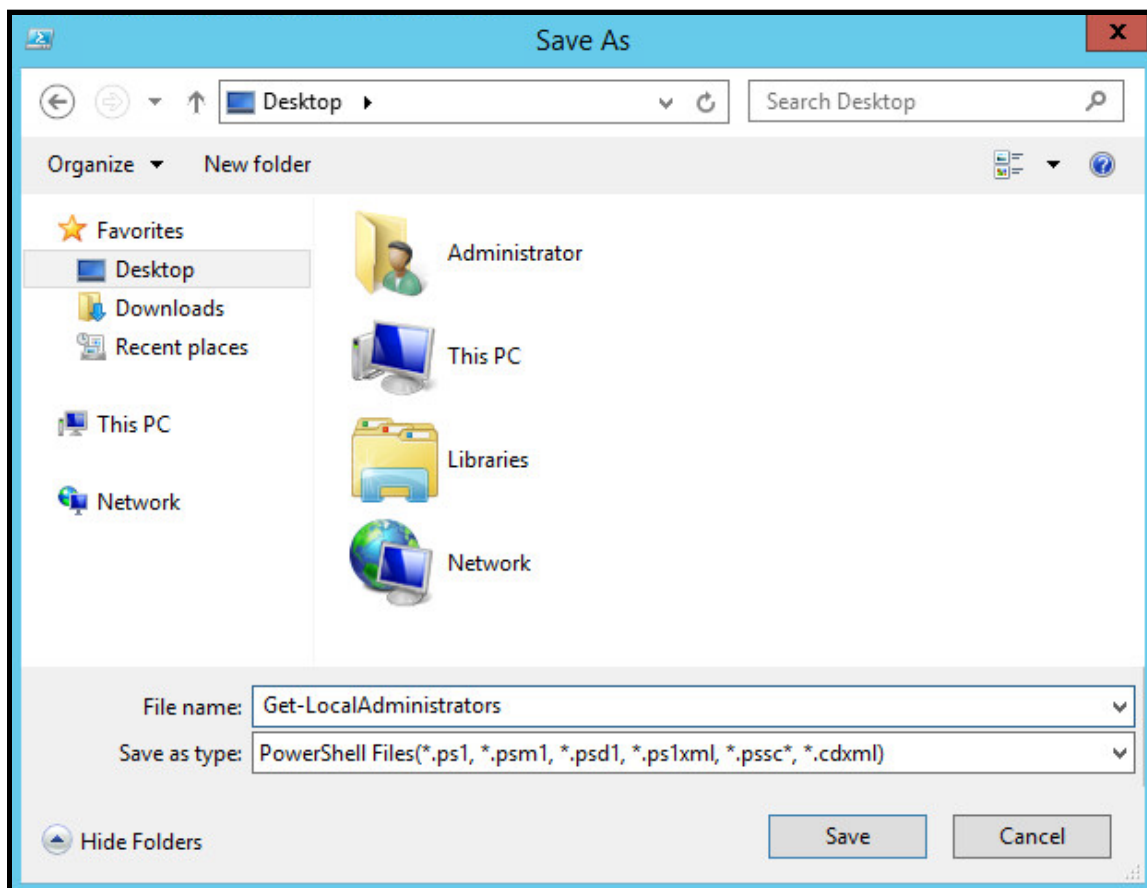
```
1 $Hostnames = (Get-ADComputer -Filter "ObjectClass -like 'Computer']").Name
2 $Authorized = "Domain Admins"
3
4 Invoke-Command -ComputerName $Hostnames -ArgumentList $Authorized {
5     $Authorized = $args[0]
6     $NotAuthorized = Get-LocalGroupMember -Name "Administrators" |
7         Where-Object {
8             $Authorized -notmatch $_.Name.Split("\\")[1]
9         } | Select-Object -ExpandProperty Name
10
11     return $NotAuthorized
12 } -ErrorAction Ignore
```

How to List Local Administrators Using PowerShell

Step 7. Click-on “File > Save As...”

