Having FUN* with COM

James Forshaw (@tiraniddo)

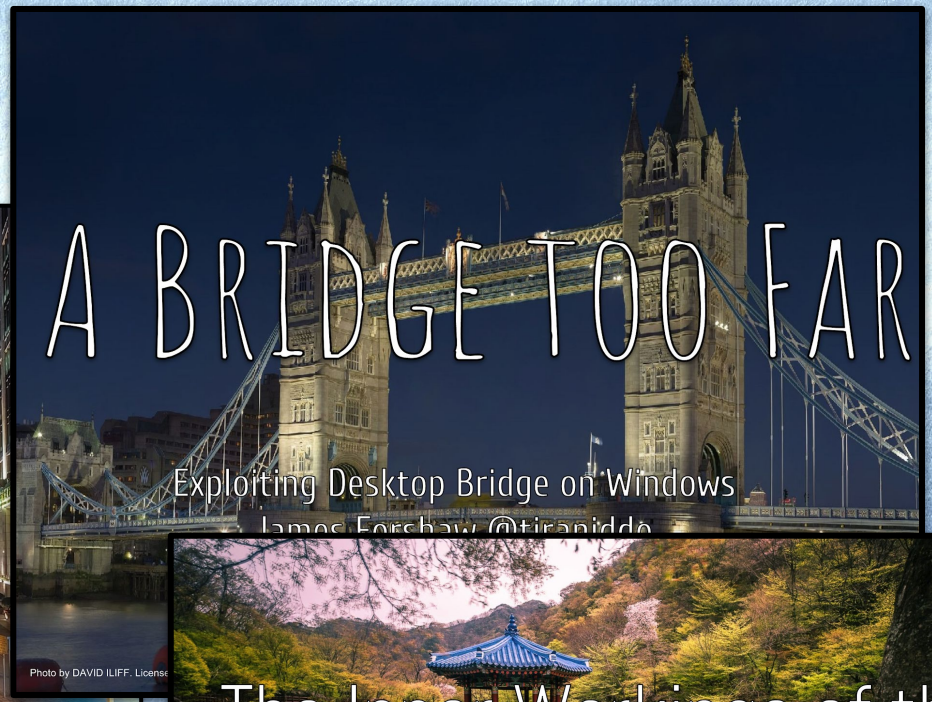
Infiltrate 2019

* FUN assumed but not guaranteed.

https://pixabay.com/photos/hot-air-balloons-adventure-balloons-1867279/

Background

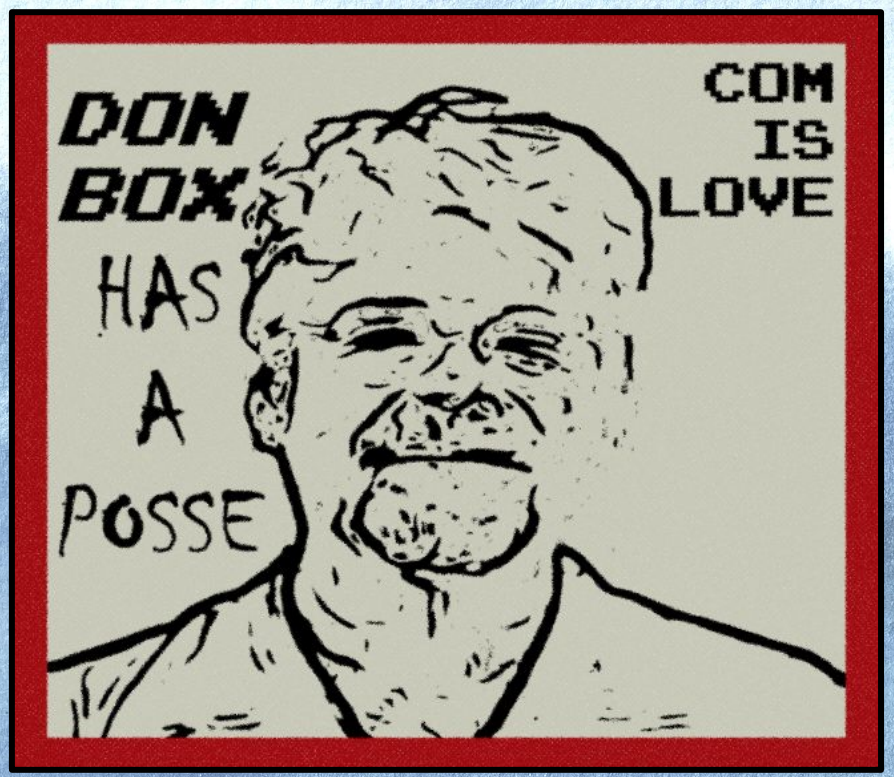

COM IN
60
SECONDS
James Forshaw (@tiraniddo)
Infiltrate 2017
https://pixabay.com/en/hong-kong-city-urban-skyscrapers-1990268/

