Microsoft-Signed Alternate PowerShell Hosts

Living of the Land++

Abusing Alternate Signed PowerShell Hosts

Why bother?

- Application whitelisting
  - Someone thought they'd block PowerShell execution by blocking powershell.exe, powershell_ise.exe, wsmprovhost.exe, etc.
  - Most application whitelisting policies will allow anything signed by Microsoft to run except tools know to be used for abuse.
  - Depending upon how the PowerShell is invoked, it could also represent a constrained language mode bypass - e.g. runscripthelper.exe
- Detection evasion
  - Evade command-line logging
  - Evade sysmon logging
  - Evade any naive logging based upon traditional PowerShell hosts

Known Alternate PowerShell Hosts

1. wsmprovhost.exe - PowerShell remoting host
2. %windir%\System32\SyncAppvPublishingServer.exe
3. powershellcustomhost.exe - IIS web deploy utility
4. SQLPS.exe
5. sdiagnhost.exe - Windows Troubleshooting Packs
6. runscripthelper.exe - MSFT telemetry code execution FTW! 👎
7. Which ones can you find?

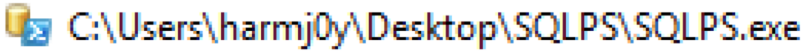
Example: sqlps.exe

sqlps Utility

03/14/2017 • 3 minutes to read •

The **sqlps** utility starts a Windows PowerShell session with the SQL Server PowerShell provider and cmdlets loaded and registered. You can enter PowerShell commands or scripts that use the SQL Server PowerShell components to work with instances of SQL Server and their objects.

Example: sqlps.exe




```
Microsoft (R) SQL Server (R) PowerShell
Version 11.0.6020.0
Copyright (c) 2012 Microsoft. All rights reserved.

PS C:\Users\harmj0y\Desktop\SQLPS> Get-Process

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)      Id  SI ProcessName
-------  ------    -----      -----     ------      --  -- -----------
    351      19     9420      18356       5.78       8   1 ApplicationFrameHost
    160      10     6296      11644       0.19    5420   0 audiodg
    126      10     5676      11684       0.03    1552   1 conhost
    227      13     5476       3508       7.25    2536   1 conhost
    174      12     6144      16728       0.23    2820   1 conhost
    105       8     5316          8       0.03    3300   1 conhost
    224      13     3876      19912       0.31    8832   1 conhost
    179      12     4648      12352       0.09    8968   1 conhost
```

Searching for “Official” hosts

- So how can you go about finding these hosts?
- **Characteristic 1:**
  - These binaries are almost always C#/.NET .exes/.dlls
- **Characteristic 2:**
  - These binaries have **System.Management.Automation.dll** as a referenced assembly
- **Characteristic 3:**
  - These may not always be “built in” binaries

Exercise: Searching for “Official” hosts

```
PS C:\> ls C:\Windows\* -Recurse -Directory -ErrorAction SilentlyContinue | % {
>>     try {
>>         $_.FullName | ls -File -ErrorAction SilentlyContinue | ? { ($_.Extension -eq '.dll') -or ($_.Extension -eq '.exe') } | % {
>>             try {
>>                 $Assembly = [Reflection.Assembly]::ReflectionOnlyLoadFrom($_.FullName)
>>
>>                 if ($Assembly.GetReferencedAssemblies().Name -contains 'System.Management.Automation') {
>>                     $_.FullName
>>                 }
>>             } catch { }
>>         }
>>     } catch { }
>> }
C:\Windows\System32\Microsoft.Uev.CmUtil.dll
C:\Windows\System32\stordiag.exe
C:\Windows\System32\SyncAppvPublishingServer.exe
```

See PowerShellHostFinder.ps1 for code the code snippet.

Abusing Alternate Signed PowerShell Hosts - Demo

Did you find %windir%\System32\~~runscripthelper.exe~~stordiag.exe?

