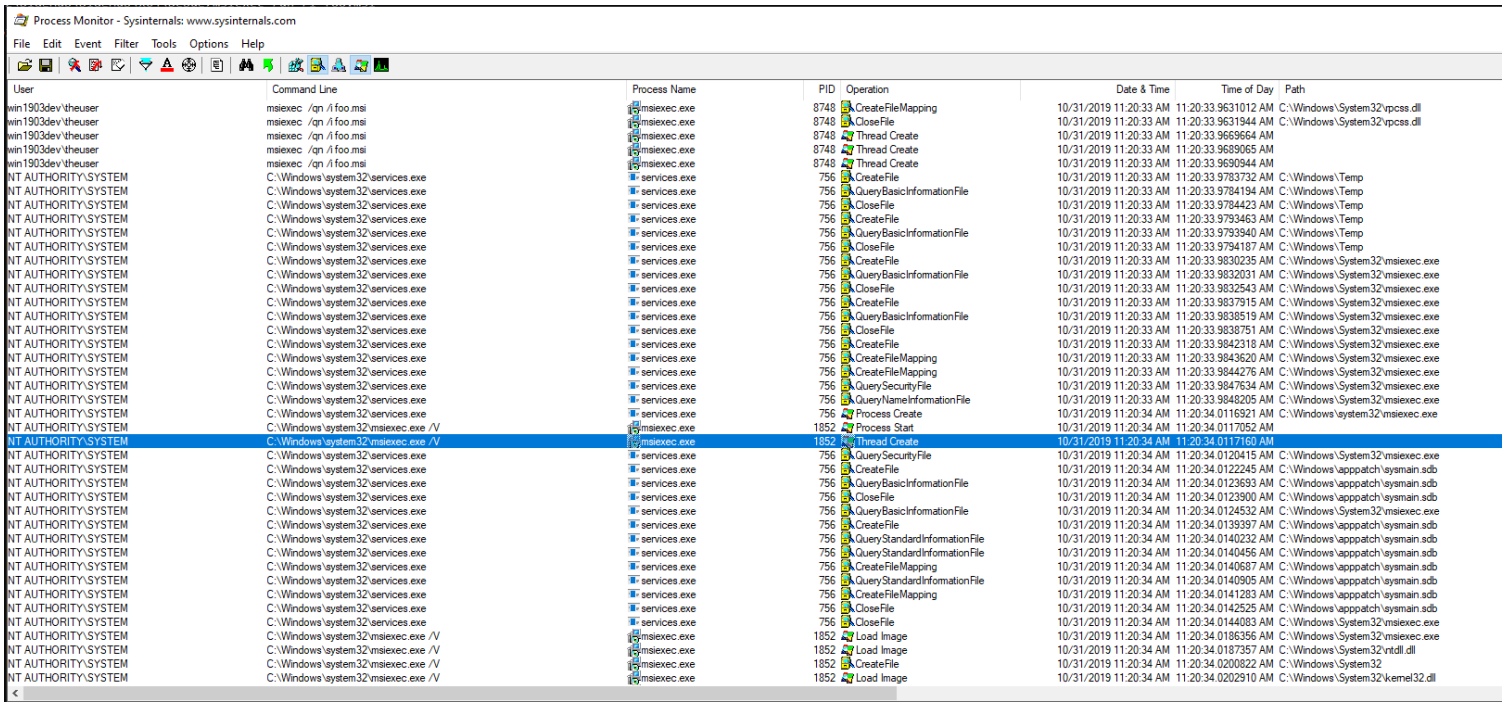
MSI - Microsoft Windows Installer Elevation of Privilege

Summary: The Microsoft Windows Installer permits under some circumstances to a "standard user" an arbitrary permissions and content overwrite with SYSTEM privileges.
(Microsoft Windows Installer: https://docs.microsoft.com/en-us/windows/win32/msi/windows-installer-portal)

Products affected: Windows 10 Enterprise (1903) with latest security update (2019 November patch) and probably also other versions (not tested). Windows 10 Enterprise - INSIDER PREVIEW (Fast ring) 10.0.19033 - Build 19033

Description:
It is possible and allowed by "Windows Installer service" to install a MSI package as "standard user".
I have noticed, in my research, that when a "standard user" installs a MSI package , the "Windows Installer service" triggers some operations with SYSTEM privileges. (see image below from Procmon)