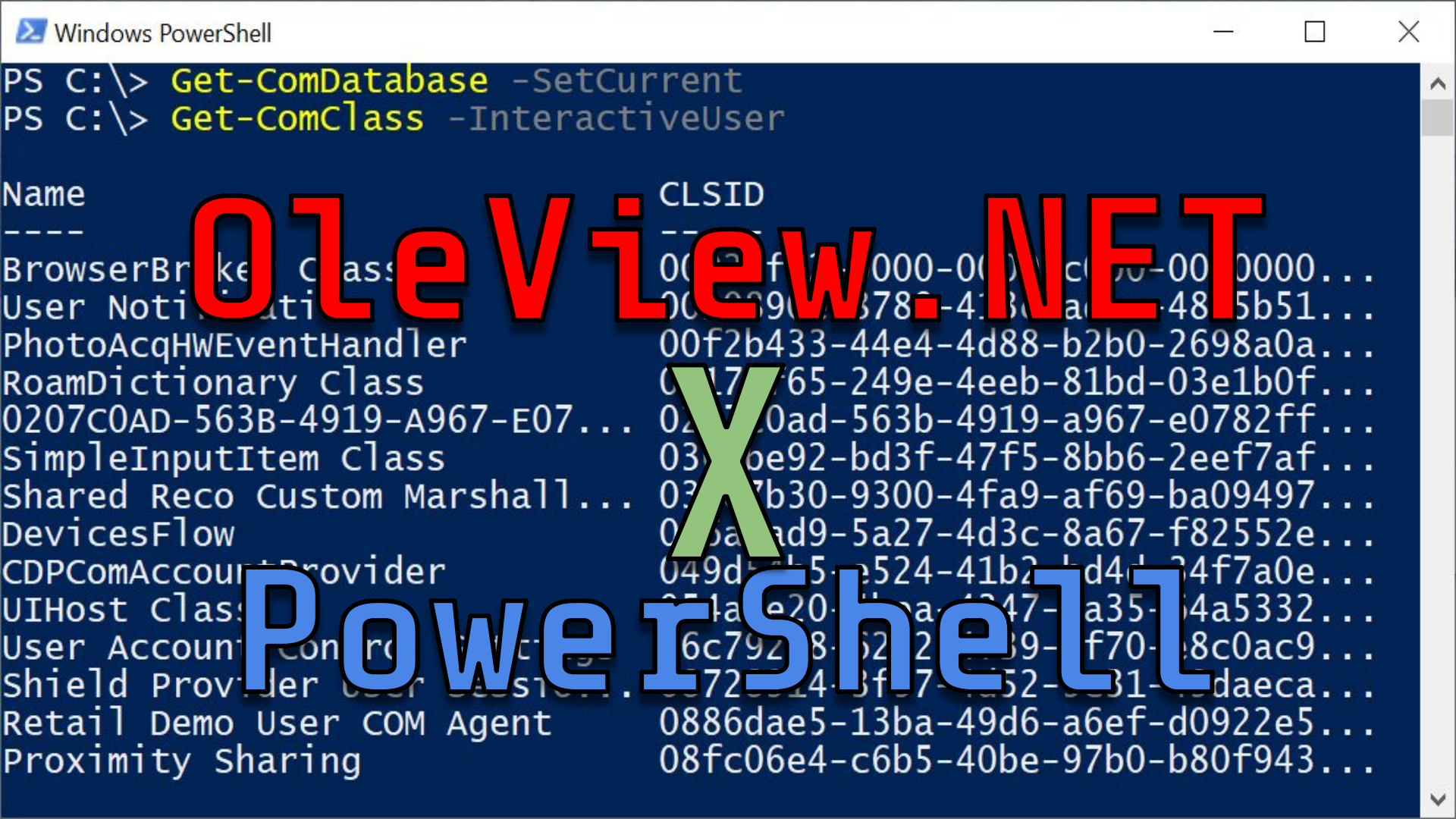

A BRIDGE TOO FAR
Exploiting Desktop Bridge on Windows
James Forshaw @tiraniddo
Photo by DAVID ILIFF. License


