« USERLAND PERSISTENCE WITH SCHEDULED TASKS AND COM HANDLER HIJACKING

"FILELESS" UAC BYPASS USING EVENTVWR.EXE AND REGISTRY HIJACKING »

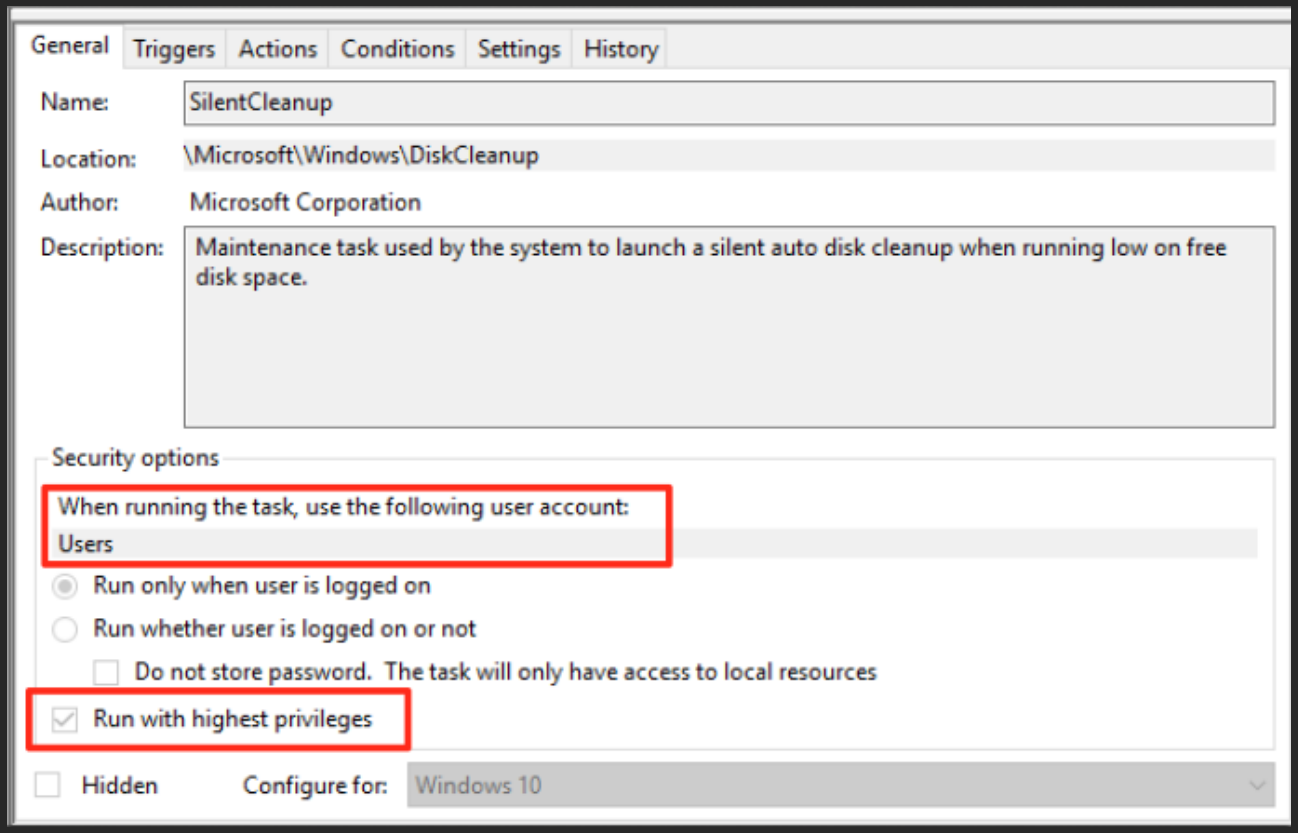
BYPASSING UAC ON WINDOWS 10 USING DISK CLEANUP

July 22, 2016 by enigma0x3

Matt Graeber (@mattifestation) and I recently dug into Windows 10, and discovered a rather interesting method of bypassing User Account Control (if you aren't familiar with UAC you can read more about it here). Currently, there are a couple of public UAC bypass techniques, most of which require a privileged file copy using the IFileOperation COM object or WUSA extraction to take advantage of a DLL hijack. You can dig into some of the public bypasses here (by @hfiref0x). The technique covered in this post differs from the other methods and provides a useful alternative as it does not rely on a privileged file copy or any code injection.