User	Command Line	Process Name	PID	Operation	Date & Time	Time of Day	Path
win1903dev\theuser	msiexec /qn /i foo.msi	msiexec.exe	8748	CreateFileMapping	10/31/2019 11:20:33 AM	11:20:33.9631012 AM	C:\Windows\System32\vpcss.dll
win1903dev\theuser	msiexec /qn /i foo.msi	msiexec.exe	8748	CloseFile	10/31/2019 11:20:33 AM	11:20:33.9631944 AM	C:\Windows\System32\vpcss.dll
win1903dev\theuser	msiexec /qn /i foo.msi	msiexec.exe	8748	Thread Create	10/31/2019 11:20:33 AM	11:20:33.9669664 AM	
win1903dev\theuser	msiexec /qn /i foo.msi	msiexec.exe	8748	Thread Create	10/31/2019 11:20:33 AM	11:20:33.9689065 AM	
win1903dev\theuser	msiexec /qn /i foo.msi	msiexec.exe	8748	Thread Create	10/31/2019 11:20:33 AM	11:20:33.9690944 AM	
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	CreateFile	10/31/2019 11:20:33 AM	11:20:33.9783732 AM	C:\Windows\Temp
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	QueryBasicInformationFile	10/31/2019 11:20:33 AM	11:20:33.9784194 AM	C:\Windows\Temp
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	CloseFile	10/31/2019 11:20:33 AM	11:20:33.9784423 AM	C:\Windows\Temp
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	CreateFile	10/31/2019 11:20:33 AM	11:20:33.9793463 AM	C:\Windows\Temp
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	QueryBasicInformationFile	10/31/2019 11:20:33 AM	11:20:33.9793940 AM	C:\Windows\Temp
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	CloseFile	10/31/2019 11:20:33 AM	11:20:33.9794187 AM	C:\Windows\Temp
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	CreateFile	10/31/2019 11:20:33 AM	11:20:33.9830235 AM	C:\Windows\System32\msiexec.exe
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	QueryBasicInformationFile	10/31/2019 11:20:33 AM	11:20:33.9832031 AM	C:\Windows\System32\msiexec.exe
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	CloseFile	10/31/2019 11:20:33 AM	11:20:33.9832543 AM	C:\Windows\System32\msiexec.exe
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	CreateFile	10/31/2019 11:20:33 AM	11:20:33.9837915 AM	C:\Windows\System32\msiexec.exe
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	QueryBasicInformationFile	10/31/2019 11:20:33 AM	11:20:33.9838519 AM	C:\Windows\System32\msiexec.exe
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	CloseFile	10/31/2019 11:20:33 AM	11:20:33.9838751 AM	C:\Windows\System32\msiexec.exe
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	CreateFile	10/31/2019 11:20:33 AM	11:20:33.9842318 AM	C:\Windows\System32\msiexec.exe
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	CreateFileMapping	10/31/2019 11:20:33 AM	11:20:33.9843620 AM	C:\Windows\System32\msiexec.exe
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	CreateFileMapping	10/31/2019 11:20:33 AM	11:20:33.9844276 AM	C:\Windows\System32\msiexec.exe
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	QuerySecurityFile	10/31/2019 11:20:33 AM	11:20:33.9847634 AM	C:\Windows\System32\msiexec.exe
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	QueryNameInformationFile	10/31/2019 11:20:33 AM	11:20:33.9848205 AM	C:\Windows\System32\msiexec.exe
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	Process Create	10/31/2019 11:20:34 AM	11:20:34.0116921 AM	C:\Windows\system32\msiexec.exe
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	1852	Process Start	10/31/2019 11:20:34 AM	11:20:34.0117052 AM	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	1852	Thread Create	10/31/2019 11:20:34 AM	11:20:34.0117160 AM	
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	QuerySecurityFile	10/31/2019 11:20:34 AM	11:20:34.0120415 AM	C:\Windows\System32\msiexec.exe
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	CreateFile	10/31/2019 11:20:34 AM	11:20:34.0122245 AM	C:\Windows\apppatch\sysmain.sdb
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	QueryBasicInformationFile	10/31/2019 11:20:34 AM	11:20:34.0123693 AM	C:\Windows\apppatch\sysmain.sdb
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	CloseFile	10/31/2019 11:20:34 AM	11:20:34.0123900 AM	C:\Windows\apppatch\sysmain.sdb
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	QueryBasicInformationFile	10/31/2019 11:20:34 AM	11:20:34.0124532 AM	C:\Windows\System32\msiexec.exe
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	CreateFile	10/31/2019 11:20:34 AM	11:20:34.0139397 AM	C:\Windows\apppatch\sysmain.sdb
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	QueryStandardInformationFile	10/31/2019 11:20:34 AM	11:20:34.0140232 AM	C:\Windows\apppatch\sysmain.sdb
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	QueryStandardInformationFile	10/31/2019 11:20:34 AM	11:20:34.0140456 AM	C:\Windows\apppatch\sysmain.sdb
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	CreateFileMapping	10/31/2019 11:20:34 AM	11:20:34.0140687 AM	C:\Windows\apppatch\sysmain.sdb
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	QueryStandardInformationFile	10/31/2019 11:20:34 AM	11:20:34.0140905 AM	C:\Windows\apppatch\sysmain.sdb
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	CreateFileMapping	10/31/2019 11:20:34 AM	11:20:34.0141283 AM	C:\Windows\apppatch\sysmain.sdb
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	CloseFile	10/31/2019 11:20:34 AM	11:20:34.0142525 AM	C:\Windows\apppatch\sysmain.sdb
NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe	services.exe	756	CloseFile	10/31/2019 11:20:34 AM	11:20:34.0144083 AM	C:\Windows\System32\msiexec.exe
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	1852	Load Image	10/31/2019 11:20:34 AM	11:20:34.0186356 AM	C:\Windows\System32\msiexec.exe
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	1852	Load Image	10/31/2019 11:20:34 AM	11:20:34.0187357 AM	C:\Windows\System32\ntdll.dll
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	1852	CreateFile	10/31/2019 11:20:34 AM	11:20:34.0200822 AM	C:\Windows\System32
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	1852	Load Image	10/31/2019 11:20:34 AM	11:20:34.0202910 AM	C:\Windows\System32\kernel32.dll