The Inner Workings of the Windows Runtime
James Forshaw @tiraniddo
https://www.flickr.com/photos/giuseppemilo/40760404654

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
User Notification
PhotoAcqHWEventHandler
RoamDictionary Class
0207C0AD-563B-4919-A967-E07...
SimpleInputItem Class
Shared Reco Custom Marshall...
DevicesFlow
CDPComAccountProvider
UIHost Class
User Account Control
Shield Provider
Retail Demo User COM Agent
Proximity Sharing
0886dae5-13ba-49d6-a6ef-d0922e5...
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.

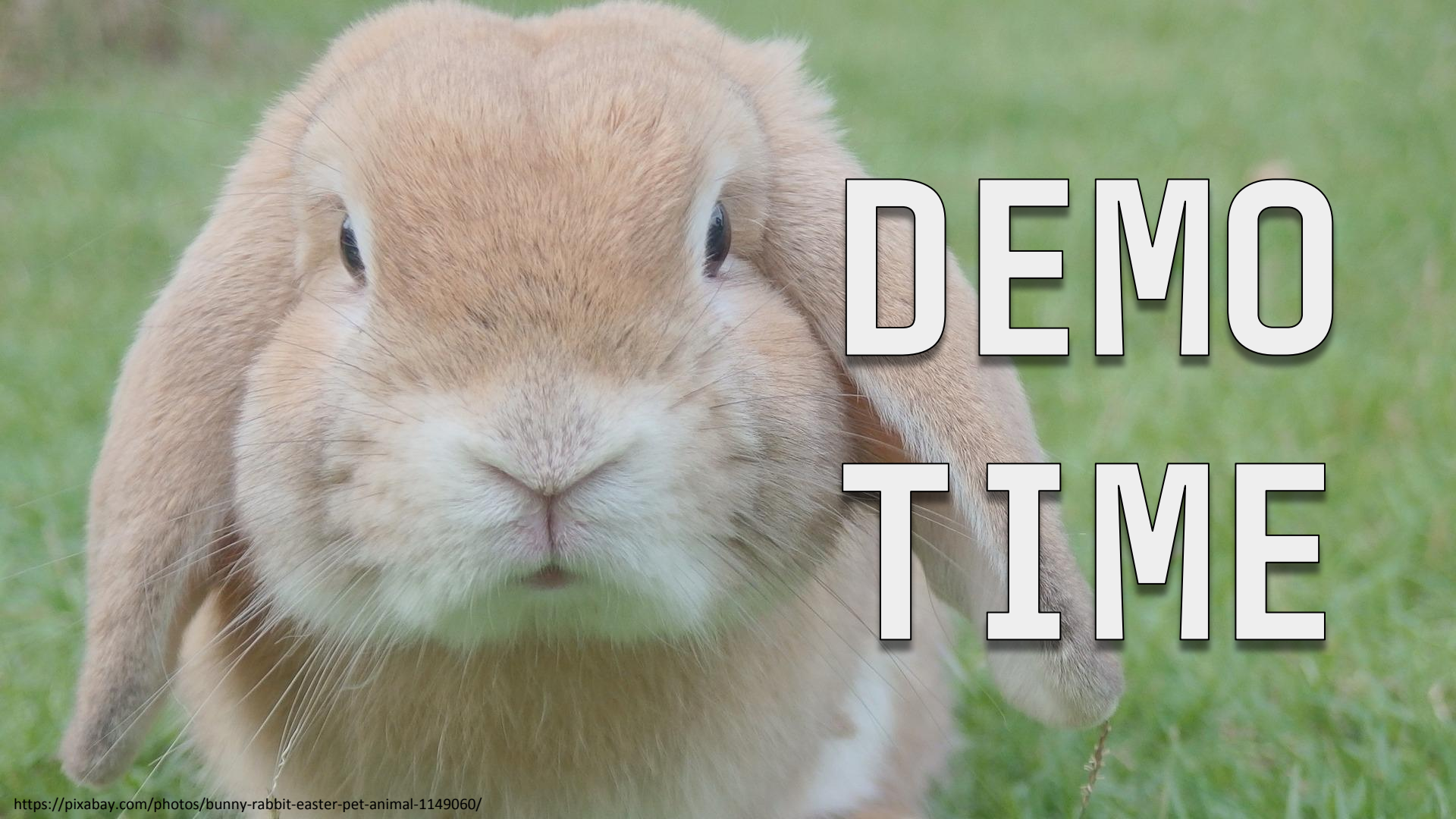
```
PS> Get-ComClass
```

Get all COM classes which have a registered local server.

