15 ways to bypass the PowerShell Execution Policy

The PowerShell execution policy is the setting that determines which type of PowerShell scripts (if any) can be run on the system. By default it is set to "Restricted". You can take a look at the current configuration with the “Get-ExecutionPolicy” PowerShell command.

Bypassing the PowerShell Execution Policy:

1. Copy and paste your PowerShell script into an interactive console. However, keep in mind that you will be limited by your current user’s privileges. This is the most basic example and can be handy for running quick scripts when you have an interactive console.

2. Echo the Script and Pipe it to PowerShell Standard In. Simply ECHO your script into PowerShell standard input.

3. Use the Windows “type” command or PowerShell “Get-Content” command to read your script from the disk and pipe it into PowerShell standard input. You could read it from a network share if you’re trying to avoid writing to the disk.

4. Download Script from URL and Execute with Invoke Expression. This technique can be used to download a PowerShell script from the internet and execute it without having to write to disk.

5. Use the Command Switch This technique is very similar to executing a script via copy and paste, but it can be done without the interactive console. It’s nice for simple script execution, but more complex scripts usually end up with parsing errors. You can place these types of PowerShell commands into batch files and place them into autorun locations (like the all users startup folder) to help during privilege escalation.

6. Use the EncodeCommand Switch. This is very similar to the “Command” switch, but all scripts are provided as a Unicode/base64 encoded string.

7. Use the Invoke-Command Cmdlet. It’s typically executed through an interactive PowerShell console or one liner using the “Command” switch, but it can also be used to execute commands against remote systems where PowerShell remoting has been enabled.

8. Use the Invoke-Expression Cmdlet. This is another one that's typically executed through an interactive PowerShell console or one liner using the "Command" switch.

9. Use the "Bypass" Execution Policy Flag. When this flag is used Microsoft states that "Nothing is blocked and there are no warnings or prompts".

10. Use the "Unrestricted" Execution Policy Flag. This similar to the "Bypass" flag. However, when this flag is used Microsoft states that it "Loads all configuration files and runs all scripts. If you run an unsigned script that was downloaded from the Internet, you are prompted for permission before it runs."

11. Use the "Remote-Signed" Execution Policy Flag. Create your script then sighn it. Finally, run it using the command below:

- *PowerShell.exe -ExecutionPolicy Remote-signed -File .runme.ps1*

12. Disable ExecutionPolicy by Swapping out the AuthorizationManager This is one of the more creative approaches. A function can be executed via an interactive PowerShell console or by using the "command" switch. Once the function is called it will swap out the "AuthorizationManager" with null. As a result, the execution policy is essentially set to unrestricted for the remainder of the session.

13. Set the ExecutionPolicyfor the Process Scope. The execution policy can be applied at many levels. This includes the process which you have control over. Using this technique the execution policy can be set to unrestricted for the duration of your Session.

14. Set the ExecutionPolicyfor the CurrentUser Scope via Command This option is similar to the process scope, but applies the setting to the current user's environment persistently by modifying a registry key.

15. Set the ExecutionPolicyfor the CurrentUser Scope via the Registry. Change the execution policy for the current user's environment persistently by modifying a registry key directly.

- *HKEY_CURRENT_USER\Software\Microsoft\PowerShell\ 1 \ShellIds\Microsoft.PowerShell*