Going forward with my research I found that, after a package is installed on the system, a "standard user" could force a repair of the product with "/f" command line parameter.
In our case, for example, command "msiexec /qn /fa foo.msi" triggers the repair operation .
With this command I've seen a couple of interesting operations came out that caught my attention.
As SYSTEM, the "Windows Installer service" try to set the permissions of the package files that are going to be reinstalled.
After that, it read and writes the content of the package files (stored within msi package).
See image below from Procmon.

NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	CreateFile	C:\Users\theuser\AppData\Local\fakemsi	SUCCESS	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	QueryNetworkOpenInformationFile	C:\Users\theuser\AppData\Local\fakemsi	SUCCESS	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	CloseFile	C:\Users\theuser\AppData\Local\fakemsi	SUCCESS	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	CreateFile	C:\Users\theuser\AppData\Local\fakemsi\foo.txt	NAME NOT FOUND	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	CreateFile	C:\Users\theuser\AppData\Local\fakemsi	SUCCESS	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	QueryBasicInformationFile	C:\Users\theuser\AppData\Local\fakemsi	SUCCESS	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	CloseFile	C:\Users\theuser\AppData\Local\fakemsi	SUCCESS	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	CreateFile	C:\Users\theuser\AppData\Local\fakemsi\foo.txt	PRIVILEGE NOT HELD	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	CreateFile	C:\Users\theuser\AppData\Local\fakemsi\foo.txt	SUCCESS	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	CloseFile	C:\Users\theuser\AppData\Local\fakemsi\foo.txt	SUCCESS	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	CreateFile	C:\Users\theuser\AppData\Local\fakemsi\foo.txt	SUCCESS	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	CreateFile	C:\Users\theuser\AppData\Local\fakemsi\foo.txt	SUCCESS	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	SetBasicInformationFile	C:\Users\theuser\AppData\Local\fakemsi\foo.txt	SUCCESS	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	CloseFile	C:\Users\theuser\AppData\Local\fakemsi\foo.txt	SUCCESS	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	SetSecurityFile	C:\Users\theuser\AppData\Local\fakemsi\foo.txt	SUCCESS	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	SetEndOfFileInformationFile	C:\Users\theuser\AppData\Local\fakemsi\foo.txt	SUCCESS	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	SetAllocationInformationFile	C:\Users\theuser\AppData\Local\fakemsi\foo.txt	SUCCESS	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	WriteFile	C:\Users\theuser\AppData\Local\fakemsi\foo.txt	SUCCESS	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	ReadFile	C:\Users\theuser\AppData\Local\fakemsi\foo.txt	SUCCESS	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	SetBasicInformationFile	C:\Users\theuser\AppData\Local\fakemsi\foo.txt	SUCCESS	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	CloseFile	C:\Users\theuser\AppData\Local\fakemsi\foo.txt	SUCCESS	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	CreateFile	C:\Windows\Installer	SUCCESS	
NT AUTHORITY\SYSTEM	C:\Windows\system32\msiexec.exe /V	msiexec.exe	6392	QueryBasicInformationFile	C:\Windows\Installer	SUCCESS	