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

Enumerate all accessible interfaces for a class.

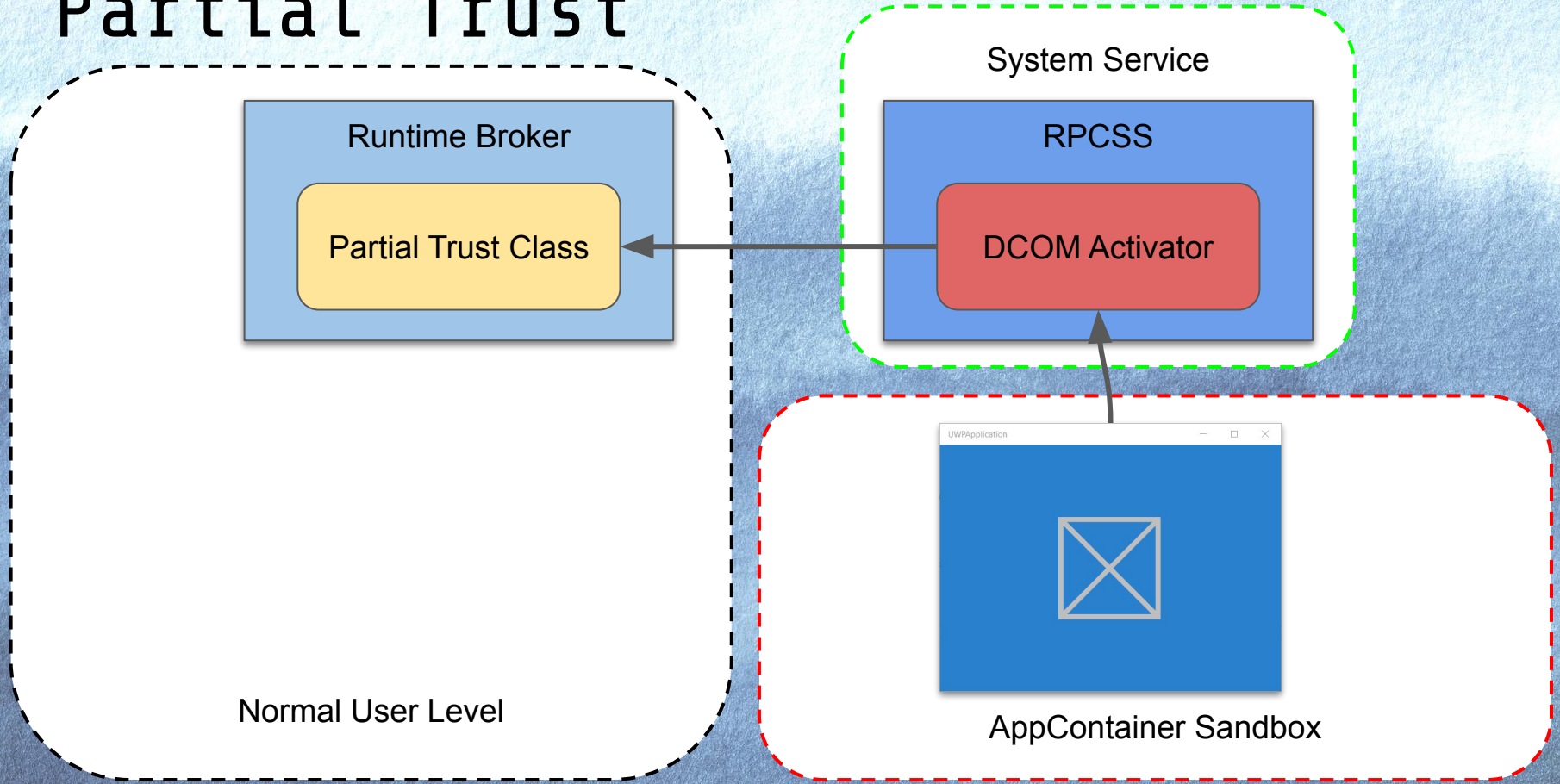
```
PS> $intfs = Get-ComClassInterface $cls
```



