



Having FUN* with COM

James Forshaw (@tiraniddo)

Infiltrate 2019

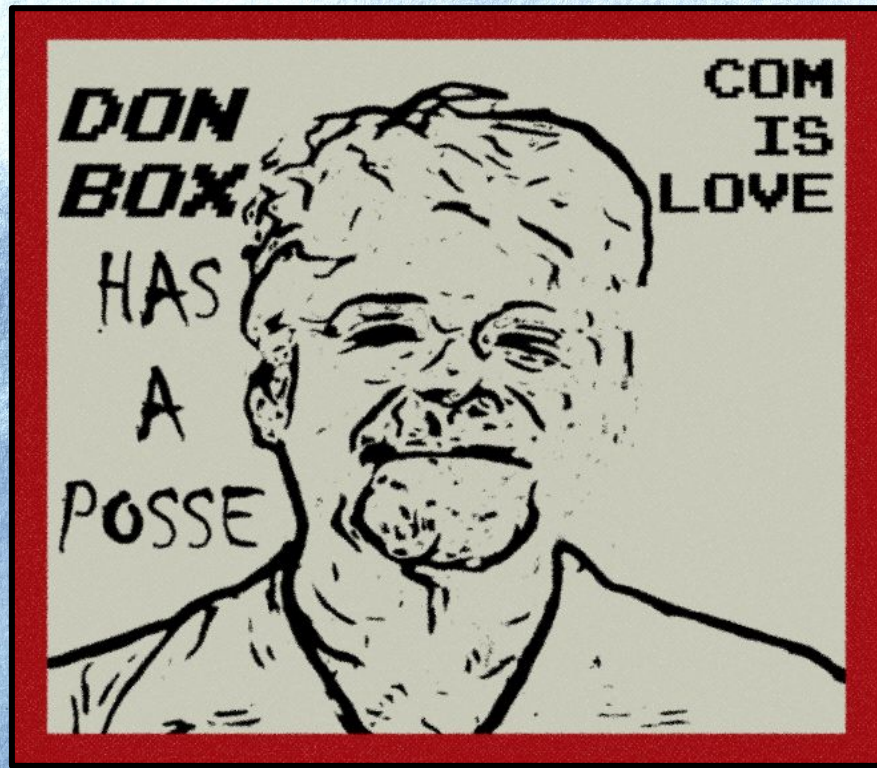
* FUN assumed but not guaranteed.

Background



Agenda

- New and Improved Tooling
- Escaping Sandboxes
- Cross-Session Attacks
- Abusing COM Marshaling
- Mem Read → Code Execution
- Persistence Tricks



Windows PowerShell

PS C:\> Get-ComDatabase -SetCurrent

PS C:\> Get-ComClass -InteractiveUser

Name	CLSID
BrowserBroker Class	00000000-0000-0000-0000-00000000...
User Notification	00000000-0000-0000-0000-00000000...
PhotoAcqHWEventHandler	00f2b433-44e4-4d88-b2b0-2698a0a...
RoamDictionary Class	00000000-0000-0000-0000-00000000...
0207C0AD-563B-4919-A967-E07...	0207c0ad-563b-4919-a967-e0782ff...
SimpleInputItem Class	03000000-0000-0000-0000-00000000...
Shared Reco Custom Marshall...	03000000-0000-0000-0000-00000000...
DevicesFlow	03000000-0000-0000-0000-00000000...
CDPComAccountProvider	049d0000-0000-0000-0000-00000000...
UIHost Class	05000000-0000-0000-0000-00000000...
User Account Control	06c79200-0000-0000-0000-00000000...
Shield Provider	07200000-0000-0000-0000-00000000...
Retail Demo User COM Agent	0886dae5-13ba-49d6-a6ef-d0922e5...
Proximity Sharing	08fc06e4-c6b5-40be-97b0-b80f943...

OleView.NET

X

PowerShell

Parse registry information into a database and set as current database.

```
PS> Get-ComDatabase -SetCurrent
```

Get all COM classes from the current database.

```
PS> Get-ComClass
```

Get all COM classes which have a registered local server.

```
PS> Get-ComClass -ServerType LocalServer32
```

Enumerate all accessible interfaces for a class.

```
PS> $intfs = Get-ComClassInterface $cls
```

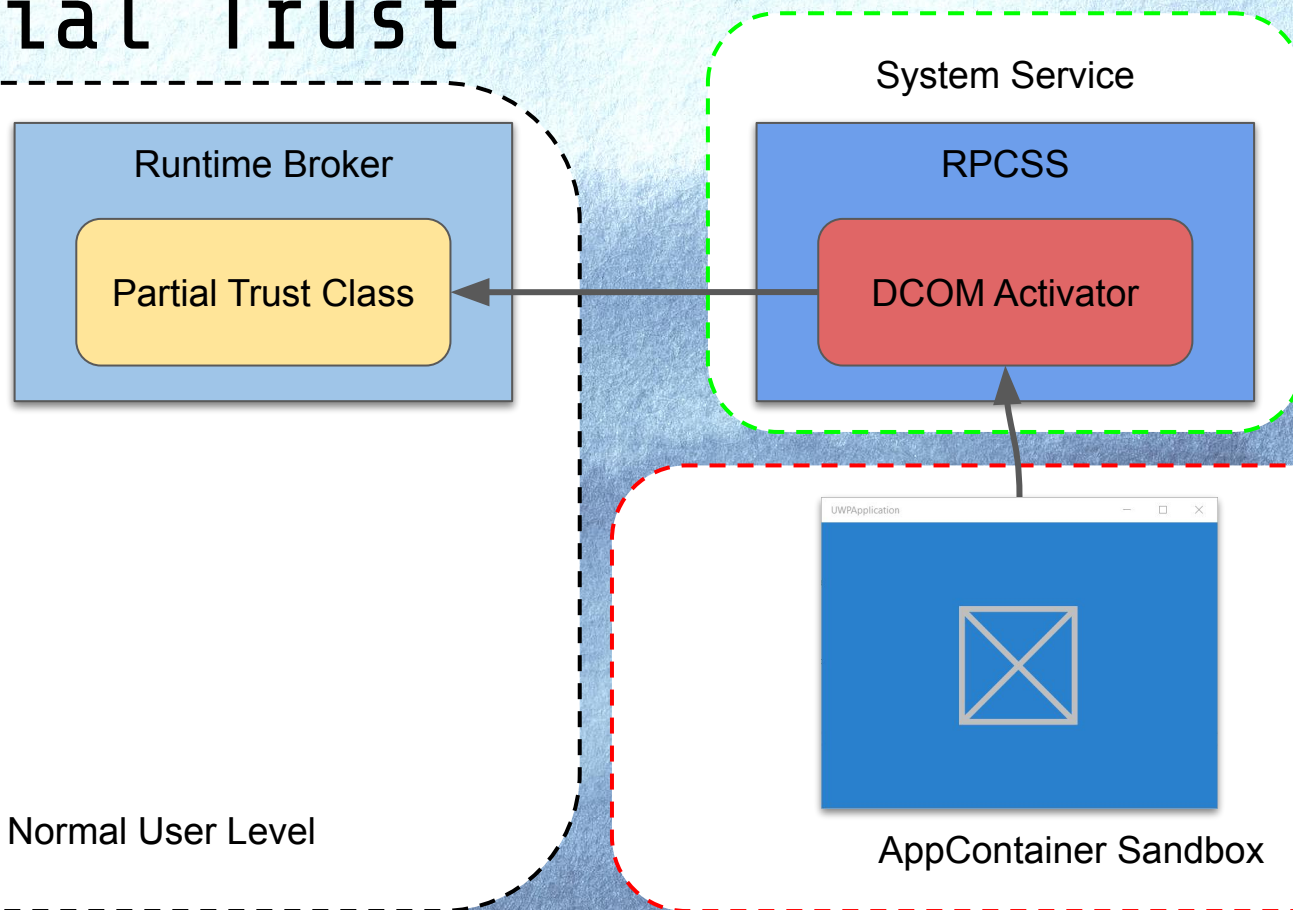



DEMO
TIME

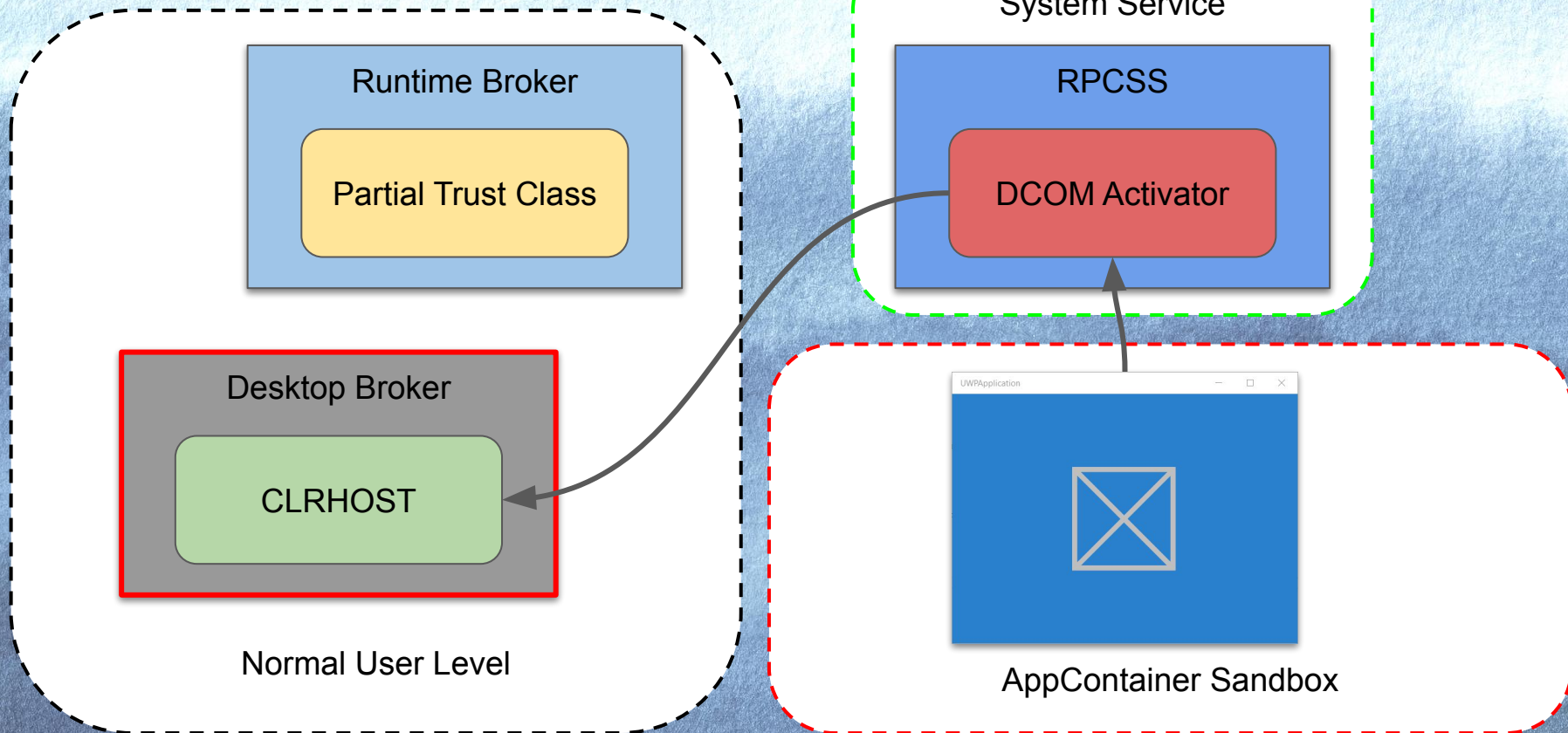


Escaping Sandboxes

Partial Trust



Desktop Broker



Brokered Windows Runtime Components for a side-loaded UWP app

This article discusses an enterprise-targeted feature supported by Windows 10, which allows touch-friendly .NET apps to use the existing code responsible for key business-critical

operations. <https://docs.microsoft.com/en-us/windows/uwp/winrt-components/brokered-windows-runtime-components-for-side-loaded-windows-store-apps>

```
<Extension Category="windows.activatableClass.inProcessServer">
  <InProcessServer>
    <Path>clrhost.dll</Path>
    <ActivatableClass ActivatableClassId="Runtime.Class">
      <ActivatableClassAttribute Name="DesktopApplicationPath"
                                Type="string" Value="c:\test" />
    </ActivatableClass>
  </InProcessServer>
</Extension>
```

Relative path I'd expect

Weird absolute path I'd not.