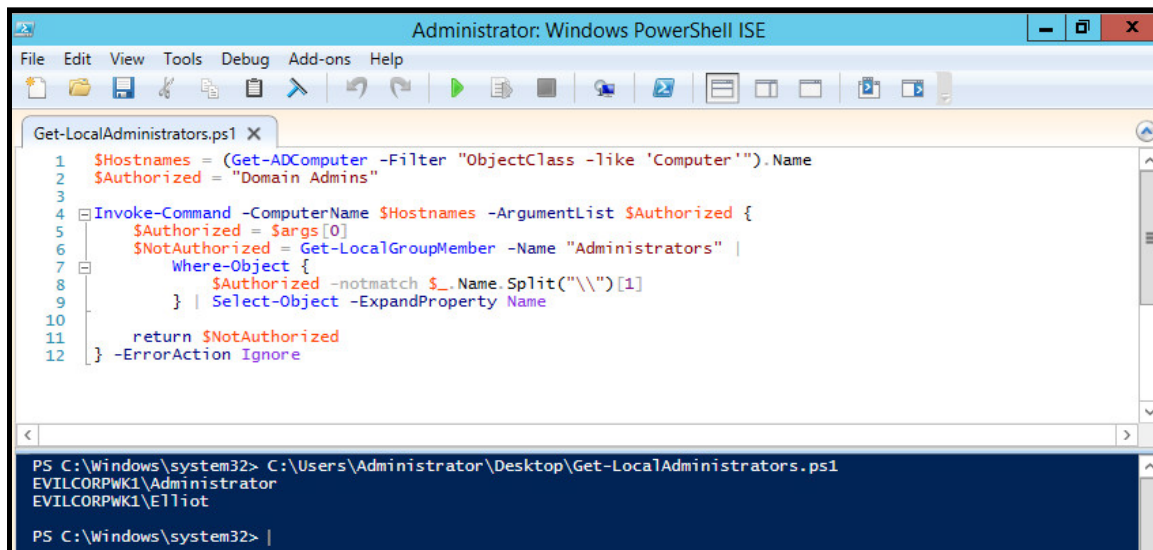


When prompted, save the file to the “Desktop” and call it “Get-LocalAdministrators.”



How to List Local Administrators Using PowerShell

Step 7. Click-on the “Run Script” button (it looks like a green “play” symbol). If you need to troubleshoot, remove the “-ErrorAction Ignore” option from the script (DO NOT remove the curly brace preceding this option).

The screenshot shows the Windows PowerShell ISE interface. The title bar reads "Administrator: Windows PowerShell ISE". The menu bar includes File, Edit, View, Tools, Debug, Add-ons, and Help. The toolbar contains various icons for file operations and execution. The script editor shows a file named "Get-LocalAdministrators.ps1" with the following code:

```
1 $Hostnames = (Get-ADComputer -Filter "ObjectClass -like 'Computer'").Name
2 $Authorized = "Domain Admins"
3
4 Invoke-Command -ComputerName $Hostnames -ArgumentList $Authorized {
5     $Authorized = $args[0]
6     $NotAuthorized = Get-LocalGroupMember -Name "Administrators" |
7         Where-Object {
8             $Authorized -notmatch $_.Name.Split("\\")[1]
9         } | Select-Object -ExpandProperty Name
10
11     return $NotAuthorized
12 } -ErrorAction Ignore
```

The console window at the bottom shows the execution of the script from a system32 prompt. The output lists the local administrators: "EVILCORPWK1\Administrator" and "EVILCORPWK1\Elliot".

```
PS C:\Windows\system32> C:\Users\Administrator\Desktop\Get-LocalAdministrators.ps1
EVILCORPWK1\Administrator
EVILCORPWK1\Elliot
PS C:\Windows\system32> |
```