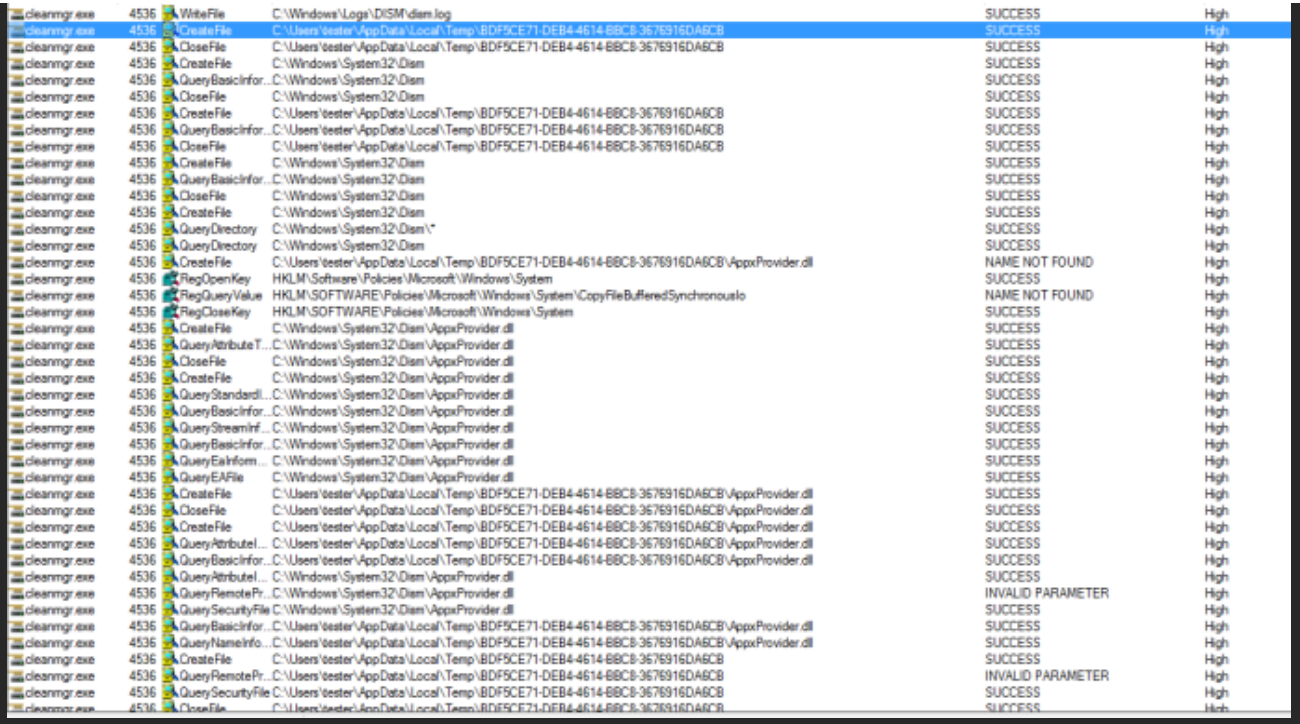
A common technique used to investigate loading behavior on Windows is to use SysInternals Process Monitor to analyze how a process behaves when executed. After investigating some default Scheduled Tasks that exist on Windows 10 and their corresponding actions, we found that a scheduled task named "SilentCleanup" is configured on stock Windows 10 installations to be launchable by unprivileged users but to run with elevated/high integrity privileges. To find this, we

simply went through each task and inspected the security options for “Run with Highest Privileges” to be checked with a non-elevated User Account (such as ‘Users’).