Desktop Broker Security

```
Windows PowerShell
PS C:\> Get-ComAppId -Name 'WinRTDesktopBroker' | fl RunAs, Flags, LaunchPermission, AccessPermission

RunAs      : Interactive User
Flags      : IUServerUnmodifiedLogonToken,
            IUServerSelfSidInLaunchPermission,
            RequireSideLoadedPackage
LaunchPermission : 0:SYG:SYD:(A;;;CCDCSW;;;PS)(A;;;CCDCSW;;;AC)S:(ML;;;NX;
                  ;;LW)
AccessPermission : 0:SYG:SYD:(A;;;CCDC;;;PS)(A;;;CCDC;;;SY)(A;;;CCDC;;;LS)
                  (A;;;CCDC;;;NS)(A;;;CCDC;;;AC)S:(ML;;;NX;;;LW)
```

Interactive User runs
outside sandbox

“Require” Side Loaded Package?

Allows current user and all
AppContainers to launch/access

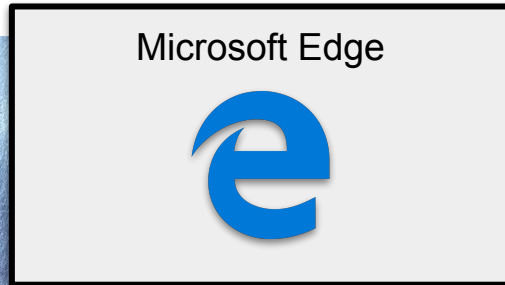
Side Loaded Package Launch Check

```
void IsSideLoadedPackage(LPCWSTR PackageName,  
                        bool *IsSideloaded) {  
    PackageOrigin origin;  
    *IsSideloaded = false;  
    GetStagedPackageOrigin(package_name, &origin);  
  
    *IsSideloaded = origin != PackageOrigin_Store;  
    return S_OK;  
}
```

```
enum PackageOrigin {  
    PackageOrigin_Inbox,  
    PackageOrigin_Store,  
    PackageOrigin_LineOfBusiness  
};
```


Side Loaded Package Launch Check

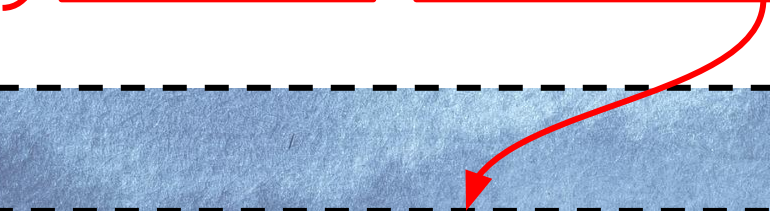
```
void IsSideLoadedPackage(LPCWSTR PackageName,  
                        bool *IsSideloaded) {  
    PackageOrigin origin;  
    *IsSideloaded = false;  
    GetStagedPackageOrigin(package_name, &origin);  
  
    *IsSideloaded = origin != PackageOrigin_Store;  
    return S_OK;  
}
```



```
enum PackageOrigin {  
    PackageOrigin_Inbox,  
    PackageOrigin_Store,  
    PackageOrigin_LineOfBusiness  
};
```


Supported Interfaces

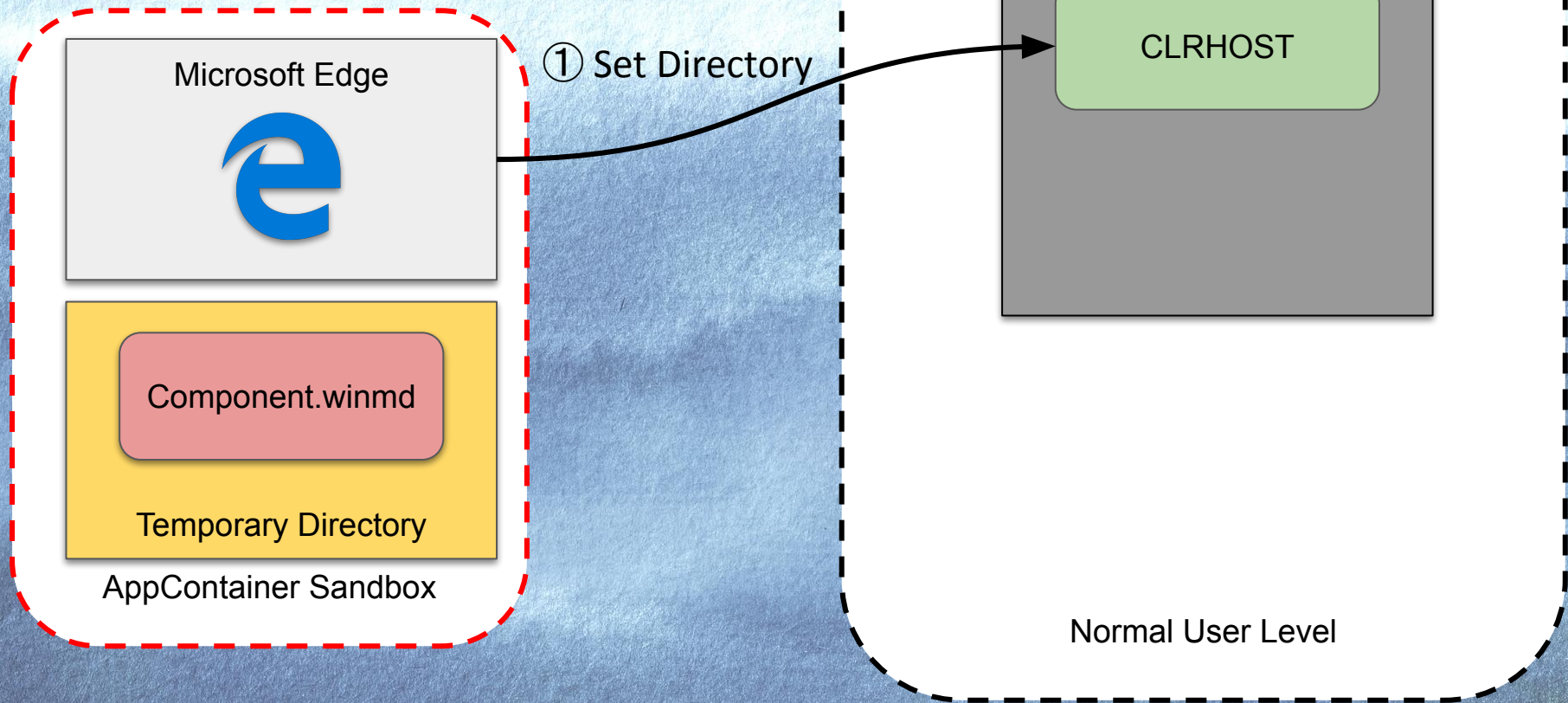
```
interface IWinRTDesktopBroker : IUnknown {  
    HRESULT GetClassActivatorForApplication(  
        Specify arbitrary directory HSTRING dir, IWinRTClassActivator** ppv);  
};
```



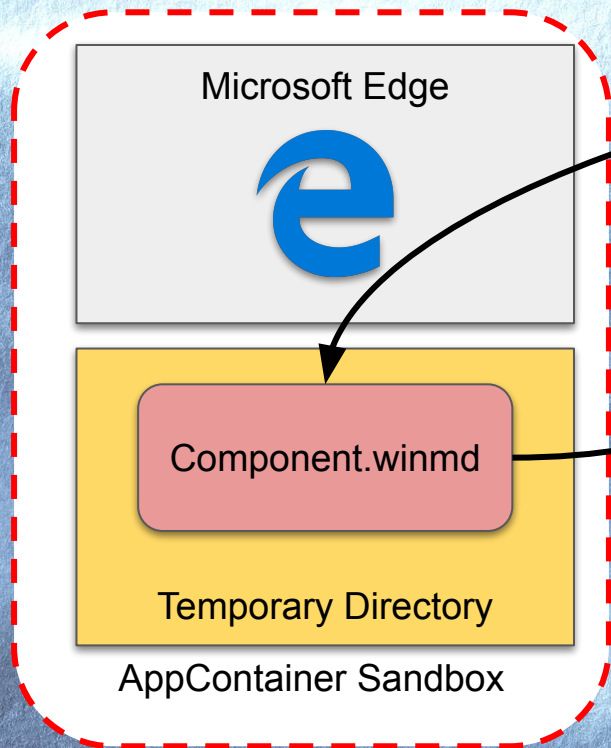
```
interface IWinRTClassActivator : IUnknown {  
    HRESULT ActivateInstance(HSTRING activatableClassId,  
                             IInspectable** ppv);  
    HRESULT GetActivationFactory(HSTRING activatableClassId,  
                                 REFIID riid, IUnknown** ppv);  
};
```

Also a .NET COM object!

