LEE HOLMES

Precision Computing - Software Design and Development

DETECTING AND PREVENTING POWERSHELL DOWNGRADE ATTACKS

Fri, Mar 17, 2017 4-minute read

With the advent of PowerShell v5's awesome new security features, old versions of PowerShell have all of the sudden become much more attractive for attackers and Red Teams.

PowerShell Downgrade Attacks

There are two ways to do this:

Command Line Version Parameter

The simplest technique is:

```
PowerShell –Version 2 –Command <…>
```

(or of course any of the –Version abbreviations). PowerShell.exe itself is just a simple native application that hosts the CLR, and the –Version switch tells PowerShell which version of the PowerShell assemblies to load. Unfortunately, the PowerShell v5 enhancements did NOT include time travel, so the v2 binaries that were shipped in 2008 did NOT include the code we wrote in 2014. The 2.0 .NET Framework (which is required for PowerShell's V2 engine) is not included by default in Win10+, but an attacker or Red Teamer could enable it or install it. Prior to Windows 10, where it is available by default, they could just use it.

Hosting Applications Compiled using V2 Reference Assemblies

When somebody compiles a C# application to leverage the PowerShell engine, they link against reference assemblies when they do that. If they link against the PowerShell v2 reference assemblies during development, Windows will use the PowerShell v2 engine (if available) when the application runs. Otherwise, PowerShell's type forwarding will run the application using the currently installed PowerShell engine. This is what happens when PowerShell Empire's "psinject" module attempts to load PowerShell into another process (such as notepad).

Detection and Prevention

You have several options to detect and prevent PowerShell Downgrade Attacks.

Event Log

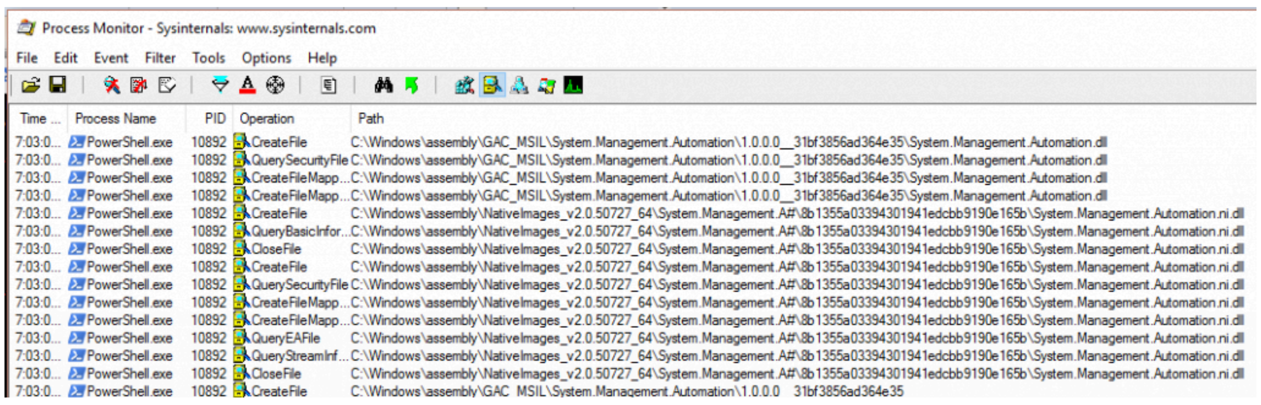
As a detection mechanism, the "Windows PowerShell" classic event log has event ID 400. This is the "Engine Lifecycle" event, and includes the Engine Version. Here is an example query to find lower versions of the PowerShell engine being loaded:

```
Get-WinEvent -LogName "Windows PowerShell" |
    Where-Object Id -eq 400 |
    Foreach-Object {
        $version = [Version] (
            $_.Message -replace '(?s).*EngineVersion=([\d\.]+)*.*','$1')
        if($version -lt ([Version] "5.0")) { $_ }
}
```

LEE HOLMES

Precision Computing - Software Design and Development

When the CLR loads PowerShell assemblies, it will first load the managed assemblies from the GAC (if they are available). It will also load the native images that contain pre-jitted code if the assemblies are NGEN'd (which they are). Here is what loading PowerShell v2 looks like:



Process Monitor - Sysinternals: www.sysinternals.com
File Edit Event Filter Tools Options Help

```
Time ...  Process Name    PID    Operation          Path
7:03:0... PowerShell.exe  10892  CreateFile         C:\Windows\assembly\GAC_MSIL\System.Management.Automation\1.0.0.0__31bf3856ad364e35\System.Management.Automation.dll
7:03:0... PowerShell.exe  10892  QuerySecurityFile  C:\Windows\assembly\GAC_MSIL\System.Management.Automation\1.0.0.0__31bf3856ad364e35\System.Management.Automation.dll
7:03:0... PowerShell.exe  10892  CreateFileMapp...  C:\Windows\assembly\GAC_MSIL\System.Management.Automation\1.0.0.0__31bf3856ad364e35\System.Management.Automation.dll
7:03:0... PowerShell.exe  10892  CreateFileMapp...  C:\Windows\assembly\GAC_MSIL\System.Management.Automation\1.0.0.0__31bf3856ad364e35\System.Management.Automation.dll
7:03:0... PowerShell.exe  10892  CreateFile         C:\Windows\assembly\NativeImages_v2.0.50727_64\System.Management.A#\8b1355a0339430194 1edcbb9190e165b\System.Management.Automation.ni.dll
7:03:0... PowerShell.exe  10892  QueryBasicInfor... C:\Windows\assembly\NativeImages_v2.0.50727_64\System.Management.A#\8b1355a03394301941edcbb9190e165b\System.Management.Automation.ni.dll
7:03:0... PowerShell.exe  10892  CloseFile          C:\Windows\assembly\NativeImages_v2.0.50727_64\System.Management.A#\8b1355a03394301941edcbb9190e165b\System.Management.Automation.ni.dll
7:03:0... PowerShell.exe  10892  CreateFile         C:\Windows\assembly\NativeImages_v2.0.50727_64\System.Management.A#\8b1355a03394301941edcbb9190e165b\System.Management.Automation.ni.dll
7:03:0... PowerShell.exe  10892  QuerySecurityFile  C:\Windows\assembly\NativeImages_v2.0.50727_64\System.Management.A#\8b1355a03394301941edcbb9190e165b\System.Management.Automation.ni.dll
7:03:0... PowerShell.exe  10892  CreateFileMapp...  C:\Windows\assembly\NativeImages_v2.0.50727_64\System.Management.A#\8b1355a03394301941edcbb9190e165b\System.Management.Automation.ni.dll
7:03:0... PowerShell.exe  10892  CreateFileMapp...  C:\Windows\assembly\NativeImages_v2.0.50727_64\System.Management.A#\8b1355a03394301941edcbb9190e165b\System.Management.Automation.ni.dll
7:03:0... PowerShell.exe  10892  QueryEAFile        C:\Windows\assembly\NativeImages_v2.0.50727_64\System.Management.A#\8b1355a03394301941edcbb9190e165b\System.Management.Automation.ni.dll
7:03:0... PowerShell.exe  10892  QueryStreamInf...  C:\Windows\assembly\NativeImages_v2.0.50727_64\System.Management.A#\8b1355a03394301941edcbb9190e165b\System.Management.Automation.ni.dll
7:03:0... PowerShell.exe  10892  CloseFile          C:\Windows\assembly\NativeImages_v2.0.50727_64\System.Management.A#\8b1355a03394301941edcbb9190e165b\System.Management.Automation.ni.dll
7:03:0... PowerShell.exe  10892  CreateFile         C:\Windows\assembly\GAC_MSIL\System.Management.Automation\1.0.0.0__31bf3856ad364e35
```

These can either be an audit trigger, or can be blocked outright. Be careful to not be too selective on the directories you monitor, as the CLR can also load assemblies from specific directories. For example, it is possible to use the CLR's undocumented / unsupported DEVPATH environment variable to force the CLR to use a specified version of the assemblies rather than the GAC'd version. And if you don't have a GAC'd version to override, PowerShell will do regular LoadLibrary() probing to find one – including its installation directory. In addition, PowerShell can either be launched as a 32-bit process, or 64-bit process. A 64-bit system will load 64-bit PowerShell by default. A 32-bit system will load 32-bit PowerShell. On a 64-bit system, though, Windows will implicitly change the version of PowerShell that gets launched by looking at the bitness of the launching application: a 32-bit app will load other 32-bit apps. It is also possible for users or applications to do this explicitly by launching PowerShell from the WOW directory: c:\windows\syswow64\windowspowershell\v1.0\powershell.exe.

```
PS > dir \*.dll -rec -ea ig | % FullName | ? { $\_ -match 'System\\.Management
C:\\windows\\assembly\\GAC\_MSIL\\System.Management.Automation\\1.0.0.0\_\_31b
C:\\windows\\assembly\\NativeImages\_v2.0.50727\_64\\System.Management.A#\\8b1
C:\\windows\\assembly\\NativeImages\_v4.0.30319\_32\\System.Manaa57fc8cc#\\08c
C:\\windows\\assembly\\NativeImages\_v4.0.30319\_64\\System.Manaa57fc8cc#\\407
C:\\windows\\Microsoft.NET\\assembly\\GAC\_MSIL\\System.Management.Automation\
```

If you're going down the enforcement route via AppLocker or Device Guard path, the most robust solution is to block earlier versions of the PowerShell engine by version. Be sure to block both the native image and MSIL assemblies:

```
C:\Users\leeholm>powershell -version 2 -noprofile -command "(Get-Item ([PSObje

ProductVersion   FileVersion      FileName
--------------   -----------      --------
6.1.7600.16385   6.1.7600.16385   C:\WINDOWS\assembly\GAC_MSIL\System.Manageme


C:\Users\leeholm>powershell -noprofile -command "(Get-Item ([PSObject].Assembl

ProductVersion   FileVersion      FileName
--------------   -----------      --------
10.0.14986.1000  10.0.14986.1000  C:\WINDOWS\Microsoft.Net\assembly\GAC_MSIL\S


C:\Users\leeholm>powershell -version 2 -noprofile -command "(Get-Item (Get-Pro

ProductVersion   FileVersion      FileName
--------------   -----------      --------
6.1.7600.16385   6.1.7600.16385   C:\WINDOWS\assembly\NativeImages_v2.0.50727_


C:\Users\leeholm>powershell -noprofile -command "(Get-Item (Get-Process -id $p

ProductVersion   FileVersion      FileName
--------------   -----------      --------
10.0.14986.1000  10.0.14986.1000  C:\WINDOWS\assembly\NativeImages_v4.0.30319_
```

#PowerShell #security

Click to Load Comments

LEE HOLMES

Precision Computing - Software Design and Development