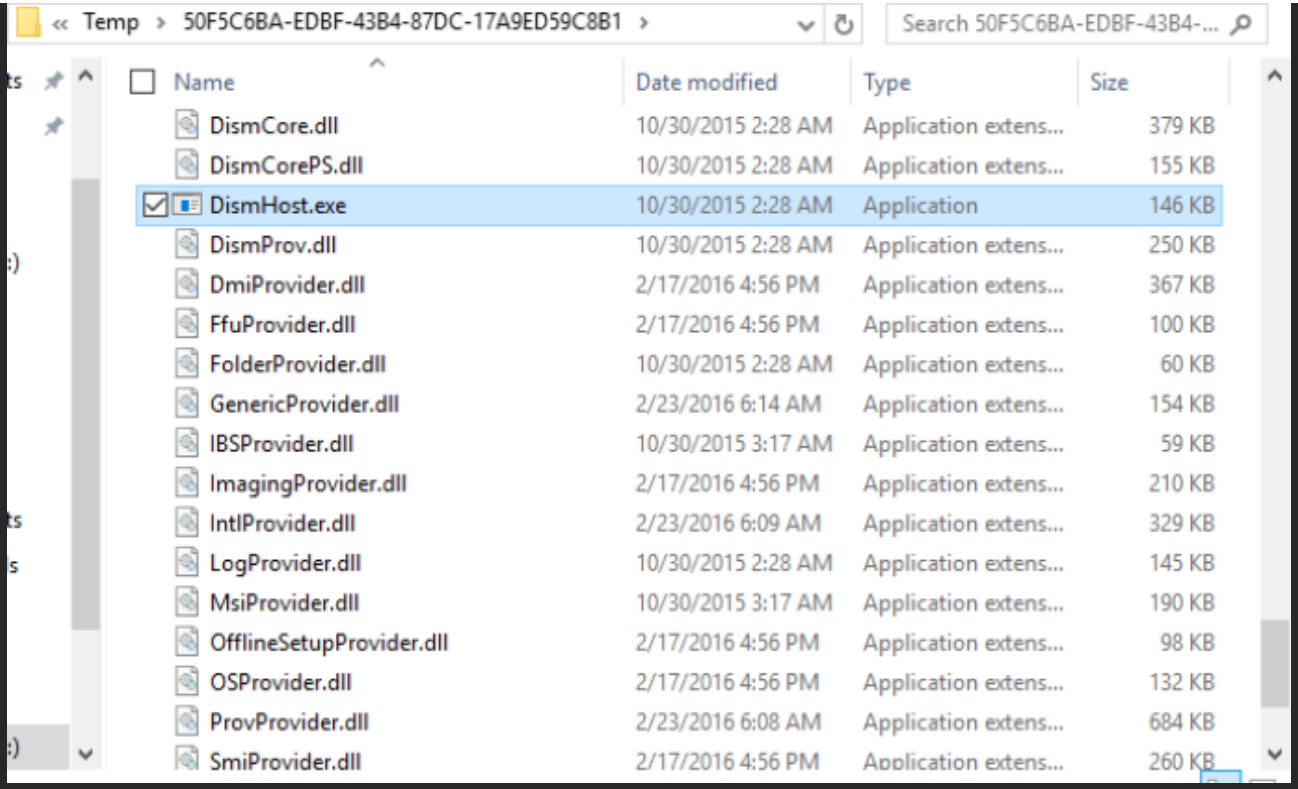
Taking a closer look with procmon, we found that the actual process started by the scheduled task, cleanmgr.exe, auto-elevates due to “execute with highest privileges” being set in the task configuration.

Let’s dive in a bit more. When cleanmgr.exe executes, it creates a new folder with the name of a GUID in “C:\Users\<username>\AppData\Local\Temp”.