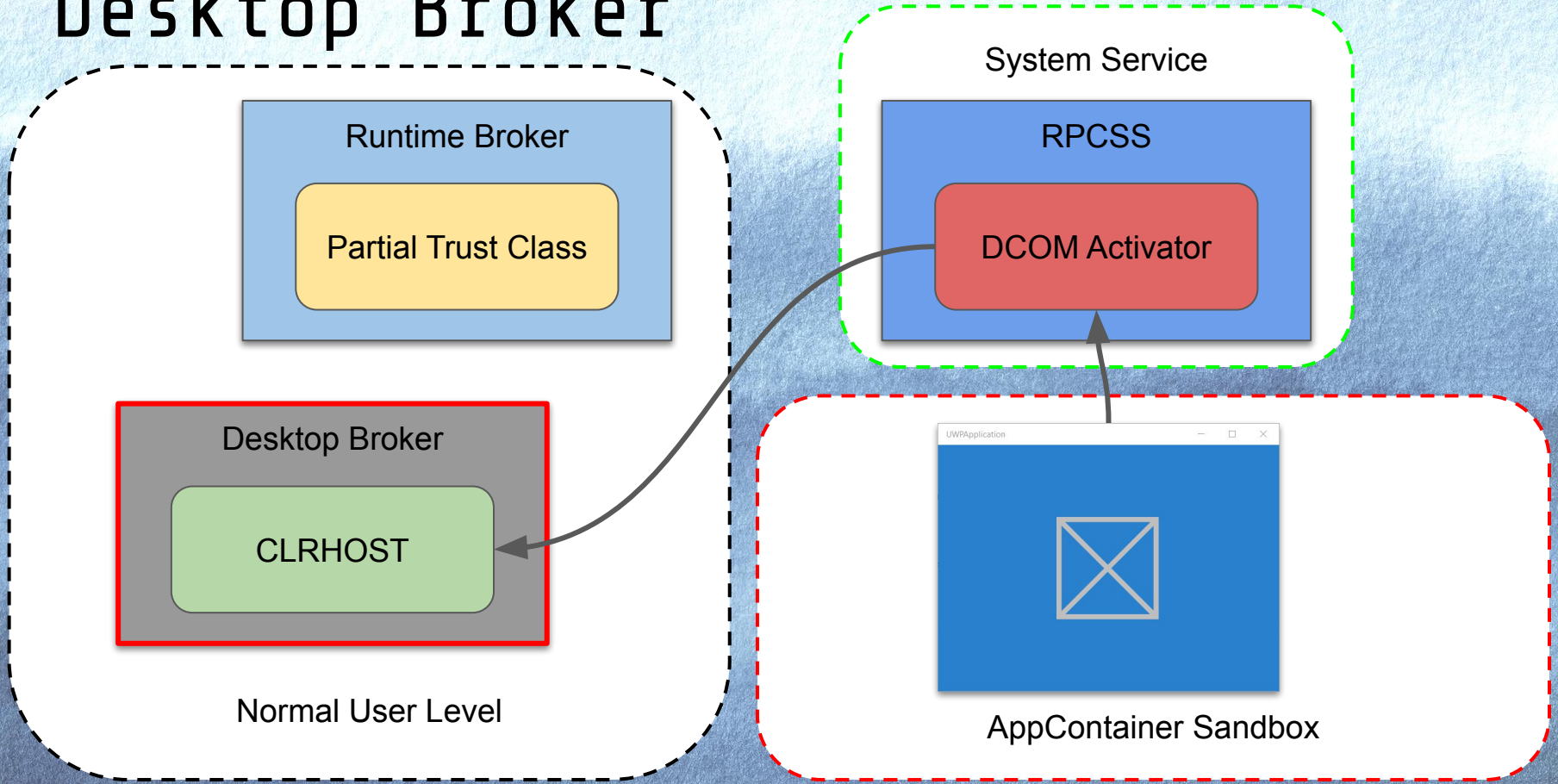
https://pixabay.com/photos/bunny-rabbit-easter-pet-animal-1149060/