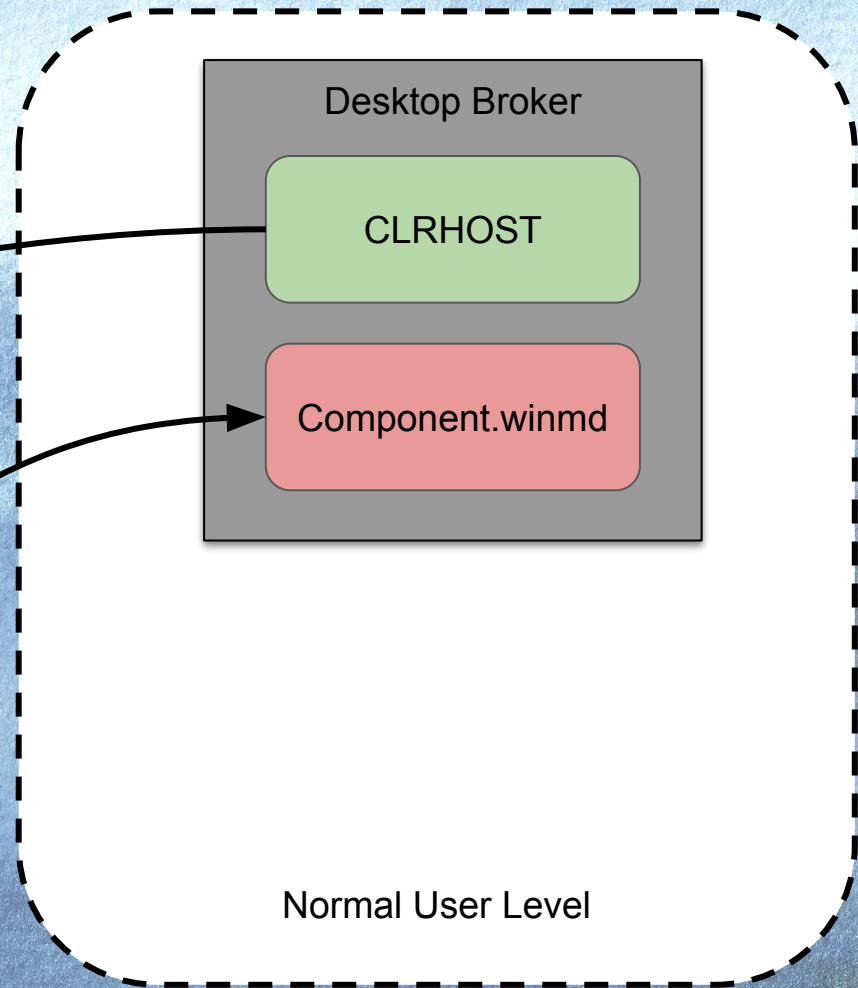
Exploitation



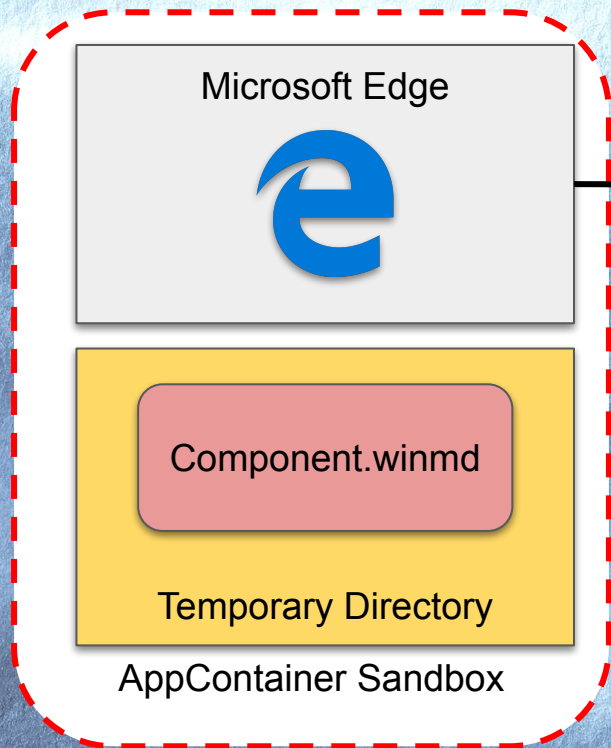
Exploitation



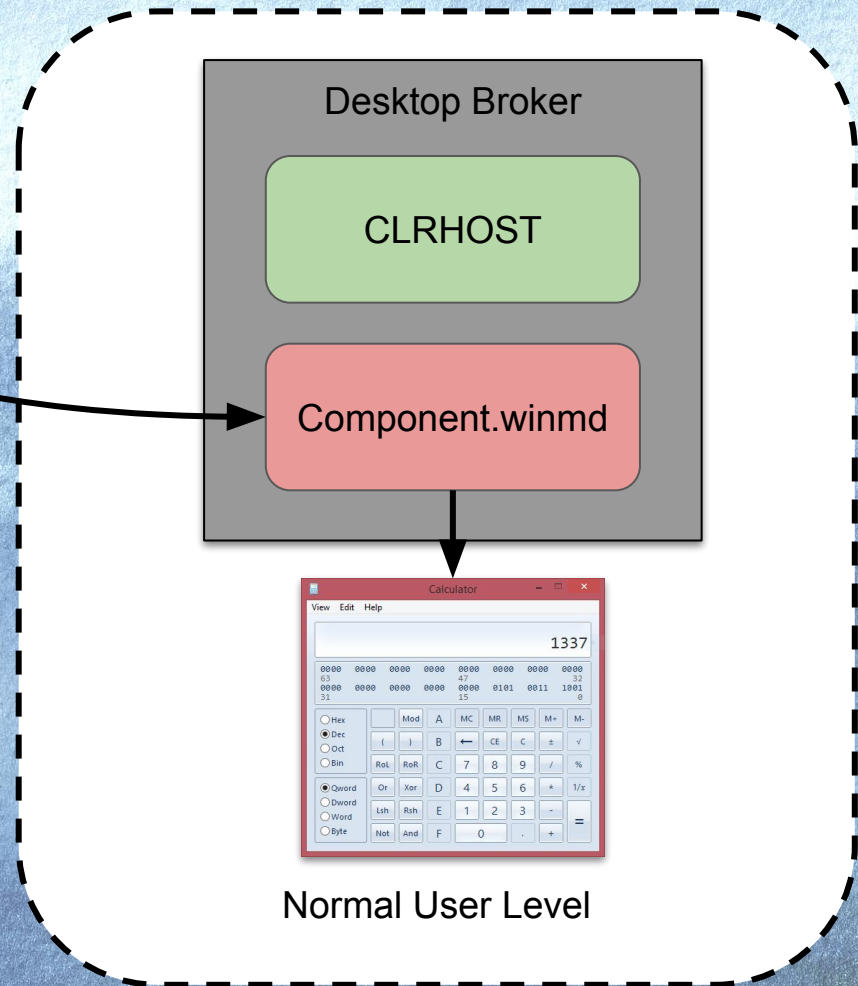
② Load
WINMD



Exploitation



③ Execute Component



Fixed as CVE-2019-0552

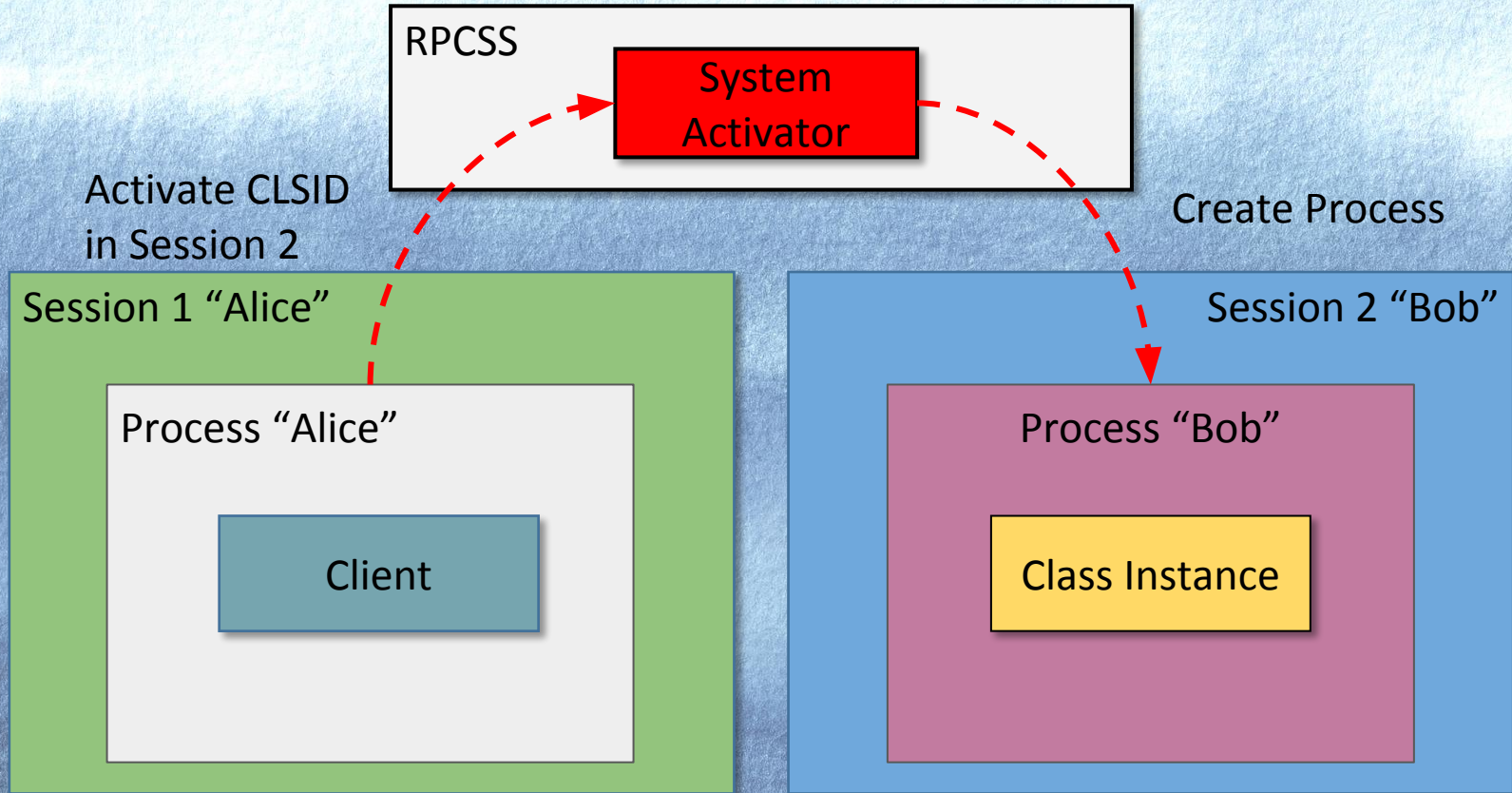
```
void IsSideLoadedPackage(LPCWSTR PackageName,  
                        bool *IsSideloaded) {  
    PackageOrigin origin;  
    *IsSideloaded = false;  
    GetStagedPackageOrigin(package_name, &origin);  
  
    *IsSideloaded = origin == PackageOrigin_LineOfBusiness;  
    return S_OK;  
}
```

Only allow Side Loaded applications



Cross-Session Attacks

Session Moniker in Action



Fixed?

CVE-2017-0100 | Windows COM Session Elevation of Privilege Vulnerability

Security Vulnerability

Published: 03/14/2017

An elevation of privilege exists in Windows when a DCOM object in Helppane.exe, configured to run as the interactive user, fails to properly authenticate the client. An attacker who successfully exploited the vulnerability could run arbitrary code in another user's session.

To exploit the vulnerability, an attacker would first have to log on to the system. An attacker could then run a specially crafted application that could exploit the vulnerability after another user logged on to the same system via Terminal Services or Fast User Switching.

Not Really!

Issue 1224: Windows: Bad Fix for COM Session

[Code](#)[< Prev](#)

90 of 139

[Next >](#)

Moniker EoP

Reported by forshaw@google.com on Sat, Mar 25, 2017, 12:29 AM GMT

[Back to list](#)

Windows: Bad Fix for COM Session Moniker EoP

So....

← Passive-Aggressive Britishism

The previous fix for CVE-2017-0100 sounds wrong on the face of it. Rather than fixing the underlying Session creation bug you "fixed" the HxHelpPane class. Even if this was a correct fix ultimately it just requires you to find an alternative COM object to abuse.

