Defense Spotlight: Finding Hidden Windows Services

By Joshua Wright

Published: 2020-10-14 · Archived: 2026-04-02 11:53:57 UTC

In the previous article, I wrote about a technique that can be used to hide a Windows service from view using standard service enumeration techniques including Get-Service, sc query, and services.exe:

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

After establishing access to a system, an attacker can use this as a means to obtain persistent access to the host using the Command & Control (C2) mechanism of their choosing, evading typical service enumeration techniques.

If an attacker hides a service using the sc sdset technique, Windows will generate a logging event: Security log Event ID 4674:

```
 PS C:\WINDOWS\system32> $dacl="D:(D;;DCLCWPDTSD;;;IU)(D;;DCLCWPDTSD;;;SU)(D;;DCLCWPDTSD;;;BA)(A;;CCLCSWLOCRRC;
PS C:\WINDOWS\system32> Clear-Eventlog -Logname Security ; & $env:SystemRoot\System32\sc.exe sdset SWCUEngine $d
[SC] SetServiceObjectSecurity SUCCESS

EventID : 4674
Message : An operation was attempted on a privileged object.

          Subject:
                Security ID:            S-1-5-21-2977773840-2930198165-1551093962-1000
                Account Name:           Sec504
                Account Domain:         SEC504STUDENT
                Logon ID:               0x35aec

          Object:
                Object Server:  SC Manager
                Object Type:    SERVICE OBJECT
                Object Name:    SWCUEngine
                Object Handle:  0xffffbb06ee98e928

          Process Information:
                Process ID:     0x25c
```

```
                Process Name:   C:\Windows\System32\services.exe

          Requested Operation:
                Desired Access: %%1539

                Privileges:             SeSecurityPrivilege

EventID : 1102
Message : The audit log was cleared.
          Subject:
                Security ID:    S-1-5-21-2977773840-2930198165-1551093962-1000
                Account Name:   Sec504
                Domain Name:    SEC504STUDENT
                Logon ID:       0x35aec

PS C:\WINDOWS\system32>
```

Using the Splunk Boss of the SOC v3 data (courtesy of the amazing Dave Herrald), it appears that Security Event ID 4674 *An operation was attempted on a privileged object* events are pretty rare indeed, particularly when applied to the services.exe process. This seems like a good candidate for a DeepBlueCLI event detect.

First, you'll need to download DeepBlueCLI. This is easiest if you've already installed Git on your system:

```
 PS C:\> mkdir c:\tools
PS C:\> cd tools
PS C:\tools> git clone https://github.com/sans-blue-team/DeepBlueCLI.git
Cloning into 'DeepBlueCLI'...
remote: Enumerating objects: 31, done.
remote: Counting objects: 100% (31/31), done.
remote: Compressing objects: 100% (21/21), done.
Receiving objects: 100% (493/493), 5.56 MiB | 13.85 MiB/s, done.
Resolving deltas: 100% (271/271), done.
```

By default, DeepBlue.ps1 reads from the local Security log, which will identify this attack:

```
 PS C:\tools> cd .\DeepBlueCLI\
PS C:\tools\DeepBlueCLI> .\DeepBlue.ps1

Date    : 10/13/2020 10:59:21 AM
Log     : Security
EventID : 4674
Message : Possible Hidden Service Attempt
Results : User requested to modify the Dynamic Access Control (DAC) permissions of a service,
          possibly to hide it from view.
          User: Sec504
          Target service: SWCUEngine
```

```
          Desired Access: WRITE_DAC

Command :
Decoded :
```

A sample [EVTX log capture is also available for testing](#).

This detect is useful since it also reveals the target service name. You can confirm that the service is hidden by attempting to enumerate it and to interrogate it directly. First, we confirm that the service is hidden:

```
 PS C:\tools\DeepBlueCLI> Get-Service | Select-Object Name | Select-String -Pattern 'SWCUEngine'
PS C:\tools\DeepBlueCLI>
```

In this output we see that the service name is not displayed in the output of Get-Service. Since we know the name of the service from the DeepBlueCLI output though, we can attempt to control the service using Set-Service:

```
 PS C:\tools\DeepBlueCLI> 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
...
```

Notice how in the Set-Service command, the error message is *Access is denied*. Compared to a service that really does not exist, shown below:

```
 PS C:\tools\DeepBlueCLI> Set-Service -Name JoshNonexistentService -Status Stopped
Set-Service : Service JoshNonexistentService was not found on computer '.'.
At line:1 char:1
...
```

The message *Service ... was not found* indicates that there really is no service, while the message *Access is denied* tells us that it does exist, and is hidden on the local system.

This detection mechanism is useful, but is still problematic. What if an attacker clears the event log? What if an attacker [purges the log entry](#)? What if the event log information rolls over prior to detection?

Fortunately, Windows also maintains a registry key for each service in HKLM\System\CurrentControlSet\Services. We can cross-reference the registry key list with the service name list from Get-Service to identify any outliers. This gets a little complicated since there are multiple entries in the Services registry key for Windows drivers (not services) and duplicate [per-user services](#) with a locally unique identifier (LUID) at the end of the service name.

Joshua Wright 11:04 AM
@Jon, and I put together a script to identify hidden services on Windows:

```
Compare-Object -ReferenceObject (Get-Service | Select-Object -ExpandProperty Name | % { $_ -replace "_[0-9a-f]{2,8}$" } ) -DifferenceObject (gci -path hklm:\system\currentcontrolset\services | % { $_.Name -Replace "HKEY_LOCAL_MACHINE\\","HKLM:\" } | Where { Get-ItemProperty -Path "$_" -name objectname -erroraction 'ignore' } | % { $_.substring(40) }) -PassThru | ?{$_.sideIndicator -eq "=>"}
```

Tim Medin 11:05 AM
ew

Joshua Wright 11:05 AM
@Tim Medin Don't hate the player.

```
 PS C:\> Compare-Object `
>>         -ReferenceObject (Get-Service |
>>                 Select-Object -ExpandProperty Name |
>>                 % { $_ -replace "_[0-9a-f]{2,8}$" } ) `
>>         -DifferenceObject (gci -path hklm:\system\currentcontrolset\services |
>>                 % { $_.Name -Replace "HKEY_LOCAL_MACHINE\\","HKLM:\" } |
>>                 ? { Get-ItemProperty -Path "$_" -name objectname -erroraction 'ignore' } |
>>                 % { $_.substring(40) }) -PassThru |
>>         ? { $_.sideIndicator -eq "=>" }
SWCUEngine
WUDFWpdFs
PS C:\>
```

By enumerating the registry for services and comparing against the output of Get-Service, we can identify hidden services. The output here reveals SWCUEngine as hidden, but it also reveals WUDFWpdFs (Microsoft Windows Portable Devices file system driver) as hidden as well. We can confirm this by using the Set-Service command as shown earlier in this article:

```
 PS C:\> Set-Service -Name WUDFWpdFs -Status Stopped
Set-Service : Service 'WUDFWpdFs (WUDFWpdFs)' cannot be stopped due to the following error: Cannot stop WUDFWpdF
At line:1 char:1
...
```

Again here, we can confirm that WUDFWpdFs is a service from the lack of a *was not found on computer* error message.

Hiding Windows services is a nice opportunity for an adversary to try to avoid detection, but once defenders know that this is possible, then it becomes an easy detect. Consider adding the checkhiddensvc.ps1 script to your next threat-hunting exercise.

Special thanks to Jon Gorenflo for help in researching detection methods for this article!