https://pixabay.com/photos/sandbox-children-child-sand-play-1583289/

Partial Trust



Desktop Broker



Brokered Windows Runtime Components for a side-loaded UWP app

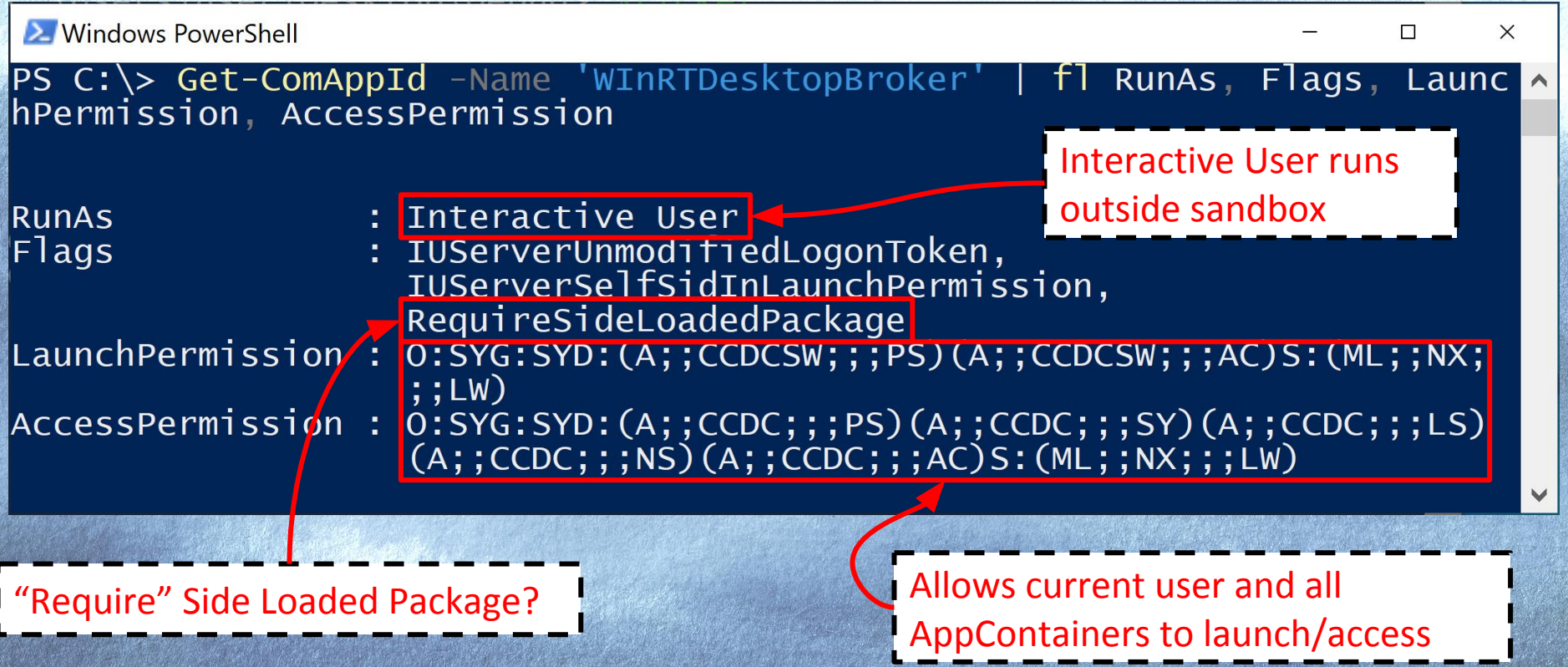
This article discusses an enterprise-targeted feature supported by Windows 10, which allows touch-friendly .NET apps to use the existing code responsible for key business-critical operations. https://docs.microsoft.com/en-us/windows/uwp/winrt-components/brokered-windows-runtime-components-for-side-loaded-windows-store-apps

```
<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



Side Loaded Package Launch Check

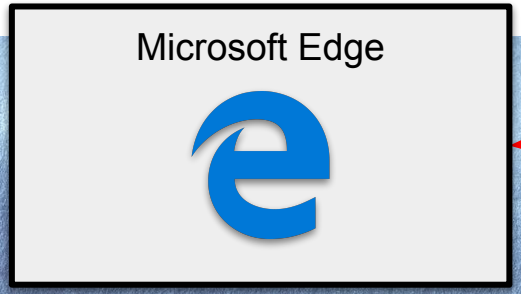
```
void IsSideLoadedPackage(LPCWSTR PackageName,
                         bool *IsSideloaded) {
  PackageOrigin origin;
  *IsSideloaded = false;
  GetStagedPackageOrigin(package_name, &origin);