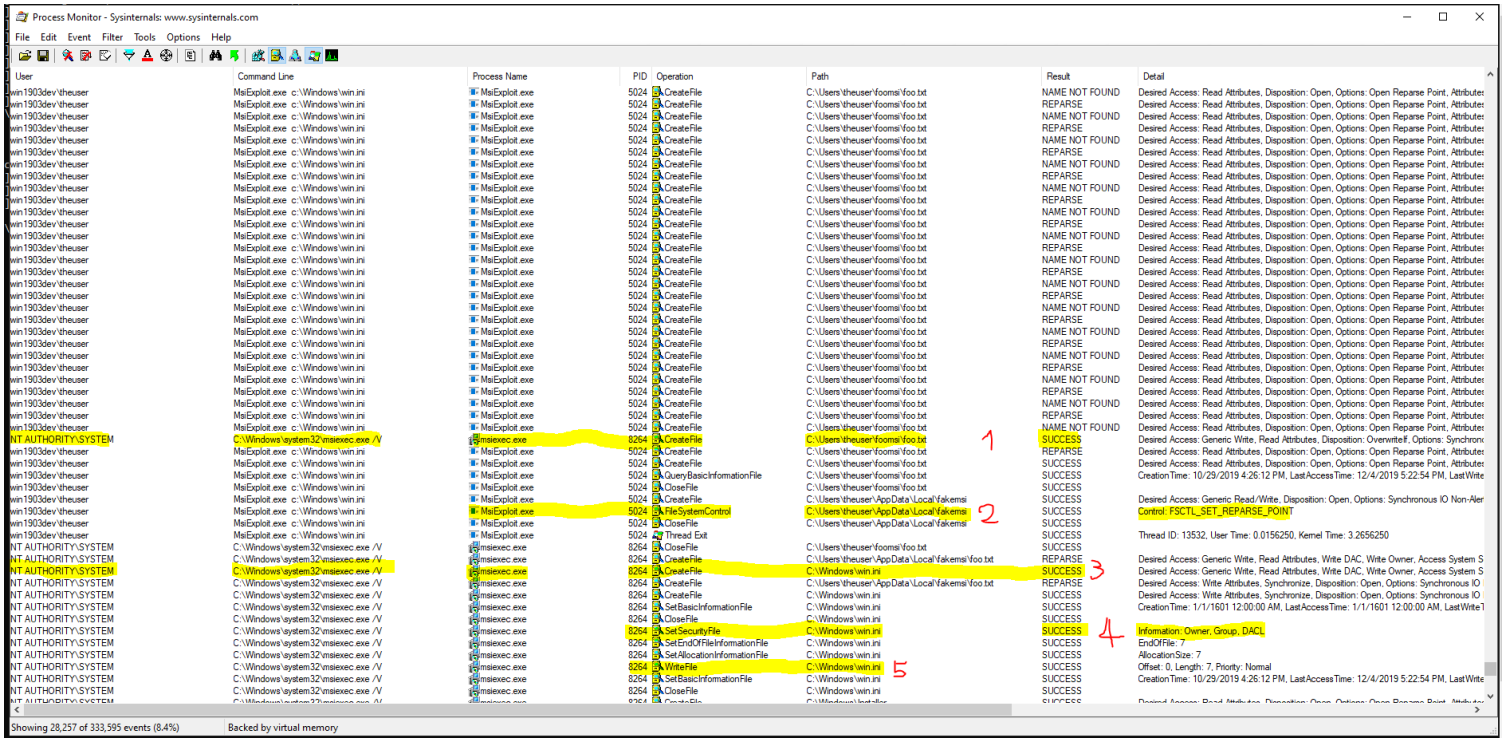
As we can see, the first time "Windows Installer service" tries to open one the files impersonating the "standard user" but as a result it gets a "PRIVILEGE NOT HELD", then, after that, it closes the file and reopens it as SYSTEM without impersonating! Afterward it continues to set the permissions of the file as SYSTEM and writes its content.