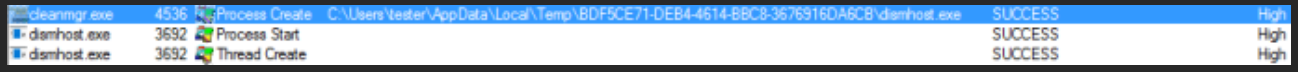


cleanmgr.exe	4536	WriteFile	C:\Windows\Logs\DISM\dism.log	SUCCESS	High
cleanmgr.exe	4536	CreateFile	C:\Users\tester\AppData\Local\Temp\BDF5CE71-DEB4-4614-BBC8-3676916DA6CB	SUCCESS	High
cleanmgr.exe	4536	CloseFile	C:\Users\tester\AppData\Local\Temp\BDF5CE71-DEB4-4614-BBC8-3676916DA6CB	SUCCESS	High
cleanmgr.exe	4536	CreateFile	C:\Windows\System32\Dism	SUCCESS	High
cleanmgr.exe	4536	QueryBasicInfor...	C:\Windows\System32\Dism	SUCCESS	High
cleanmgr.exe	4536	CloseFile	C:\Windows\System32\Dism	SUCCESS	High
cleanmgr.exe	4536	CreateFile	C:\Users\tester\AppData\Local\Temp\BDF5CE71-DEB4-4614-BBC8-3676916DA6CB	SUCCESS	High
cleanmgr.exe	4536	QueryBasicInfor...	C:\Users\tester\AppData\Local\Temp\BDF5CE71-DEB4-4614-BBC8-3676916DA6CB	SUCCESS	High
cleanmgr.exe	4536	CloseFile	C:\Users\tester\AppData\Local\Temp\BDF5CE71-DEB4-4614-BBC8-3676916DA6CB	SUCCESS	High
cleanmgr.exe	4536	CreateFile	C:\Windows\System32\Dism	SUCCESS	High
cleanmgr.exe	4536	QueryBasicInfor...	C:\Windows\System32\Dism	SUCCESS	High
cleanmgr.exe	4536	CloseFile	C:\Windows\System32\Dism	SUCCESS	High
cleanmgr.exe	4536	CreateFile	C:\Windows\System32\Dism	SUCCESS	High
cleanmgr.exe	4536	QueryDirectory	C:\Windows\System32\Dism*	SUCCESS	High
cleanmgr.exe	4536	QueryDirectory	C:\Windows\System32\Dism	SUCCESS	High
cleanmgr.exe	4536	CreateFile	C:\Users\tester\AppData\Local\Temp\BDF5CE71-DEB4-4614-BBC8-3676916DA6CB\AppxProvider.dll	NAME NOT FOUND	High
cleanmgr.exe	4536	RegOpenKey	HKLM\Software\Policies\Microsoft\Windows\System	SUCCESS	High
cleanmgr.exe	4536	RegQueryValue	HKLM\SOFTWARE\Policies\Microsoft\Windows\System\CopyFileBufferedSynchronousIo	NAME NOT FOUND	High
cleanmgr.exe	4536	RegCloseKey	HKLM\SOFTWARE\Policies\Microsoft\Windows\System	SUCCESS	High
cleanmgr.exe	4536	CreateFile	C:\Windows\System32\Dism\AppxProvider.dll	SUCCESS	High
cleanmgr.exe	4536	QueryAttributeT...	C:\Windows\System32\Dism\AppxProvider.dll	SUCCESS	High
cleanmgr.exe	4536	CloseFile	C:\Windows\System32\Dism\AppxProvider.dll	SUCCESS	High
cleanmgr.exe	4536	CreateFile	C:\Windows\System32\Dism\AppxProvider.dll	SUCCESS	High
cleanmgr.exe	4536	QueryStandardI...	C:\Windows\System32\Dism\AppxProvider.dll	SUCCESS	High
cleanmgr.exe	4536	QueryBasicInfor...	C:\Windows\System32\Dism\AppxProvider.dll	SUCCESS	High
cleanmgr.exe	4536	QueryStreamInf...	C:\Windows\System32\Dism\AppxProvider.dll	SUCCESS	High
cleanmgr.exe	4536	QueryBasicInfor...	C:\Windows\System32\Dism\AppxProvider.dll	SUCCESS	High
cleanmgr.exe	4536	QueryEaInform...	C:\Windows\System32\Dism\AppxProvider.dll	SUCCESS	High
cleanmgr.exe	4536	QueryEAFile	C:\Windows\System32\Dism\AppxProvider.dll	SUCCESS	High
cleanmgr.exe	4536	CreateFile	C:\Users\tester\AppData\Local\Temp\BDF5CE71-DEB4-4614-BBC8-3676916DA6CB\AppxProvider.dll	SUCCESS	High
cleanmgr.exe	4536	CloseFile	C:\Users\tester\AppData\Local\Temp\BDF5CE71-DEB4-4614-BBC8-3676916DA6CB\AppxProvider.dll	SUCCESS	High
cleanmgr.exe	4536	CreateFile	C:\Users\tester\AppData\Local\Temp\BDF5CE71-DEB4-4614-BBC8-3676916DA6CB\AppxProvider.dll	SUCCESS	High
cleanmgr.exe	4536	QueryAttributeI...	C:\Users\tester\AppData\Local\Temp\BDF5CE71-DEB4-4614-BBC8-3676916DA6CB\AppxProvider.dll	SUCCESS	High
cleanmgr.exe	4536	QueryBasicInfor...	C:\Users\tester\AppData\Local\Temp\BDF5CE71-DEB4-4614-BBC8-3676916DA6CB\AppxProvider.dll	SUCCESS	High
cleanmgr.exe	4536	QueryAttributeI...	C:\Windows\System32\Dism\AppxProvider.dll	SUCCESS	High
cleanmgr.exe	4536	QueryRemotePr...	C:\Windows\System32\Dism\AppxProvider.dll	INVALID PARAMETER	High
cleanmgr.exe	4536	QuerySecurityFile	C:\Windows\System32\Dism\AppxProvider.dll	SUCCESS	High
cleanmgr.exe	4536	QueryBasicInfor...	C:\Users\tester\AppData\Local\Temp\BDF5CE71-DEB4-4614-BBC8-3676916DA6CB\AppxProvider.dll	SUCCESS	High
cleanmgr.exe	4536	QueryNameInfo...	C:\Users\tester\AppData\Local\Temp\BDF5CE71-DEB4-4614-BBC8-3676916DA6CB\AppxProvider.dll	SUCCESS	High
cleanmgr.exe	4536	CreateFile	C:\Users\tester\AppData\Local\Temp\BDF5CE71-DEB4-4614-BBC8-3676916DA6CB	SUCCESS	High
cleanmgr.exe	4536	QueryRemotePr...	C:\Users\tester\AppData\Local\Temp\BDF5CE71-DEB4-4614-BBC8-3676916DA6CB	INVALID PARAMETER	High
cleanmgr.exe	4536	QuerySecurityFile	C:\Users\tester\AppData\Local\Temp\BDF5CE71-DEB4-4614-BBC8-3676916DA6CB	SUCCESS	High
cleanmgr.exe	4536	CloseFile	C:\Users\tester\AppData\Local\Temp\BDF5CE71-DEB4-4614-BBC8-3676916DA6CB	SUCCESS	High