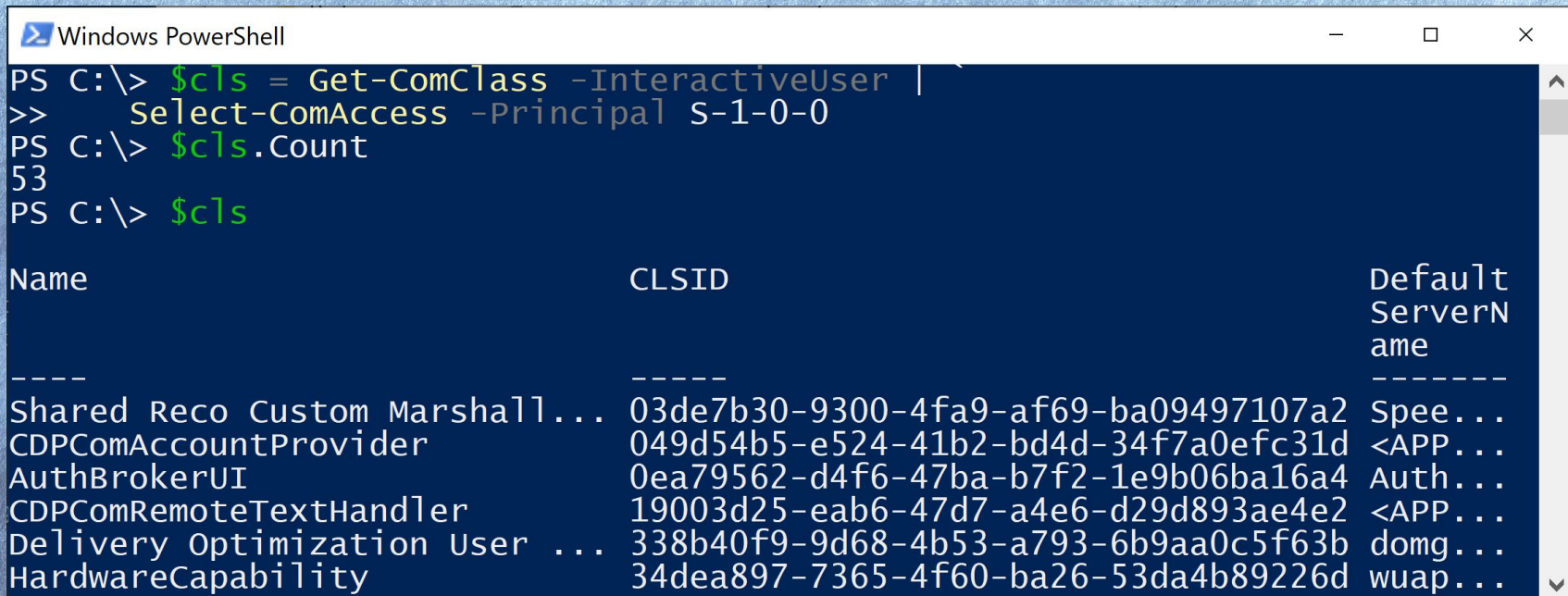
However looking at the fix in HelpPane.exe you can see that the fix isn't actually sufficient. The bug is in the check.

```
if ( imp_token_il >= process_token_il
    && (imp_token_il >= SECURITY_MANDATORY_HIGH_RID
    || EqualSid(process_token_user, imp_token_user)))
{
    ShellExecuteW(NULL, L"open", path, NULL, NULL, SW_SHOW);
}
```

← Integrity levels are NOT a security boundary enforcement mechanism.

Find Potential Cross-Session Objects

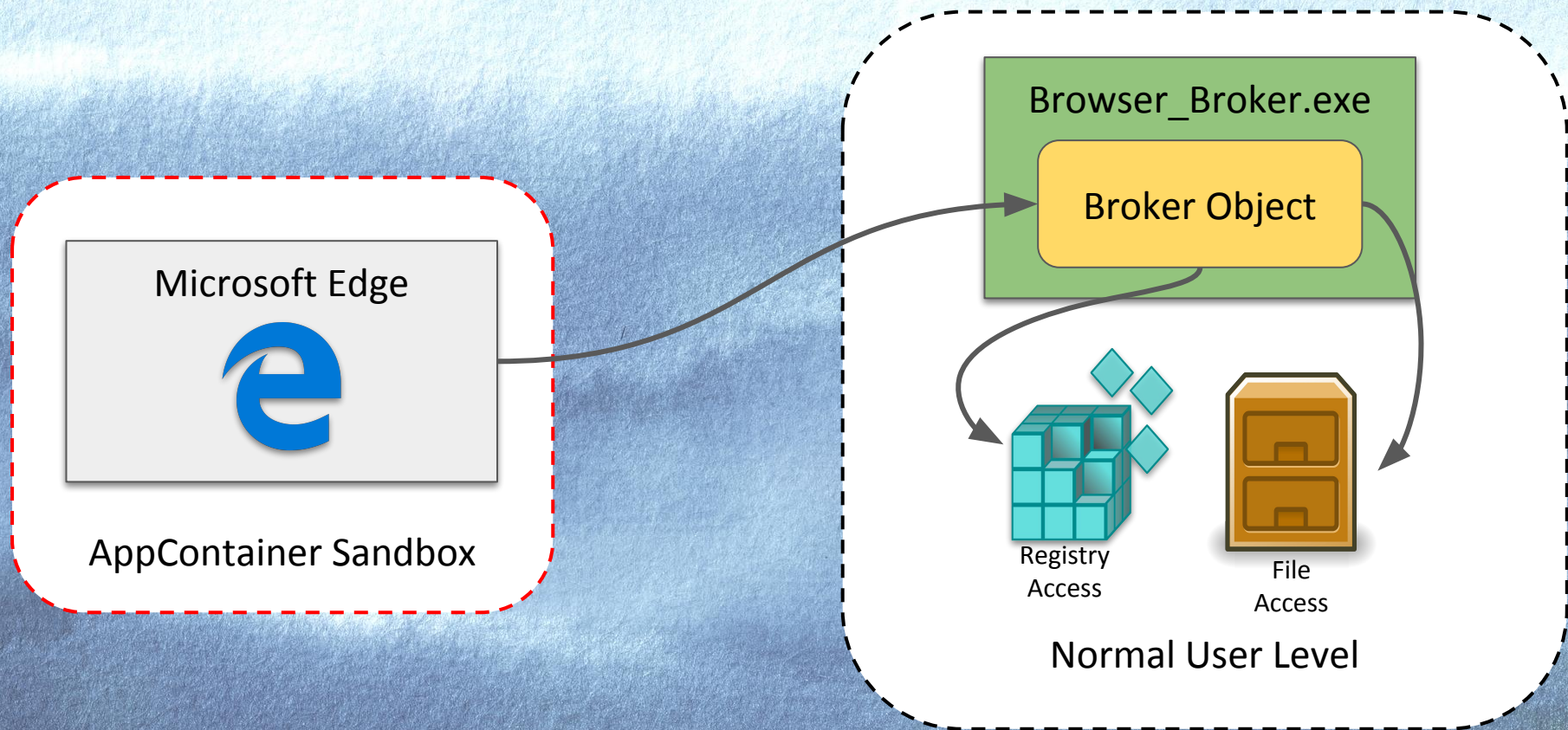
```
PS> Get-ComClass -InteractiveUser | `
    Select-ComAccess -Principal S-1-0-0
```



```
Windows PowerShell
PS C:\> $cls = Get-ComClass -InteractiveUser | `
>>     Select-ComAccess -Principal S-1-0-0
PS C:\> $cls.Count
53
PS C:\> $cls
```

Name	CLSID	Default ServerName
Shared Reco Custom Marshall...	03de7b30-9300-4fa9-af69-ba09497107a2	Spee...
CDPComAccountProvider	049d54b5-e524-41b2-bd4d-34f7a0efc31d	<APP...
AuthBrokerUI	0ea79562-d4f6-47ba-b7f2-1e9b06ba16a4	Auth...
CDPComRemoteTextHandler	19003d25-eab6-47d7-a4e6-d29d893ae4e2	<APP...
Delivery Optimization User ...	338b40f9-9d68-4b53-a793-6b9aa0c5f63b	domg...
HardwareCapability	34dea897-7365-4f60-ba26-53da4b89226d	wuap...

Browser Broker



Browser Broker Security

OLE BrowserBrokerServer Launch... - □ ×

Owner: BUILTIN\Administrators
Group: BUILTIN\Administrators
Integrity: Low
DACL SACL

Launch Permissions

ACL Entries

Type	Account	Access
Allowed	Everyone	Execute
Allowed	microsoftedge_8wekyb3d8bbwe	Execute

Everyone can launch

Specific Access

Name	Access Mask
<input checked="" type="checkbox"/> Execute	0x00000001
<input checked="" type="checkbox"/> Execute Local	0x00000002
<input type="checkbox"/> Execute Remote	0x00000004
<input checked="" type="checkbox"/> Activate Local	0x00000008
<input type="checkbox"/> Activate Remote	0x00000010

OLE Access Security - □ ×

Owner: BUILTIN\Administrators
Group: BUILTIN\Administrators
Integrity: N/A
DACL

Access Permissions

ACL Entries

Type	Account	Access
Allowed	NT AUTHORITY\Authenticated Users	Execu
Allowed	BUILTIN\Guests	Execu
Allowed	microsoftedge_8wekyb3d8bbwe	Execu
Allowed	NAMED CAPABILITIES\Lpac Web Platform	Execu

Specific Access

Name	Access Mask
<input checked="" type="checkbox"/> Execute	0x00000001
<input checked="" type="checkbox"/> Execute Local	0x00000002
<input type="checkbox"/> Execute Remote	0x00000004

Browser Broker Limitations

```
Windows PowerShell
PS C:\> $cls = Get-ComClass -Name "BrowserBroker Class"
PS C:\> $o = New-ComObject $cls
Exception calling "CreateInstanceAsObject" with "2" argument(s): "Server
execution failed (Exception from HRESULT: 0x80080005
(CO_E_SERVER_EXEC_FAILURE))"
At C:\Users\user\Documents\WindowsPowerShell\Modules\oleviewdotnet\1.7\OleviewDotNet.psm1:1448 char:17
+ ... $obj = $Class.CreateInstanceAsObject($Remo ...
+ ~~~~~
+ CategoryInfo          : NotSpecified
+ FullyQualifiedErrorId : COMException

PS C:\>
```

```
HRESULT BrokerAuthenticateCOMCaller() {
HANDLE TokenHandle;
LPWSTR FamilyName;

OpenThreadToken(TOKEN_QUERY, &TokenHandle);
RtlQueryPackageClaims(TokenHandle, &FamilyName);

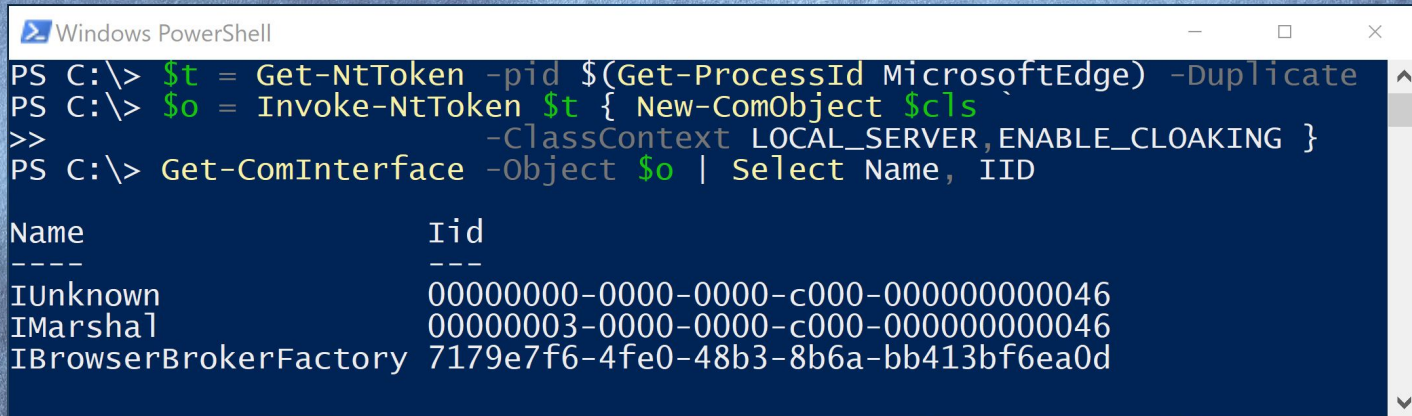
if (strcmp(FamilyName, "MicrosoftEdge")) {
    Log("Caller is not Edge");
    DebugBreak();
}
}
```

Caller must be Edge

Creating Instance to Test

```
PS> $token = Get-NtToken -Duplicate `
        -pid $(Get-Process MicrosoftEdge).Id
```

```
PS> Invoke-NtToken $token { New-ComObject $cls `
        -ClassContext LOCAL_SERVER, ENABLE_CLOAKING }
```



The screenshot shows a Windows PowerShell window with the following commands and output:

```
PS C:\> $t = Get-NtToken -pid $(Get-ProcessId MicrosoftEdge) -Duplicate
PS C:\> $o = Invoke-NtToken $t { New-ComObject $cls
>> -ClassContext LOCAL_SERVER, ENABLE_CLOAKING }
PS C:\> Get-ComInterface -Object $o | Select Name, IID
```

Name	Iid
IUnknown	00000000-0000-0000-c000-000000000046
IMarshal	00000003-0000-0000-c000-000000000046
IBrowserBrokerFactory	7179e7f6-4fe0-48b3-8b6a-bb413bf6ea0d

Create with
impersonation
token

Chaining to ShellExecute

Current Session

```
HANDLE hToken = OpenEdgeProcessToken();  
ImpersonateLoggedOnUser(hToken);
```



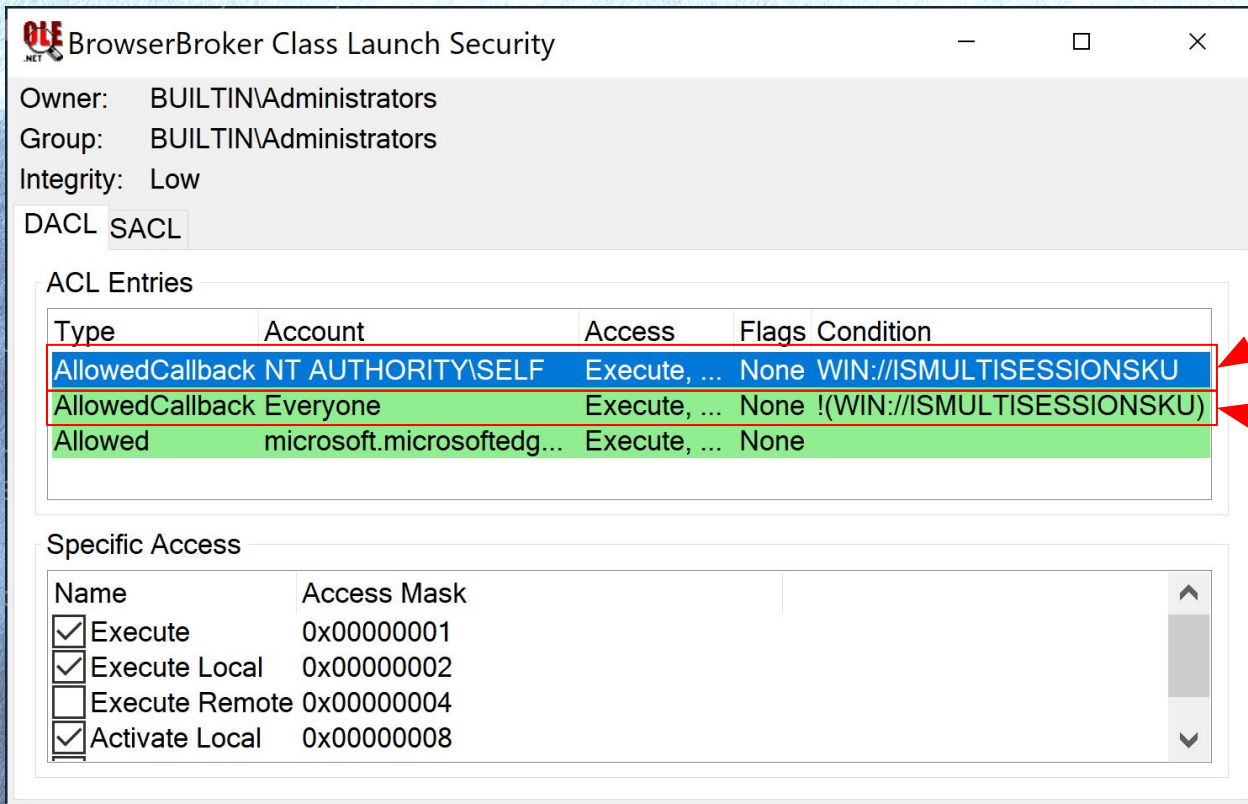
```
interface IDownloadExecutionBroker : IUnknown {  
    // Other Methods...  
    HRESULT ExecuteFile(LPCWSTR path, LPCWSTR verb);  
}
```



Other User Session

```
ShellExecute(verb, path, ...);
```

Fixed as CVE-2019-0566



Added "SELF" SID
access on
multi-session SKUs.

Left Everyone group
for non-multi-session
SKUs



COM Marshalling

Slave to the Cat

1 = Standard OBJREF
4 = Custom OBJREF

'MEOW'	OBJREF Type
IID (lower 64 bits)	
IID (upper 64 bits)	

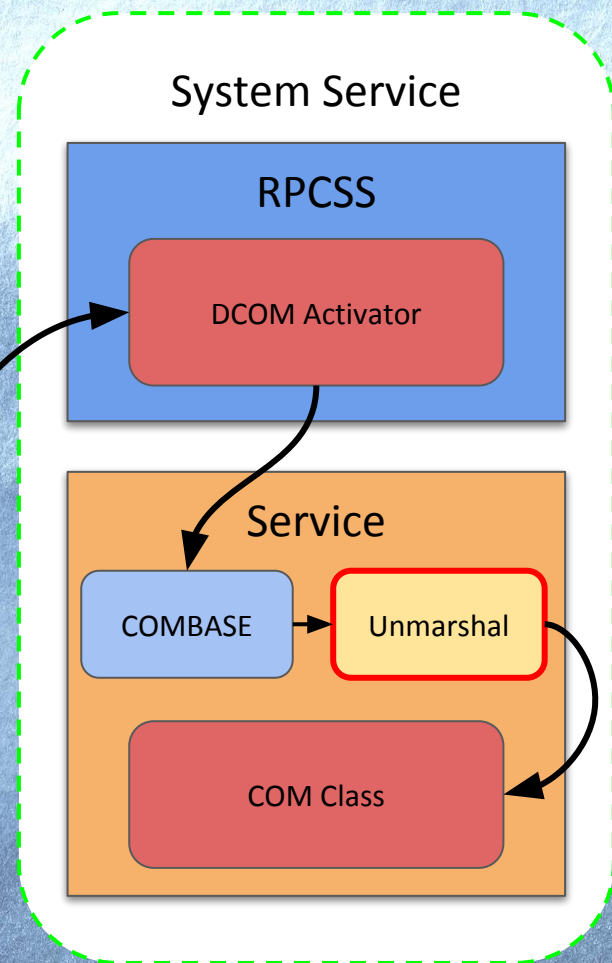
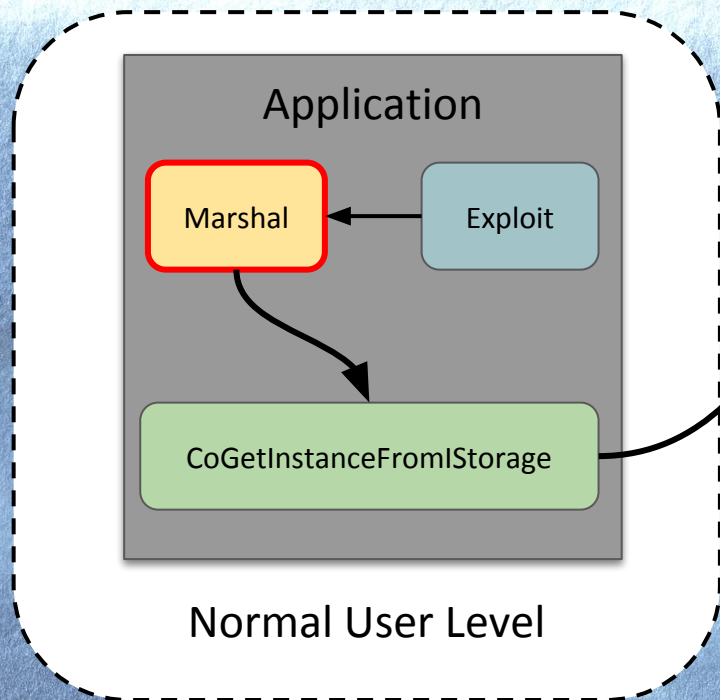
Flags	References
Object Exporter ID (OXID)	
Object ID (OID)	
Interface Pointer ID (IPID)	
IPID (upper 64 bits)	
Binding information for remote access	

Standard OBJREF

CLSID (lower 64 bits)	
CLSID (upper 64 bits)	
Reserved	Extension Size
Custom Data	

Custom OBJREF

Object Injecting



"Fake" Custom Marshaler

```
class MyFakeObject : IMarshal, IStorage {  
    HRESULT GetUnmarshalClass (CLSID *pCid) {  
        pCid = CLSID_WhatIWant;  
        return S_OK;  
    }  
  