This is clearly a point of possible exploitation! In order to obtain the desired a result, a "race condition" has to be successfully exploited. The "race condition" is going to occur between the moment when "Windows Installer service" closes the file as "standard user" and reopens it as SYSTEM just before it writes the DACLs and the content.
Now that the logical workflow seems a little bit clear (I hope), I'll try to describe the steps executed by the exploit.

First of all: I've built a MSI package (foo.msi) that can be installed also as a "standard user"; this means that MSI service will install all files in C:\Users\[USER]\AppData\Local . In our particular case, I've built this MSI package which installs only the file "foo.txt" into C:\Users\[USER]\AppData\Local\fakemsi\ directory (C# VS2017 project will also be sent with this report).
In order to make the MSI package installable by "standard user", I've run against it the following command:
"C:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x86\MsiInfo.exe"
"C:\temp2\Setup1\Setup1\Debug\foo.msi" -w 10
(MsiInfo belongs to WDK, so watch out for your version
REF : https://docs.microsoft.com/en-us/windows/win32/msi/msiinfo-exe)

These are the steps that the exploit performs:

- Before exploits removes temporary old directories used for junctions.
- Create an empty directory C:\Users\[USER]\foomsi
- Create empty directory C:\Users\[USER]\AppData\Local\fakemsi ; this is the directory that MSI package will create in order to save "foo.txt" file
- Create a junctions from C:\Users\[USER]\AppData\Local\fakemsi to C:\Users\[USER]\foomsi (MSI service will not delete it) so later we can abuse of this junction to gain privileges. That's a trickie part.
- Create a sort of symbolic link in "\RPC Control" object namespace. This link will be named "foo.txt" and it points to the file we want to own (c:\windows\win.ini in this case)
- Remove the msi package (even if it doesn't exist): command "msiexec /qn /i foo.msi"
- Install the msi package: command "msiexec /qn /i foo.msi"
- Start thread to win the race condition and just after milliseconds triggers Window Installer service with command "msiexec /qn /fa foo.msi" in order to exploit "setSecurity" operation and win the race.
- Thread starts watching for C:\Users\[USER]\AppData\Local\fakemsi\foo.txt existance
- As soon as C:\Users\[USER]\AppData\Local\fakemsi\foo.txt has been renamed by MSI service, and it does not exist anymore, the exploit will set a reparse point (a junction) from C:\Users\[USER]\AppData\Local\fakemsi\ to "\RPC Control"
- At this point, MSI service creates again C:\Users\[USER]\AppData\Local\fakemsi\foo.txt , and it will REPARSE to "\RPC Control" where there is "foo.txt" file that is a link pointing to the target file, in this way it's going to exploit setSecurity operation (we'll gain file content overwrite too). That's the core of the race condition (and of the exploit) that we'll try to win; matters of milliseconds
- Below a screenshot (from procmon) of a successful exploitation. I have marked all the importation operations that can be observed:
  1) MSI service executes "CreateFile" successfully impersonating "normal" user.
  2) Exploit sets mountpoint from C:\Users\[USER]\AppData\Local\fakemsi\ to "\RPC Control"
  3) MSI service does a REPARSE to target file (C:\windows\win.ini in this case) and executes "CreateFile" successfully as SYSTEM user
  4) MSI service sets security DACL as SYSTEM ; it gives FULL CONTROL to the "normal" user to the target file.
  5) MSI service writes content read from "foo.txt" file inside foo.msi package.