Once cleanmgr.exe creates the temporary folder, it then copies multiple DLLs along with “dismhost.exe” into the new folder:



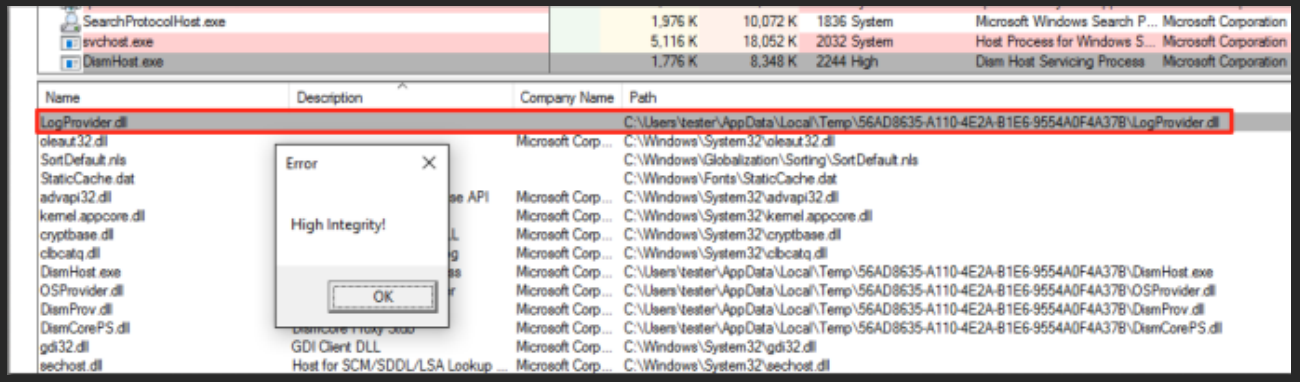
After copying DismHost.exe and its DLLs to “C:\Users\<username>\AppData\Temp\<guid>”, cleanmgr.exe then starts “dismhost.exe” out of the newly created path as a high integrity process:



Since dismhost.exe launches out of “C:\Users\<username>\AppData\Local\Temp\<guid>”, it begins to load DLLs out of the same folder in a certain order:

dismhost.exe	3692	Load Image	C:\Windows\System32\sechost.dll	SUCCESS	High
dismhost.exe	3692	Load Image	C:\Windows\System32\clbcatq.dll	SUCCESS	High
dismhost.exe	3692	Load Image	C:\Users\tester\AppData\Local\Temp\BDF5CE71-DEB4-4614-BBC8-3676916DA6CB\DismCorePS.dll	SUCCESS	High
dismhost.exe	3692	Load Image	C:\Users\tester\AppData\Local\Temp\BDF5CE71-DEB4-4614-BBC8-3676916DA6CB\DismProv.dll	SUCCESS	High
dismhost.exe	3692	Load Image	C:\Users\tester\AppData\Local\Temp\BDF5CE71-DEB4-4614-BBC8-3676916DA6CB\OSProvider.dll	SUCCESS	High
dismhost.exe	3692	Load Image	C:\Windows\System32\advapi32.dll	SUCCESS	High
dismhost.exe	3692	Load Image	C:\Windows\System32\SSShim.dll	SUCCESS	High
dismhost.exe	3692	Load Image	C:\Windows\System32\downlevel\api-ms-win-base-util-l1-1-0.dll	SUCCESS	High

Because the current medium integrity process has write access to the user's %TEMP% directory, it is possible to hijack a DLL loaded by dismhost.exe and obtain code execution in a high integrity process. This is commonly known as a "BypassUAC" attack.
Since this particular situation is a race condition, we have to replace the target DLL before dismhost.exe loads it. We examined the entire process more closely and determined that "LogProvider.dll" is the last DLL loaded by dismhost.exe, giving us the best chance for a hijack opportunity. With this information, we can use a WMI event to monitor for the creation of "C:\Users\<username>\AppData\Local\Temp\<guid>" and then assign that WMI event an action of hijacking "LogProvider.dll" by copying our "malicious" DLL into "C:\Users\<username>\AppData\Local\Temp\<guid>" and naming it "LogProvider.dll". Since this action happens before dismhost.exe loads it, it will load our DLL instead of the intended one.



Once dismhost.exe loads the DLL, it will load as high integrity, allowing us to bypass User Access Control and obtain code execution as a high integrity process.

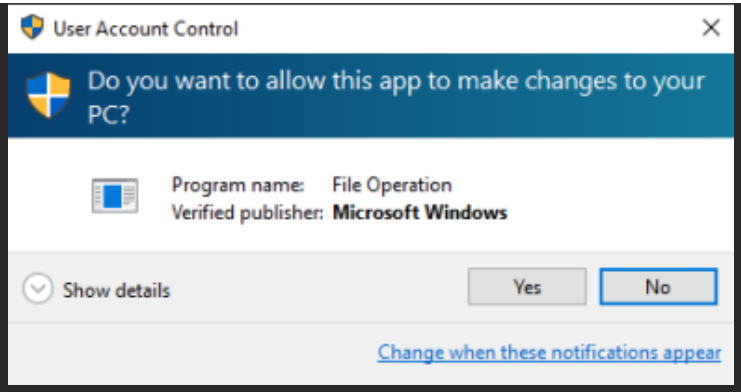
After additional testing, this technique does not apply to standard user accounts as cleanmgr.exe does not extract any files to %TEMP%. When executed as a standard user in low or medium integrity, the task runs as medium integrity and never elevates past that.