    HRESULT MarshalInterface(IStream *pStm) {  
        return pStm->Write(MarshaledData,  
                           sizeof(MarshaledData));  
    }  
}
```

Implement IMarshal and IStorage

Specify CLSID for object to create

Write data to unmarshal

Restriction on Custom Marshaling

EOAC_NO_CUSTOM_MARSHAL

All Windows
Versions

Specifying this flag helps protect server security when using DCOM or COM+. It reduces the chances of executing arbitrary DLLs because it allows the marshaling of only CLSIDs that are implemented in Ole32.dll, ComAdmin.dll, ComSvc.dll, or Es.dll, or that implement the CATID_MARSHALER category ID. Any service that is critical to system operation should set this flag.


COMGLB_UNMARSHALING_POLICY

Windows 8+

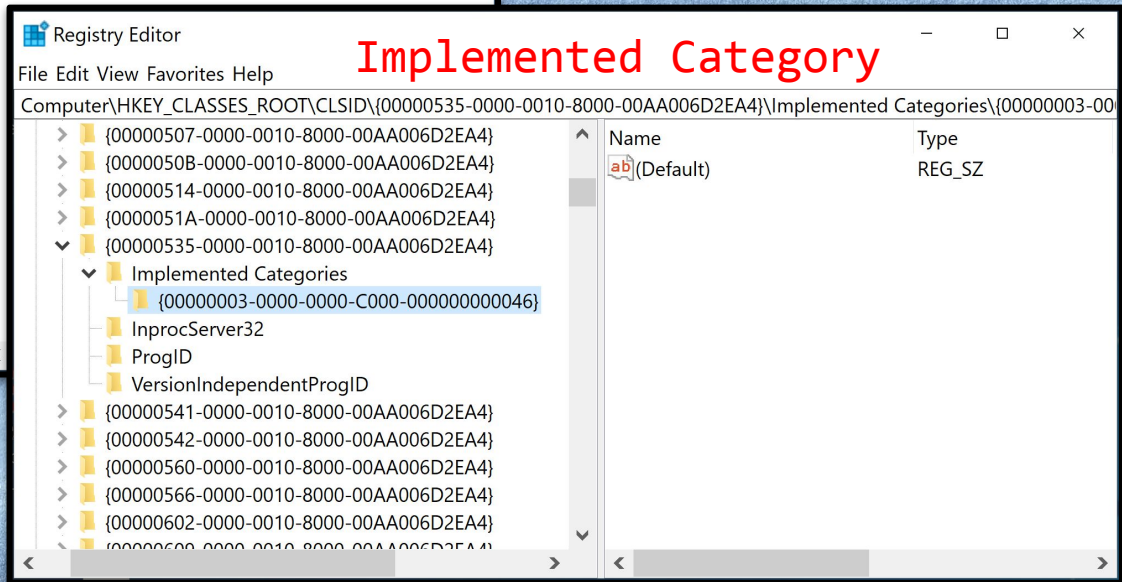
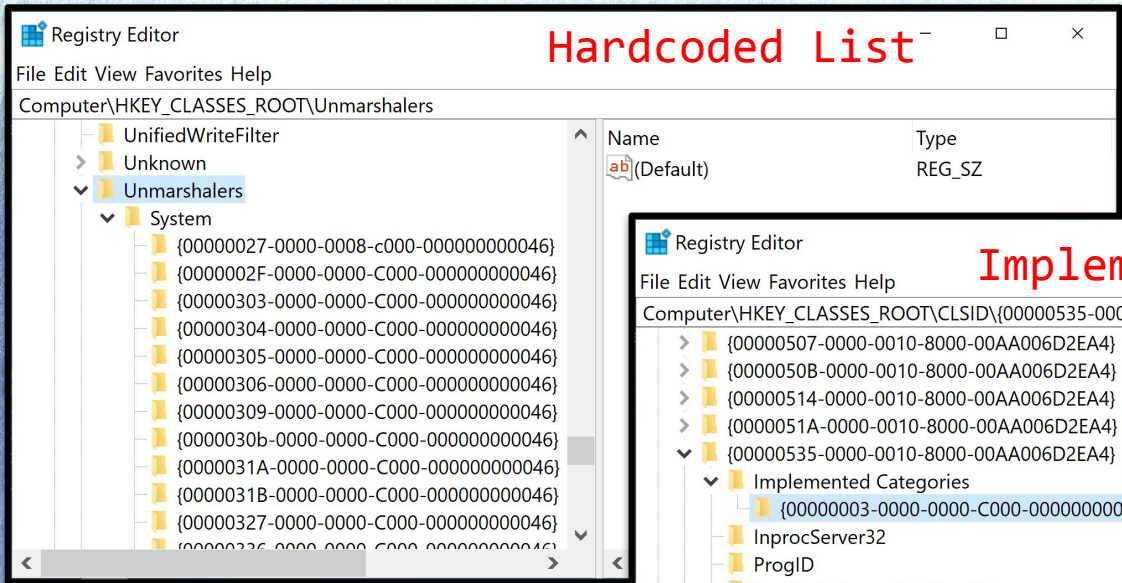
Possible values for the COMGLB_UNMARSHALING_POLICY property are:

- COMGLB_UNMARSHALING_POLICY_NORMAL: Unmarshaling behavior is the same as versions before than Windows 8. **EOAC_NO_CUSTOM_MARSHAL** restrictions apply if this flag is set in [CoInitializeSecurity](#). Otherwise, there are no restrictions. This is the default for processes that aren't in the app container.
- COMGLB_UNMARSHALING_POLICY_STRONG: Unmarshaling allows only a system-trusted list of hardened unmarshallers and unmarshallers allowed per-process by the [CoAllowUnmarshallerCLSID](#) function. This is the default for processes in the app container.
- COMGLB_UNMARSHALING_POLICY_HYBRID: Unmarshaling data whose source is app container allows only a system-trusted list of hardened unmarshallers and unmarshallers allowed per-process by the [CoAllowUnmarshallerCLSID](#) function. Unmarshaling behavior for data with a source that's not app container is unchanged from previous versions.

Bypassed if a "System
Trusted" marshaler is used.



System Trusted Marshalers



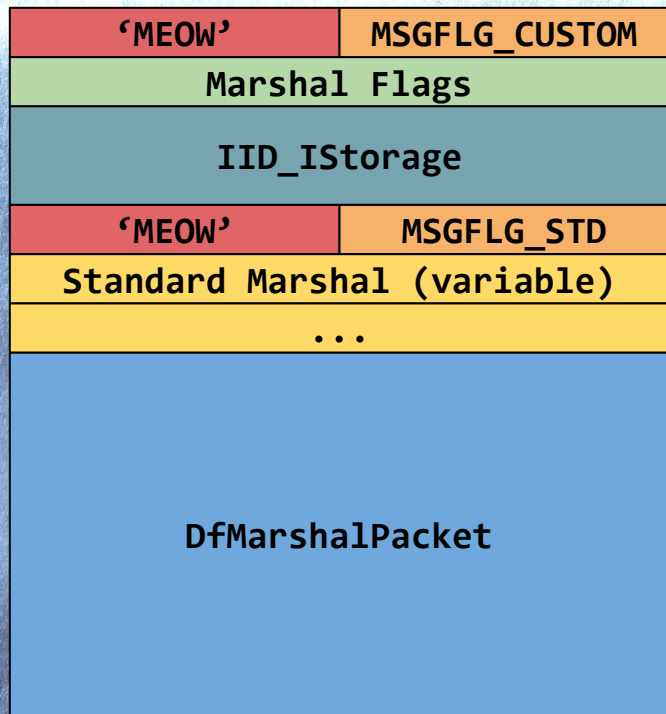
Find Trusted Marshallers

```
PS> Get-ComClass -TrustedMarshaller
```

```
Windows PowerShell
PS C:\> $cls = Get-ComClass -TrustedMarshaller
PS C:\> $cls.Count
167
PS C:\> $cls
```

Name	CLSID	Default ServerName
-----	-----	-----
CLSID_RecordInfo	0000002f-0000-0000-c000-000000000046	olea...
FileMoniker	00000303-0000-0000-c000-000000000046	comb...
ItemMoniker	00000304-0000-0000-c000-000000000046	comb...
AntiMoniker	00000305-0000-0000-c000-000000000046	comb...
PointerMoniker	00000306-0000-0000-c000-000000000046	comb...
CompositeMoniker	00000309-0000-0000-c000-000000000046	comb...
DfMarshal	0000030b-0000-0000-c000-000000000046	coml...

DocFile Marshaller (DfMarshal)



```
struct SDfMarshalPacket {  
    CBasedPubDocFilePtr pdf;  
    // ...  
    CBasedGlobalFileStreamPtr fsBase;  
    unsigned int ulHeapName;  
    unsigned int cntxid;  
    GUID cntxkey;  
    CPerContext *ppc;  
    HANDLE hMem;  
};
```


Shared memory handle

Pretty Bad Code

<i>Issue Number</i>	<i>Description</i>
1644	Master Issue, Background and Description
1645	Missing Pointer Bounds Checks
1646	Shared Allocator
1647	Arbitrary File Deletion
1648	Steal Arbitrary Handles from Privileged Process

Shared Allocator (Issue 1646)

```
CExposedDocFile* GetExposedDocFile() {  
    CExposedDocFile* df = new CExposedDocFile();  
    df->AddRef();  
    return df;  
}
```



```
void* operator new(size_t size) {  
    CSmAlloc* a = GetTlsSmAlloc();  
    return a->Alloc(size);  
}
```

```
CExposedDocFile() {  
    this->VTable = &vft_IStorage;  
    // ...  
}
```


Handle 2284 - 0x280FE130000 (ReadWrite)																
	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
000002D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000002E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000002F0	00	00	00	00	00	00	00	00	10	01	00	00	00	00	00	00
00000300	00	00	00	00	00	00	00	00	70	F4	6D	55	F8	7F	00	00
00000310	38	F4	6D	55	F8	7F	00	00	08	03	C7	D3	0D	02	00	00
00000320	48	03	C7	D3	0D	02	00	00	50	53	53	54	00	00	00	00
00000330	01	00	00	00	01	00	00	00	18	F4	6D	55	F8	7F	00	00
00000340	D0	F3	6D	55	F8	7F	00	00	A8	F3	6D	55	F8	7F	00	00
00000350	78	F3	6D	55	F8	7F	00	00	50	F3	6D	55	F8	7F	00	00
00000360	F0	FA	6D	55	F8	7F	00	00	00	00	00	00	00	00	00	00
Position	776/0x308															
Selection Length	6/0x6															
Byte	112/0x70															
SByte	112/0x70															
Int16 (Little Endian)	-2960/0xF470															
Int16 (Big Endian)	28916/0x70F4															
Int32 (Little Endian)	1433269360/0x556DF470															
Int32 (Big Endian)	1895066965/0x70F46D55															
Int64 (Little Endian)	140704561886320/0x7FF8556DF470															
Int64 (Big Endian)	8139250642574049280/0x70F46D55F87F0000															
UInt16 (Little Endian)	62576/0xF470															
UInt16 (Big Endian)	28916/0x70F4															
UInt32 (Little Endian)	1433269360/0x556DF470															
UInt32 (Big Endian)	1895066965/0x70F46D55															

0:006> !address 7FF8556DF470


Usage: Image
 Base Address: 00007ff8`556df000
 End Address: 00007ff8`556ec000
 Region Size: 00000000`0000d000
 Protect: 00000002 PAGE_READONLY
 Image Path: C:\WINDOWS\System32\coml2.dll
 Module Name: coml2

0:006> dqs 7FF8556DF470
 coml2!CExposedDocFile::QueryInterface
 coml2!CExposedDocFile::AddRef
 coml2!CExposedDocFile::Release
 ...

Duplicating Shared Memory

```
HRESULT CSharedMemoryBlock::InitUnMarshal(void *hMem,  
                                           unsigned int dwProcessId) {  
    unsigned int dwCurrentSession;  
    unsigned int dwSourceSession;  
  
    ProcessIdToSessionId(dwProcessId, &dwSourceSession);  
    ProcessIdToSessionId(GetCurrentProcessId(), &dwCurrentSession);  
    if (dwSourceSession != dwCurrentSession)  
        return E_ACCESSDENIED;  
    HANDLE hProcess = OpenProcess(PROCESS_DUP_HANDLE, dwProcessId);  
    DuplicateHandle(hProcess, hMem, GetCurrentProcess(), ...);  
    ...  
}
```

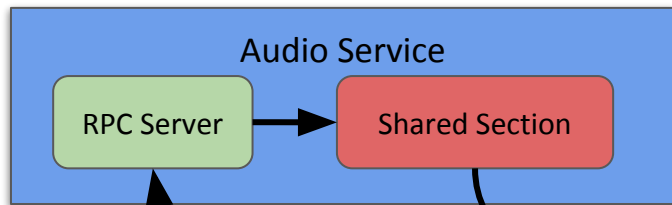
Source and destination process
must be in same session



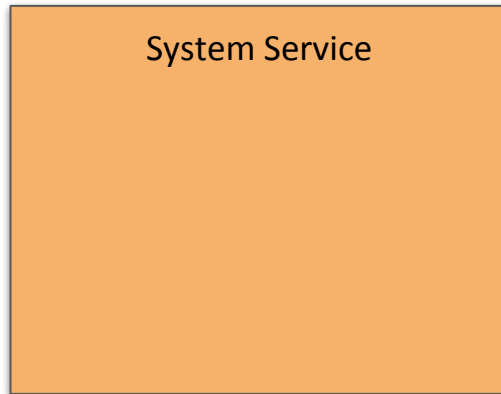
Audio Server to the Rescue

Session 0

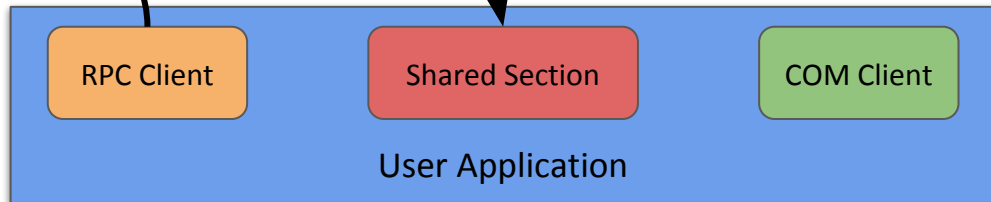
① Call Audio Service to
Allocate Shared Section



System Service



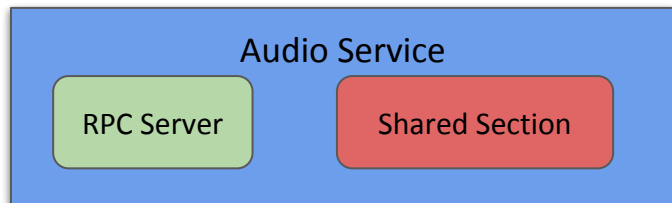
Session N



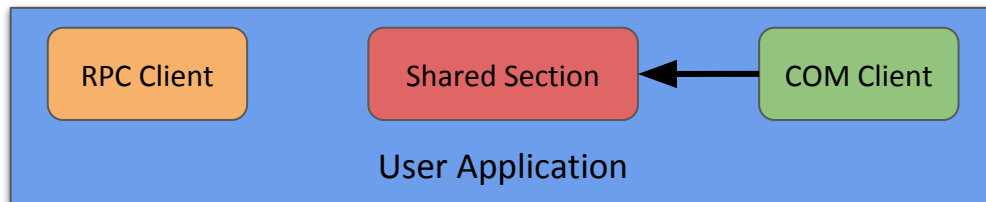
Audio Server to the Rescue

Session 0

② Modify shared section for exploit.

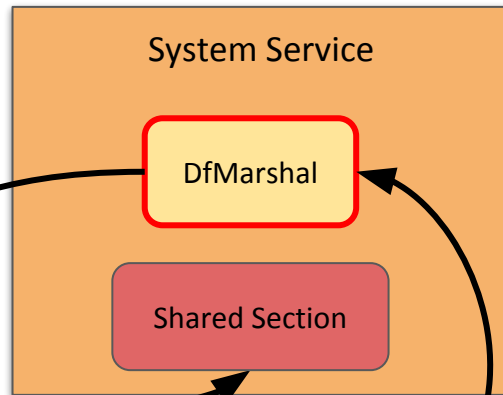
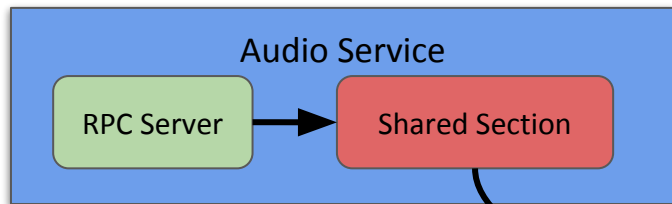


Session N

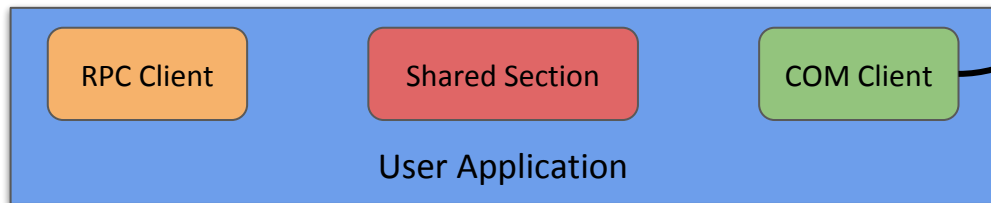


Audio Server to the Rescue

Session 0 ③ Inject DfMarshal and copy section from Audio Service



Session N



Discovering the Audio Server