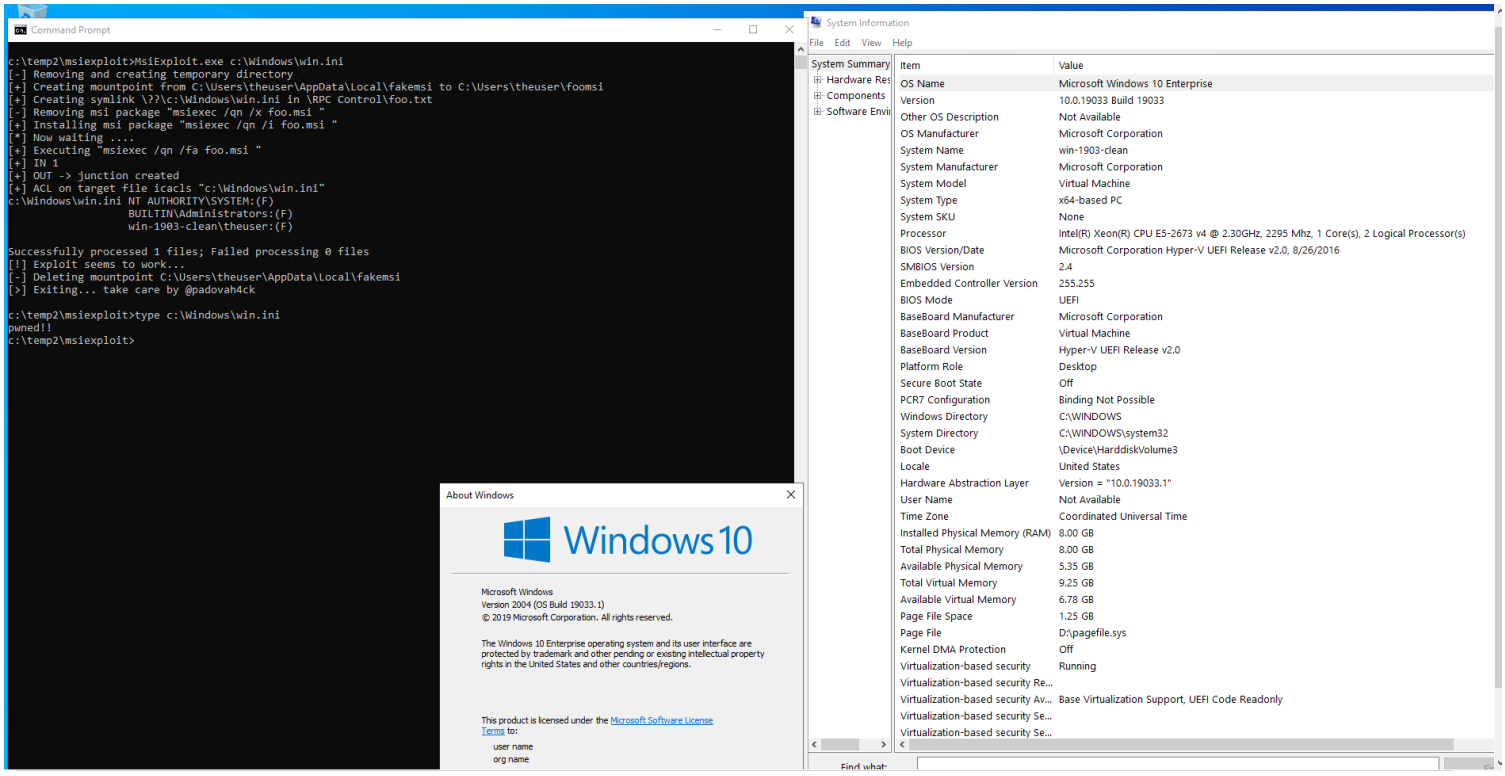
This exploit can overwrite DACL of files that are fully owned by SYSTEM.
Most of the times, in my tests, the exploit works at first time… If it doesn't work at the first time, try to re-run again, it should be work.

I have provided a full working PoC:
- Pay attention, all files in "bin_MsiExploit" need to stay on the same directory. (please read readme.txt inside zipped file)

All the source code needed:
- exploit source code - VS 2017 C++ (src_MsiExploit directory)
- msi package source code - VS 2017 C# already provided in previous emails

Screens shots:

Successfully exploitation on Windows 10 Enterprise Insider Preview (Fast ring) 10.0.19033



Conclusions:
I think that bug stands behind an incorrect impersonated operation when it comes to write the DACLs.

Best Regards,
Christian Danieli (@padovah4ck)