Update: Microsoft removed runscripthelper.exe in Win 10 RS3! It's present in the Day 4 Lab:CLM_Bypass.

Try to find a way to get it to execute your PowerShell code.

Objectives:

1. Determine what command line arguments it accepts
2. Determine the conditions required to have it execute code.
3. Bonus: Determine a way to have it execute code in a non-admin context.

Windows Troubleshooting Packs

- Troubleshooting Packs “deal with common problems such as problems that are related to printers, displays, sound, networking, system performance, and hardware compatibility.”
- Stored in %windir%\diagnostics
- They are driven by PowerShell under the hood.
- Associated with the .diagcab and .diagpkg extensions.
- Invoked with msdt.exe or Invoke-TroubleshootingPack cmdlet
- These are the sorts of things that would likely be ignored by defenders as they are common noise generators.

Windows Troubleshooting Packs

- Great guide on building your own malicious Troubleshooting Packs
  - https://cybersyndicates.com/2015/10/a-no-bull-guide-to-malicious-windows-trouble-shooting-packs-and-application-whitelist-bypass/
- We're going to hijack legitimate, signed ones though. ;)
- To get started, we need procmon…
- Double click on %windir%\diagnostics\system\AERO\DiagPackage.diagpkg
- Click through the dialogs and then end your procmon trace

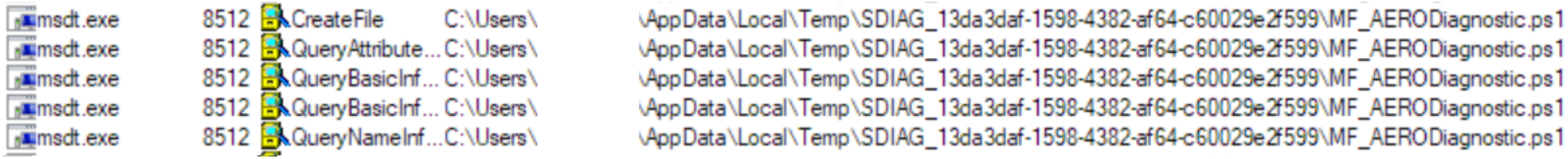
Windows Troubleshooting Packs

Microsoft.Windows.Diagnosis.SDCommon.(ni.)dll

```
private void ExecuteCommand(PowerShell ps)
{
    try
    {
        object @lock = this.m_Lock;
        lock (@lock)
        {
            this.m_PowerShell = ps;
        }
        ps.Invoke();
    }
```

Windows Troubleshooting Packs

%windir%\diagnostics\system\AERO\DiagPackage.diagpkg




```
<Script>
  <Parameters/>
  <ProcessArchitecture>Any</ProcessArchitecture>
  <RequiresElevation>false</RequiresElevation>
  <RequiresInteractivity>true</RequiresInteractivity>
  <FileName>MF_AERODiagnostic.ps1</FileName>
  <ExtensionPoint/>
</Script>
```

Windows Troubleshooting Packs

```
Command line:   "C:\WINDOWS\system32\msdt.exe" /path
"C:\Windows\diagnostics\system\AERO\DiagPackage.diagpkg"
Current directory:   C:\Windows\diagnostics\system\AERO\


Command line:   C:\WINDOWS\System32\sdiagnhost.exe -Embedding
Current directory:   C:\WINDOWS\system32\
```

- Doesn’t appear to be logged in the “Windows PowerShell” log
- Invocation is captured with scriptblock logging though.

Windows Troubleshooting Packs

Hijack/weaponization strategy:

1. PowerShell files are written to %TEMP%. An attacker controls read/write.
2. Ideally avoid using PowerShell to weaponize. Using PowerShell kind of defeats the point of using an alternate PowerShell host.
3. An attacker would need to hijack the existing code and “win the race” to get code execution.
4. Note the SDIAG_<UNIQUE_GUID> directory created.

This would be a good time to attempt

Lab: Windows Troubleshooting Packs