Matt Graeber (@mattifestation) wrote an excellent PoC PowerShell script that will register a WMI event to monitor for the creation of the GUID folder by cleanmgr.exe and once detected, it will take the specified DLL and copy it to the GUID folder with the name of “LogProvider.dll”. Once dismhost.exe goes to load “LogProvider.dll”, it will be our malicious DLL instead of the legitimate one, thus bypassing UAC and giving us code execution in High Integrity context. You can find the script here: https://gist.github.com/mattifestation/b4072a066574caccfa07fcf723952d54

To test this, you simply need the PoC script and a DLL with a standard export of dllmain. For testing, you can either create your own DLL or use a simple MessageBox one located here: https://github.com/enigma0x3/MessageBox

This technique differs from the other public techniques by having a few benefits that can be handy:

1. This technique does not require any process injection, meaning the attack won’t get flagged by security solutions that monitor for this type of behavior.
2. There is no privileged file copy required. Most UAC bypasses require some sort of privileged file copy in order to get a malicious DLL into a secure location to setup a DLL hijack. Since the scheduled task copies the required stuff to %TEMP%, no privileged file copy is required.
3. This technique cleans up after itself. After the scheduled task is done (and loads our malicious DLL), the task deletes the GUID folder (and files) that it created in %TEMP%.
4. This technique works with the UAC level being set at its highest setting (“Always Notify”) since the task is set to run with “Highest Privileges”. The majority of the public UAC bypasses rely on the IFileOperation COM object to perform a privileged file copy. IFileOperation honors the “Always Notify” UAC setting and prompts when set, causing the privileged file copy to fail:



This was disclosed to Microsoft Security Response Center (MSRC) on 07/20/2016. As expected, they responded by noting that UAC isn't a security boundary, so this doesn't classify as a security vulnerability, as stated here. While not a vulnerability, it does allow an attacker an alternate method to move to high integrity that differs from previous bypasses and introduces one more location or chokepoint that must be monitored to observe attacker behavior.

This particular technique can be remediated or fixed by disabling the task or removing the requirement for running with highest privileges. Further, if you would like to monitor for this attack, you could utilize methods/signatures to look for new WMI events as it is required to monitor for new folder creation for this attack to succeed. Combining this with App/DLL whitelisting and monitoring for abnormal modules being loaded (e.g. Sysmon event ID 7) would also limit the success of such an attack.

*Update: As always, users should follow best practices and not use an administrative account for daily computer usage.

SHARE THIS:

RELATED

"Fileless" UAC Bypass Using eventvwr.exe and Registry Hijacking
With 27 comments

Bypassing UAC using App Paths
With 3 comments

"Fileless" UAC Bypass using sdclt.exe

Bookmark the permalink.

3 THOUGHTS ON "BYPASSING UAC ON WINDOWS 10 USING DISK CLEANUP"

jen6 says:
July 27, 2016 at 3:52 pm
Hi I'm student in korea.
I think this technique is awesome.
I want to show this article for my friends
Can I translate this article and upload it?

Reply

enigma0x3 says:

July 27, 2016 at 11:04 pm

Sure thing, as long as the original source is cited.

Thanks!

Reply

Pingback: Bypassing UAC using App Paths | enigma0x3

LEAVE A REPLY

Enter your comment here...

Search … Search

ARCHIVES

- January 2018
- November 2017

RECENT POSTS

- Reviving DDE: Using OneNote and Excel for Code Execution
- Lateral Movement Using Outlook's CreateObject Method and DotNetToJScript

RECENT COMMENTS

Soc on Defeating Device Guard: A…

"Fileless… on "Fileless" UAC Byp…

- October 2017
- September 2017
- August 2017
- July 2017
- April 2017
- March 2017
- January 2017
- November 2016
- August 2016
- July 2016
- May 2016
- March 2016
- February 2016
- January 2016
- October 2015
- August 2015
- April 2015
- March 2015
- January 2015
- October 2014
- July 2014
- June 2014
- March 2014
- January 2014

Blog at WordPress.com.

- A Look at CVE-2017-8715: Bypassing CVE-2017-0218 using PowerShell Module Manifests
- UMCI Bypass Using PSWorkFlowUtility: CVE-2017-0215
- Lateral Movement using Excel.Application and DCOM

CATEGORIES

- Uncategorized

"Fileless… on Bypassing UAC usin App P…

Windows 10 UAC Looph… on Bypa UAC using App P…

NexusLogger: A New C… on "Filel UAC Byp…

META

- Register
- Log in
- Entries RSS
- Comments RSS
- WordPress.com