```
PS> $r = ls \windows\system32\*.dll | Get-RpcServer
```

```
function Select-Procedure {  
    Param([NtApiDotNet.Ndr.NdrProcedureParameter]$proc)  
  
    foreach($p in $proc.Params) {  
        if ($p.Type.Format -eq "FC_SYSTEM_HANDLE" -and  
            $p.Type.Resource -eq "Section") {  
            return $true  
        }  
    }  
}
```

Look for any procedure which marshals a section handle.



DEMO
TIME

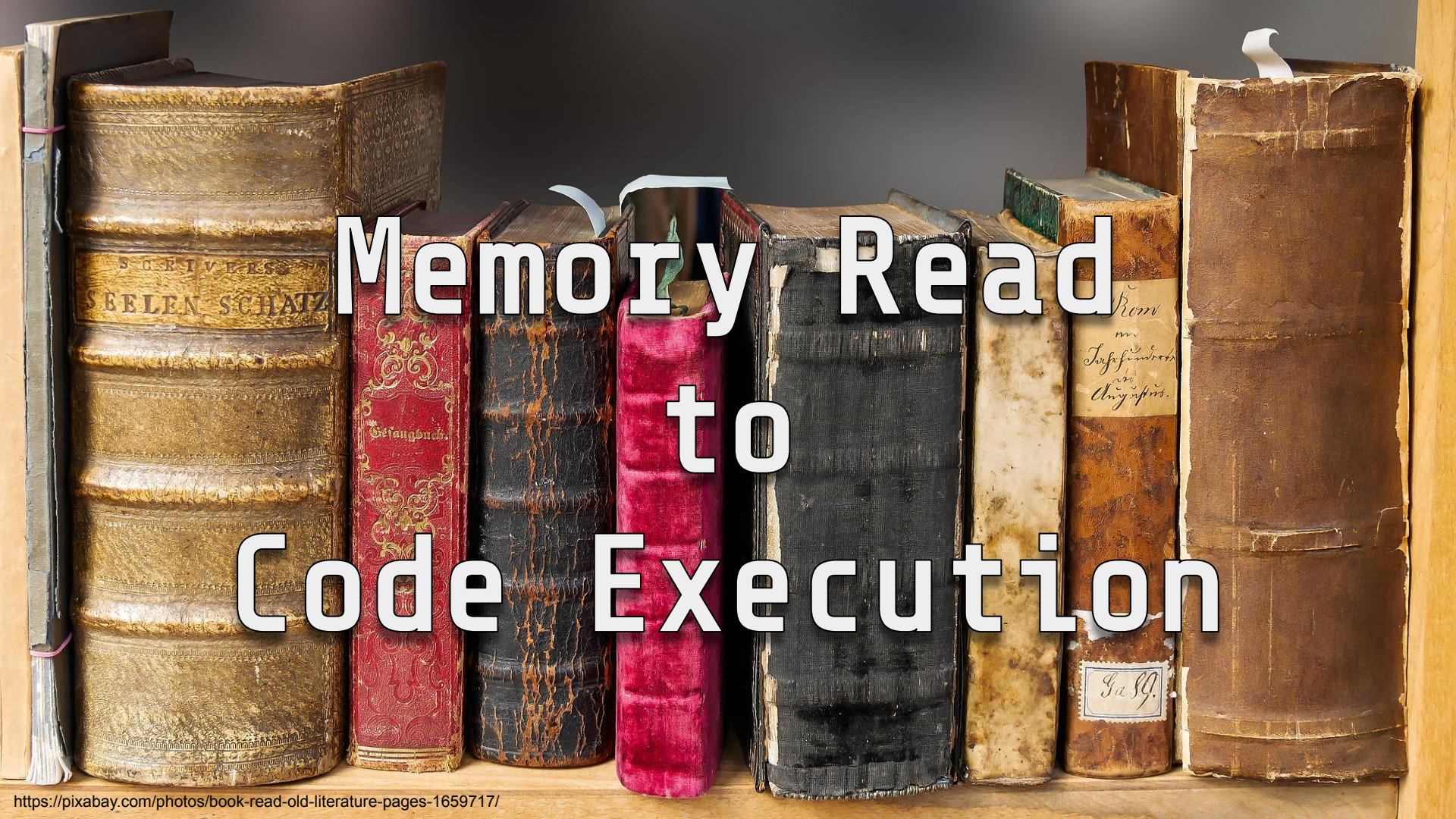
Fixed in CVE-2018-8550

Optional
hard-ban

```
bool IsSharedMemoryModeDisabledForCurrentProcess() {  
    return IsSharedMemoryModeDisabledForCurrentSystemViaPolicy()  
        || ProcessToken::IsAppContainer()  
        || IsSharedMemoryModeDisabledForCurrentProcessDueToIL();  
}
```

Block in processes
with IL \geq High

Block in
AppContainer
Processes

A row of seven old, worn books standing on a wooden shelf. The books have various colored spines: gold, red, dark blue, red, black, light brown, and dark brown. The text 'Memory Read to Code Execution' is overlaid in large, white, sans-serif font across the center of the books. The word 'Memory' is on the first line, 'Read' is on the second line, 'to' is on the third line, and 'Code Execution' is on the fourth line. The books are slightly aged and show signs of wear, with some labels visible on the spines.

Memory Read to Code Execution

Finding Code Injection

```
PS> Get-ComProcess -pid X -ParseStubMethods `
      -ResolveMethodNames | Format-ComProxy
```

Windows PowerShell

```
PS C:\> $p = Get-ComProcess -pid 7440 -ParseStubMethods `
>> -ResolveMethodNames
PS C:\> $p.Ipids | Format-ComProxy -Flags RemoveComplexTypes
[Guid("00000134-0000-0000-c000-000000000046")]
interface IRundown : IUnknown {
    HRESULT RemQueryInterface(/* Stack Offset: 4 */ [In] GUID* p0, /*
    Stack Offset: 8 */ [In] int p1, /* Stack Offset: 12 */ [In] /* range
: 1,32768 */ short p2, /* Stack Offset: 16 */ [In] /* C:(FC_TOP_LEVEL
_CONFORMANCE)(12)(FC_ZERO)(FC_USHORT)(17) */ GUID[]* p3, /* Stack Off
set: 20 */ [Out] /* C:(FC_TOP_LEVEL_CONFORMANCE)(12)(FC_ZERO)(FC_USHO
RT)(17) */ /* Unhandled */ FC_HARD_STRUCT[]* p4);
    HRESULT RemAddRef(/* Stack Offset: 4 */ [In] short p0, /* Stack O
ffset: 8 */ [In] /* C:(FC_TOP_LEVEL_CONFORMANCE)(4)(FC_ZERO)(FC_USHOR
T)(Early) */ struct Struct_1[] p1, /* Stack Offset: 12 */ [Out] /* C:
```


IRundown Interface

```
[Guid("00000134-0000-0000-c000-000000000046")]
```

```
interface IRundown : IUnknown {
```

```
    HRESULT RemQueryInterface(...);
```

```
    HRESULT RemAddRef(...);
```

```
    HRESULT RemRelease(...);
```

```
    HRESULT RemQueryInterface2(...);
```

```
    HRESULT RemChangeRef(...);
```

```
    HRESULT DoCallback(struct Struct_3* p0);
```

```
    HRESULT DoNonreentrantCallback(struct Struct_3* p0);
```

```
    HRESULT AcknowledgeMarshalingSets(...);
```

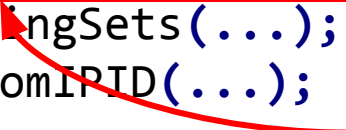
```
    HRESULT GetInterfaceNameFromIPID(...);
```

```
    HRESULT RundownOid(...);
```

IRemUnknown
implementation



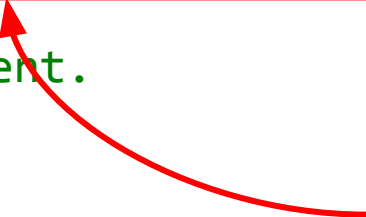
Callbacks sound
interesting



```
}
```

DoCallback Implementation

```
HRESULT CRemoteUnknown::DoCallback(XAptCallback *pCallbackData) {  
    if (CProcessSecret::g_guidProcessSecret ==  
        pCallbackData->guidProcessSecret) {  
  
        if (pCallbackData->pServerCtx == GetCurrentContext()) {  
            return pCallbackData->pfnCallback(pCallbackData->pParam);  
        }  
  
