Lab Notes: Persistence and Privilege Elevation using the Powershell Profile

Archived: 2026-04-05 17:16:28 UTC

TL;DR;

A recent ESET blog post mentions a persistence technique I wasn't previously aware of that is in use by the Turla APT group. The technique leverages the PowerShell profile to sabotage PowerShell in a way that executes arbitrary code every time Powershell is launched, upon testing I've discovered this technique may also provide a low and slow vector to Domain Admin, and other privileged admin or services accounts by leveraging common flaws admin scripts, asset management systems, and enterprise detection and response tools. This post captures my observations working from Matt Nelson's 2014 blog post (Apologies to the researcher if there is prior art I'm unaware of at the time of this post)

Privilege Elevation - Local Admin to Sloppy Admin

Setup Requirements:

1. In my testing, you need local admin rights to create the global profile
    1. $profile.AllUsersAllHosts
    2. AKA C:\Windows\System32\WindowsPowerShell\v1.0\profile.ps1
2. This does not bypass Execution Policy (check with Get-ExecutionPolicy).
    1. If it's set to AllSigned or Restricted, not only will the code not execute; the end user might notice a suspicious error message reminding them of the execution policy. (By default a Window 10 endpoint is Restricted)
3. A privileged user or preferably an automated task that runs PowerShell on the 0wned box with elevated domain privileges is needed. They also need to forget to pass –NoProfile flag when launching it (which seems like just about everything and everybody in a large enterprise)

Now any code you place in this global profile will be run by any user who launches PowerShell. We just decide what kind of PowerShell script we want our sloppy admin to execute, set our trap, and patiently wait.

As a POC I used 1 line of code:

Add-Content c:\windows\temp\test1.txt "$(Get-Date) Profile POC Launched by $(whoami)"

Within the hour a friendly enterprise asset management system ran my arbitrary code using SYSTEM, but with a phone call to IT and some trivial social engineering, this could have easily been one of the desktop admins.

Mitigation:

1. Similar to detecting persistence in the startup folder, if you can audit file writes and modifications to C:\Windows\System32\WindowsPowerShell\v1.0\profile.ps1 you can alert on this in real time. Most userbases will not be making frequent changes to this file, which should leave you with a low noise high fidelity alert
2. If you need another reason to preach the gospel of a restrictive PowerShell execution policy this may be it. Unfortunately, if your admins are using it already good luck telling them they can't use PowerShell
3. You can also audit to ensure any privileged accounts executing PowerShell on remote systems always invokes the –NoProfile command line argument

Persistence

For persistence, things are much simpler. Aforementioned mitigations 1 and 2 still apply, but the only requirement is the lax execution policy. Every user should have access to edit their own $profile and any code placed here will be executed anytime PowerShell is launched under that user context.

One Line POC:
Add-Content $profile "Invoke-Item C:\Windows\System32\calc.exe"

For detection, we need to monitor a few additional file locations, but the alert volume should still be manageable:

- C:\Windows\System32\WindowsPowerShell\v1.0\profile.ps1
- $Home\[My]Documents\WindowsPowerShell\Profile.ps1
- $Home\[My]Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1
- $PsHome\Profile.ps1
- $PsHome\Microsoft.PowerShell_profile.ps1
- $Home\[My]Documents\PowerShell\Profile.ps1
- $Home\[My]Documents\PowerShell\Microsoft.PowerShell_profile.ps1

Resources:

1. Microsoft Documentation On Powershell Profiles https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_profiles?view=powershell-6
2. Abusing Powershell Profiles https://enigma0x3.net/2014/06/16/abusing-powershell-profiles/
3. Turla Powershell Usage https://www.welivesecurity.com/2019/05/29/turla-powershell-usage/