  *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

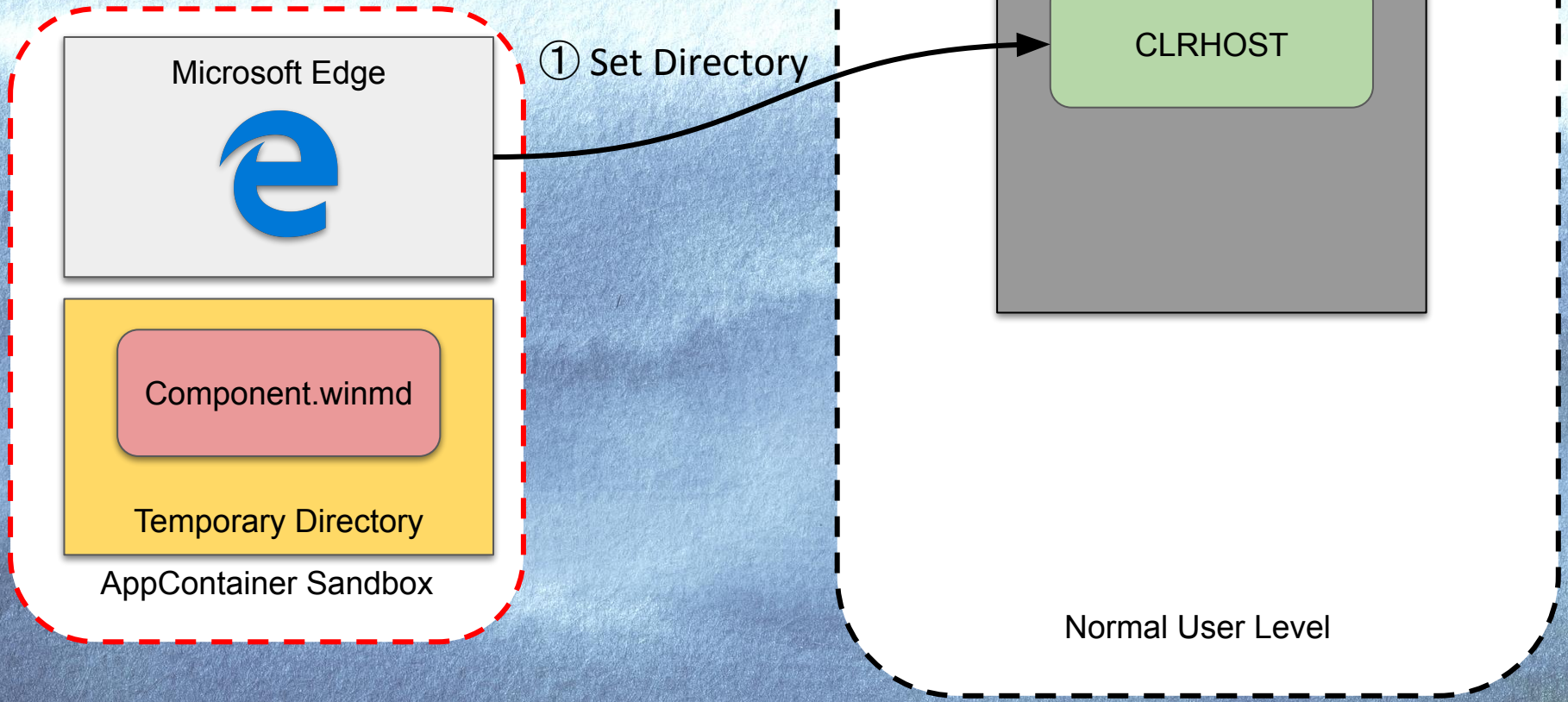
```
interface IWinRTDesktopBroker : IUnknown {
    HRESULT GetClassActivatorForApplication(
        HSTRING dir, IWinRTClassActivator** ppv);
};
```