        // Dispatch to another apartment.  
    }  
  
    return E_INVALIDARG;  
}
```



Calls arbitrary
pointer from
client

Criteria for Executing Callback

<i>Criteria</i>	<i>How?</i>
Process Secret GUID	Read fixed memory location
Context Pointer	Read fixed <i>g_pMTAEmptyCtx</i> location
IRundown IPID	Returned when connected to COM server
Target Function	Pick an exported function (N.B. CFG)
Parameter	Be imaginative.

Exploiting VirtualBox

Issue 1811: VirtualBox: COM RPC Interface Code Injection

[Code](#)[< Prev](#)

4 of 4

[Back to list](#)

Host EoP

Reported by forshaw@google.com on Tue, Mar 26, 2019, 2:46 PM GMT (19 days ago)

[Edit description](#)

VirtualBox: COM RPC Interface Code Injection Host EoP

Platform: VirtualBox 6.0.4 [r128413](#) x64 on Windows 10 1809

Class: Elevation of Privilege

Summary:

The hardened VirtualBox process on a Windows host doesn't secure its COM interface leading to arbitrary code injection and EoP.

Description:

This issue is similar in scope to others I've reported such as S0867394/CVE-2017-10204. It allows you to call arbitrary code inside the hardened process which can expose the kernel drivers to normal user processes resulting in EoP. I'm assuming that this is still an issue you'd like to fix?

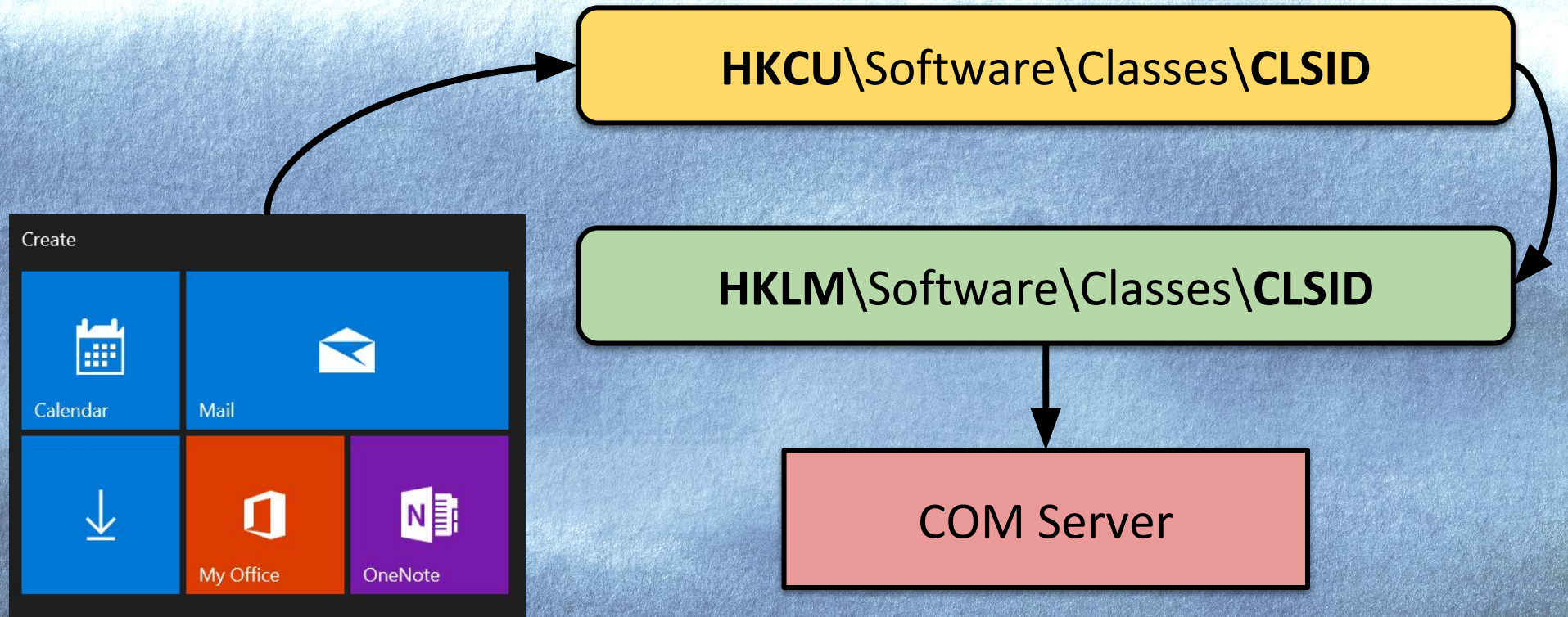


DEMO
TIME



Persistence Tricks

Class Lookup



ENTERPRISE ▼

TECHNIQUES

All

Initial Access

Execution

Persistence

- .bash_profile and .bashrc

Accessibility Features

Account Manipulation

AppCert DLLs

Applnit DLLs

[Home](#) > [Techniques](#) > [Enterprise](#) > Component Object Model Hijacking

Component Object Model Hijacking

The ^[1] (COM) is a system within Windows to enable interaction between software components through the operating system. ^[1] Adversaries can use this system to insert malicious code that can be executed in place of legitimate software through hijacking the COM references and relationships as a means for persistence. Hijacking a COM object requires a change in the Windows Registry to replace a reference to a legitimate system component which may cause that component to not work when executed. When that system component is executed through normal system operation the adversary's code will be executed instead. ^[2] An adversary is likely to hijack objects that are used frequently enough to maintain a consistent level of persistence, but are unlikely to break noticeable functionality within the system as to avoid system instability that could lead to detection.

<https://attack.mitre.org/techniques/T1122/>

What Else is in HKCU Classes?

Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help

Time of...	Process Name	PID	Operation	Path
5:31:59...	Explorer.EXE	5984	RegQueryValue	HKCU\Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\MuiCache\C:\Windows\sys...
5:31:59...	Explorer.EXE	5984	RegOpenKey	HKCU\Software\Classes\Interface\{85CB6900-4D95-11CF-960C-0080C7F4EE85}\Forward
5:31:59...	Explorer.EXE	5984	RegCloseKey	HKCU\Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\MuiCache
5:31:59...	Explorer.EXE	5984	RegOpenKey	HKCU\Software\Classes\Folder\ProgId
5:31:59...	Explorer.EXE	5984	RegOpenKey	HKCU\Software\Classes\Interface\{85CB6900-4D95-11CF-960C-0080C7F4EE85}\TypeLib
5:31:59...	Explorer.EXE	5984	RegOpenKey	HKCU\Software\Classes\Folder\shell\opennewwindow
5:31:59...	Explorer.EXE	5984	RegOpenKey	HKCU\Software\Classes\Interface\{85CB6900-4D95-11CF-960C-0080C7F4EE85}\TypeLib
5:31:59...	Explorer.EXE	5984	RegOpenKey	HKCU\Software\Classes\Interface\{85CB6900-4D95-11CF-960C-0080C7F4EE85}\TypeLib
5:31:59...	Explorer.EXE	5984	RegOpenKey	HKCU\Software\Classes\TypeLib\{EAB22AC0-30C1-11CF-A7EB-0000C05BAE0B}
5:31:59...	Explorer.EXE	5984	RegOpenKey	HKCU\Software\Classes\TypeLib\{EAB22AC0-30C1-11CF-A7EB-0000C05BAE0B}
5:31:59...	Explorer.EXE	5984	RegOpenKey	HKCU\Software\Classes\TypeLib\{EAB22AC0-30C1-11CF-A7EB-0000C05BAE0B}\1.1
5:31:59...	Explorer.EXE	5984	RegOpenKey	HKCU\Software\Classes\TypeLib\{EAB22AC0-30C1-11CF-A7EB-0000C05BAE0B}\1.1
5:31:59...	Explorer.EXE	5984	RegOpenKey	HKCU\Software\Classes\TypeLib\{EAB22AC0-30C1-11CF-A7EB-0000C05BAE0B}\1.1\0
5:31:59...	Explorer.EXE	5984	RegOpenKey	HKCU\Software\Classes\TypeLib\{EAB22AC0-30C1-11CF-A7EB-0000C05BAE0B}\1.1\0\win32
5:31:59...	Explorer.EXE	5984	RegOpenKey	HKCU\Software\Classes\TypeLib\{EAB22AC0-30C1-11CF-A7EB-0000C05BAE0B}\1.1\0\win32
5:31:59...	Explorer.EXE	5984	RegOpenKey	HKCU\Software\Classes\CLSID\{52205fd8-5dfb-447d-801a-d0b52f2e83e1}\shell\OpenNewWindow
5:31:59...	Explorer.EXE	5984	RegOpenKey	HKCU\Software\Classes\CLSID\{52205fd8-5dfb-447d-801a-d0b52f2e83e1}\shell\OpenNewWindow\cc
5:31:59...	Explorer.EXE	5984	RegOpenKey	HKCU\Software\Classes\CLSID\{52205fd8-5dfb-447d-801a-d0b52f2e83e1}\shell\OpenNewWindow\cc
5:31:59...	Explorer.EXE	5984	RegOpenKey	HKCU\Software\Classes\CLSID\{52205fd8-5dfb-447d-801a-d0b52f2e83e1}\shell\OpenNewWindow\cc
5:31:59...	Explorer.EXE	5984	RegOpenKey	HKCU\Software\Classes\CLSID\{52205fd8-5dfb-447d-801a-d0b52f2e83e1}\shell\OpenNewWindow\cc
5:31:59...	Explorer.EXE	5984	RegOpenKey	HKCU\Software\Classes\Interface\{85CB6900-4D95-11CF-960C-0080C7F4EE85}\Forward
5:31:59...	Explorer.EXE	5984	RegOpenKey	HKCU\Software\Classes\Interface\{85CB6900-4D95-11CF-960C-0080C7F4EE85}\TypeLib
5:31:59...	Explorer.EXE	5984	RegOpenKey	HKCU\Software\Classes\Interface\{85CB6900-4D95-11CF-960C-0080C7F4EE85}\TypeLib
5:31:59...	Explorer.EXE	5984	RegOpenKey	HKCU\Software\Classes\Interface\{85CB6900-4D95-11CF-960C-0080C7F4EE85}\TypeLib

Showing 3,879 of 412,023 events (0.94%) Backed by virtual memory

Type Library
Resolving


Loading Type Libraries

LoadTypeLib function

Loads and registers a type library.

C++

```
HRESULT LoadTypeLib(  
    LPCOLESTR szFile,  
    ITypeLib **pptlib  
);
```



1. If the file is a stand-alone type library it's loaded directly.


Loading Type Libraries

LoadTypeLib function

Loads and registers a type library.

C++

```
HRESULT LoadTypeLib(  
    LPCOLESTR szFile,  
    ITypeLib **pptlib  
);
```



1. If the file is a stand-alone type library it's loaded directly.
2. If the file is a DLL or an executable file, it is loaded from a resource.


Loading Type Libraries

LoadTypeLib function

Loads and registers a type library.

C++

```
HRESULT LoadTypeLib(  
    LPCOLESTR szFile,  
    ITypeLib **pptlib  
);
```



1. If the file is a stand-alone type library it's loaded directly.
2. If the file is a DLL or an executable file, it is loaded from a resource.
3. **Otherwise parse as a moniker.**

Scriptlet Monikers



Matt Nelson

@enigma0x3

Scriptlet execution in Excel via the script moniker and a hyperlink. No user warning/pop-up pre-April patch release:
gist.githubusercontent.com/enigma0x3/22ab

...

12:16 PM - 1 May 2017

104 Retweets 165 Likes





DEMO
TIME

Wrapup

- The key to finding *GOOD* bugs is *GOOD* tooling
- Complexity and backwards compatibility are still an enemy
 - Cross-session exploitation still in whack-a-mole mode
 - Desktop Broker means all LOB apps effectively Full Trust
 - COM hijacking continues
- Still plenty of things to go looking for:
 - 50+ Cross-Session objects capable objects
 - Loads of trusted marshalers

A red fox is captured in mid-jump in a snowy landscape. The fox has orange-red fur on its back and sides, with white fur on its chest and underbelly. Its tail is bushy and tipped with white. The fox is looking down and to the right. The background is a vast, flat expanse of snow with some small, dry, brown plants poking through.

Happy Hunting

COM BUGS