UAC 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
- COM Handlers
- Environment Variable
- 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...

Ain't nobody got time to triage that!
How does UAC work

Administrators

Split token
Auto-Elevating Binaries

```sh
|-> 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)
- UAC bypass archetype by Leo Davidson
  - Win 7, 8

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 <- \(ツ\)/ \(ツ\)/ \(ツ\)/

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

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

- Essentially identifies a process from its Process Environment Block (PEB).

- Semi-Documented Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProcessHeap</td>
<td>0000021e178c8000</td>
</tr>
<tr>
<td>ProcessParameters</td>
<td>0000021e178c16f0</td>
</tr>
<tr>
<td>CurrentDirectory</td>
<td>C:\Windows\System32</td>
</tr>
<tr>
<td>InProcFile</td>
<td>C:\Windows\System32\notepad.exe</td>
</tr>
<tr>
<td>CommandLine</td>
<td>C:\Windows\System32\notepad.exe</td>
</tr>
<tr>
<td>Environment</td>
<td>0000021e178c80fc</td>
</tr>
</tbody>
</table>

=> A process has access to its own memory though, mmmmm...?
- 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

We don't need this because PowerShell is a trusted executable ಥ_ಥ! (ツ)_/\
In-Memory IFileOperation

- Stephen Toub December 2007 MSDN magazine


- Slightly modified his library to REQUIREELEVATION & SILENT
PS C:\> Invoke-IFileOperation
PS C:\> $IFileOperation | Get-Member

TypeName: FileOperation.FileOperation

<table>
<thead>
<tr>
<th>Name</th>
<th>MemberType</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CopyItem</td>
<td>Method</td>
<td>void CopyItem(string source, string destination, string newName)</td>
</tr>
<tr>
<td>DeleteItem</td>
<td>Method</td>
<td>void DeleteItem(string source)</td>
</tr>
<tr>
<td>Dispose</td>
<td>Method</td>
<td>void Dispose(), void IDisposable.Dispose()</td>
</tr>
<tr>
<td>Equals</td>
<td>Method</td>
<td>bool Equals(System.Object obj)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Method</td>
<td>int GetHashCode()</td>
</tr>
<tr>
<td>GetType</td>
<td>Method</td>
<td>type GetType()</td>
</tr>
<tr>
<td>MoveItem</td>
<td>Method</td>
<td>void MoveItem(string source, string destination, string newName)</td>
</tr>
<tr>
<td>NewItem</td>
<td>Method</td>
<td>void NewItem(string folderName, string name, System.IO.FileAttributes attrs)</td>
</tr>
<tr>
<td>PerformOperations</td>
<td>Method</td>
<td>void PerformOperations()</td>
</tr>
<tr>
<td>RenameItem</td>
<td>Method</td>
<td>void RenameItem(string source, string newName)</td>
</tr>
<tr>
<td>ToString</td>
<td>Method</td>
<td>string ToString()</td>
</tr>
</tbody>
</table>
Invoke -IFileOperation Demo

```csharp
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\oobe\setupsqm.exe

  -> Win 7,8,8.1

- Or C:\Windows\System32\odbcad32.exe

  -> Win 7
Congratuations, 0day!

- "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 secruntime.dll
Global resource cache => C:\Windows\WinSxS

- Introduced as a solution to the so-called dll hell problem
- Completely breaks UAC dll hijacking assumptions, truly 0day everywhere!
- Also a good place to bypass poor application lockdown ;)

Windows Side-By-Side Assembly
sysprep Case-Study

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

```xml
<dependency>
  <dependentAssembly>
    <assemblyIdentity language="*" name="Microsoft.Windows.Common-Controls" processorArchitecture="amd64" publicKeyToken="6595b64144ccf1df" type="win32"
    version="6.0.0.0"
  />
</dependentAssembly>
</dependency>
```
- 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!
  |  --> `$IFileOperation.MoveItem("C:\Some\Path\sysprep.exe.local","C:\Windows\System32\sysprep","sysprep.exe.local")`
  |  --> `$IFileOperation.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

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?
Surely COM Objects Are fine!

- Look for low hanging fruit, lazy mode!

```powershell
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

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

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

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
Let's have a look at two case studies!

- `C:\Windows\System32\CompMgmtLauncher.exe`
  - `%ProgramData%` -> `Computer Management.lnk`
  - `Win7,8,8.1,10,10RS1`

- `schtasks /Run /TN \Microsoft\Windows\DiskCleanup\SilentCleanup /I`
  - `%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-clean-up/

- Elevated COM

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

-James Forshaw
+ UACME project – @hfiref0x
   --> https://github.com/hfiref0x/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
+ 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
+ Inside Windows 7 User Account Control – TechNet

......And many many more!......
Questions?

Contact:
+ ruben.boonen@mwrinfosecurity.com
+ @FuzzySec