RED TEAMER AND SECURITY ADDICT

ENIGMA0X3

« BYPASSING UAC USING APP PATHS

DEFEATING DEVICE GUARD: A LOOK INTO CVE-2017-0007 »

“FILELESS” UAC BYPASS USING SDCLT.EXE

March 17, 2017 by enigma0x3

Recently, I published a post on using App Paths with sdclt.exe to bypass UAC. You may remember that the App Path bypass required a file on disk. Since sdclt.exe is out there, I figured I would publish another bypass using that binary, only this one is fileless. I mentioned it in my previous post, but the Vault7 leak confirms that bypassing UAC is operationally interesting, even to nation states, as several UAC bypasses/notes were detailed in the dump. As far as public bypasses go, definitely check out the UACME project by @hfiref0x, which has a nice collection of public techniques.

In newer versions of Windows, Microsoft has shown that they are taking the bypasses seriously. This has motivated me to spend a little more time on UAC and the different methods around it.

As some of you may know, there are some Microsoft signed binaries that auto-elevate due to their manifest. You can read more about these binaries and their manifests here. While searching for more of these auto-elevating binaries by using the SysInternals tool “sigcheck“, I came across “sdclt.exe” and verified that it auto-elevates due to its manifest:

**Note: This only works on Windows 10. The manifest for sdclt.exe in Windows 7 has the requestedExecutionLevel set to “AsInvoker”, preventing auto-elevation when started from medium integrity.*

As I mentioned in my last post, a common technique used to investigate loading behavior on Windows is to use SysInternals Process Monitor to analyze how a process behaves when executed. I often work some basic binary analysis into my investigative process in order to see what other opportunities exist.

One of the first things I tend to do when analyzing an auto-elevate binary is to look for any potential command line arguments. I use IDA for this, but you can use your preferred tool. When peering into sdclt.exe, I noticed a few arguments that stood out due to interesting keywords:

These were interesting as sdclt.exe is set to auto-elevate in its manifest anyway. Looking at sdclt.exe in IDA, it checks if the argument matches “/kickoffelev”. If it does, it sets the full path for “sdclt.exe”, adds “/KickOffJob” as a parameter and then calls SxShellExecuteWithElevate.

Following that path, SxShellExecuteWithElevate starts “%systemroot%\system32\sdclt.exe /kickoffjob” with the “Runas” verb. This is essentially programmatically executing the “RunAsAdministrator” option when you right-click a binary.

The next step is to run “sdclt.exe /Kickoffelev” with procmon running. After going through the output, we see the trusty “shell\<verb>\command” registry search path in the HKEY_CURRENT_USER hive.

The next step was to add those keys and see if our binary and parameters of choice would execute. Unfortunately, nothing executed after adding the keys and starting “sdclt.exe /kickoffelev”. Looking back in procmon, our keys are queried, but sdclt.exe is actually looking for an additional value within the “command” key: “IsolatedCommand”.

We can then add our payload and parameters in a string (REG_SZ) value within the “Command” key called “IsolatedCommand”:

This is the same bug (minus the IsolatedCommand portion) that was used in the eventvwr.exe “fileless” UAC bypass. You can read about the eventvwr.exe bypass and the specific registry keys used here. Notice that instead of “shell\open\command”, we now see “shell\runas\command”. This is because sdclt.exe was invoked (again) using the “RunAs” verb via SxShellExecuteWithElevate.

After adding our payload as the “IsolatedCommand” value, running “sdclt.exe /KickOffElev” will execute our payload (and any parameters) in a high-integrity context:

To demonstrate this technique, you can find a script here: https://github.com/enigma0x3/Misc-PowerShell-Stuff/blob/master/Invoke-SDCLTBypass.ps1

The script takes a full path to your payload and any parameters. “C:\Windows\System32\cmd.exe /c notepad.exe” is a good one to validate. It will automatically add the keys, start “sdclt.exe /kickoffelev” and then cleanup.

This particular technique can be remediated or fixed by setting the UAC level to “Always Notify” or by removing the current user from the Local Administrators group. Further, if you would like to monitor for this attack, you could utilize methods/signatures to look for and alert on new registry entries in **HKCU:\Software\Classes\exefile\shell\runas\command\isolatedCommand**

Cheers,
Matt

SHARE THIS:

Twitter Facebook

Loading...

Bookmark the permalink.

LEAVE A COMMENT

Search … Search

ARCHIVES

- October 2023
- January 2020
- December 2019
- August 2019
- July 2019
- March 2019
- January 2019
- October 2018
- June 2018
- January 2018
- November 2017
- 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

RECENT POSTS

- CVE-2023-4632: Local Privilege Escalation in Lenovo System Updater
- Avira VPN Local Privilege Escalation via Insecure Update Location
- CVE-2019-19248: Local Privilege Escalation in EA’s Origin Client
- Avira Optimizer Local Privilege Escalation
- CVE-2019-13382: Local Privilege Escalation in SnagIt

CATEGORIES

- Uncategorized

RECENT COMMENTS

Ron on CVE-2019-13382

enigma0x3 on CVE-2019
Privileg…

Ron on CVE-2019-13382

Soc on Defeating Device

“Fileless… on “Fileless” U

META

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

Blog at WordPress.com.