EMBERSEC

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

AUTHORS

Ken Jenkins

Jimmy Bayne

Luke Willadsen

Bradley Wolfenden

Fairuz Rafique

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)

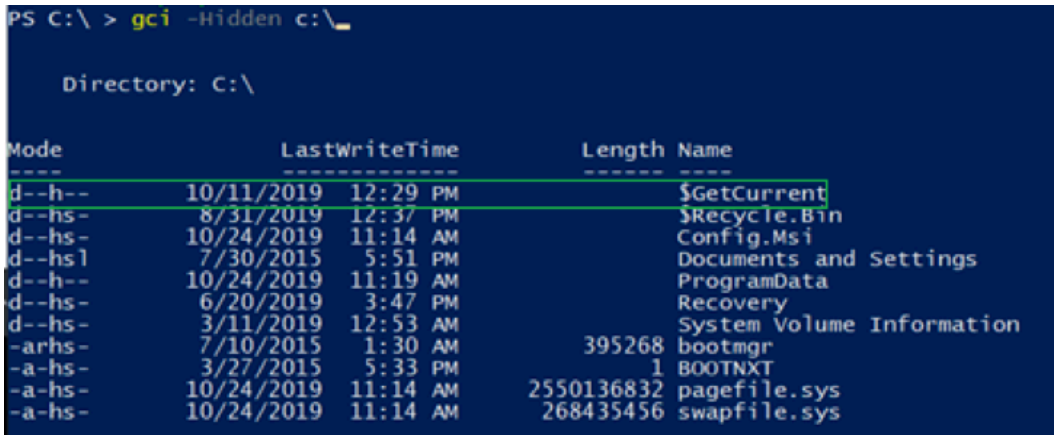
In addition to monthly updates, Microsoft releases major OS "feature" updates such as Version 1903 (released in May 2019) and Version 1909 (released this month). Interestingly, Microsoft provides an easy, alternate way to facilitate these feature updates with the Windows 10 Update Assistant (WUA), an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

Vulnerability Discovery Walkthrough

After installing WUA and updating to Windows 10 1903 on one

me discover interesting LOLBINs and vectors for defense evasion (Shameless plug – some of these adventures have 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.

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

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

Figure 6: WUA Updater

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 structure, including the ***SafeOS*** directory and the ***SetupComplete.cmd*** script. 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:

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

occur.

During the "WUA installer" update phase, the machines goes in [illegible] 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 tampering and a very out-of-place Notepad process 😊 –

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

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] -> Notepad.exe [PID 3324]

Figure 11: Symon Event (WinDeploy)

Figure 12: Symon Event (Cmd – SetupComplete Payload)

Figure 13: Symon Event (Privileged Notepad)

***Note:** We could have potentially used ProcMon and boot

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 -

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

Comments are closed.

Contact Us
(703) 224-1000
info [at] embercybersecurity.com
8484 Westpark Dr.
Suite 600, McLean, VA, 22102