Specify arbitrary directory

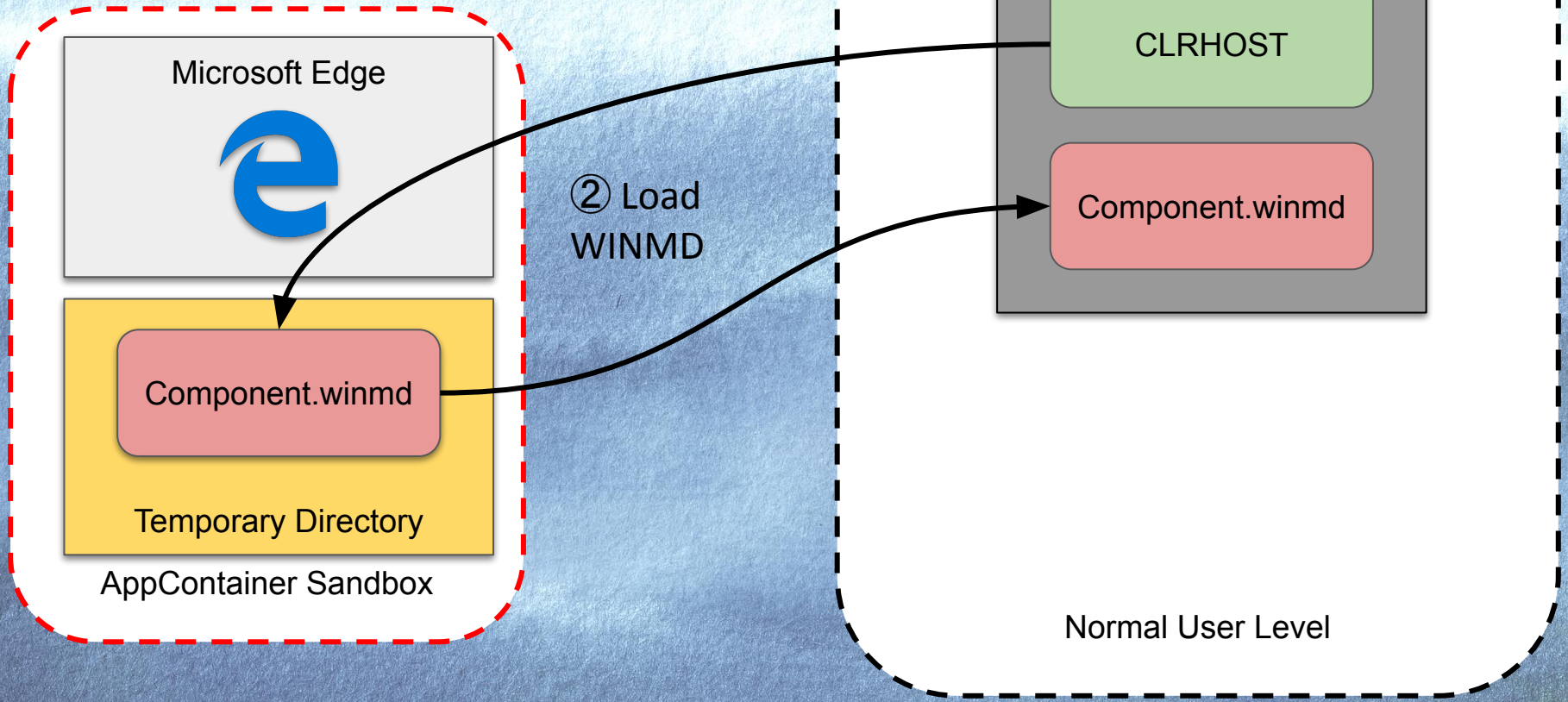
```
interface IWinRTClassActivator : IUnknown {
    HRESULT ActivateInstance(HSTRING activatableClassId,
                             IInspectable** ppv);
    HRESULT GetActivationFactory(HSTRING activatableClassId,
                                 REFIID riid, IUnknown** ppv);
}
```

Also a .NET COM object!

Exploitation



Exploitation



Exploitation

Microsoft Edge

Component.winmd

Temporary Directory

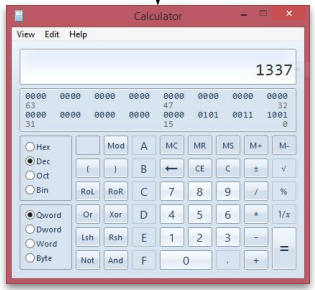
AppContainer Sandbox

③ Execute Component

Desktop Broker

CLRHOST

Component.winmd



Normal User Level

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;
}
```

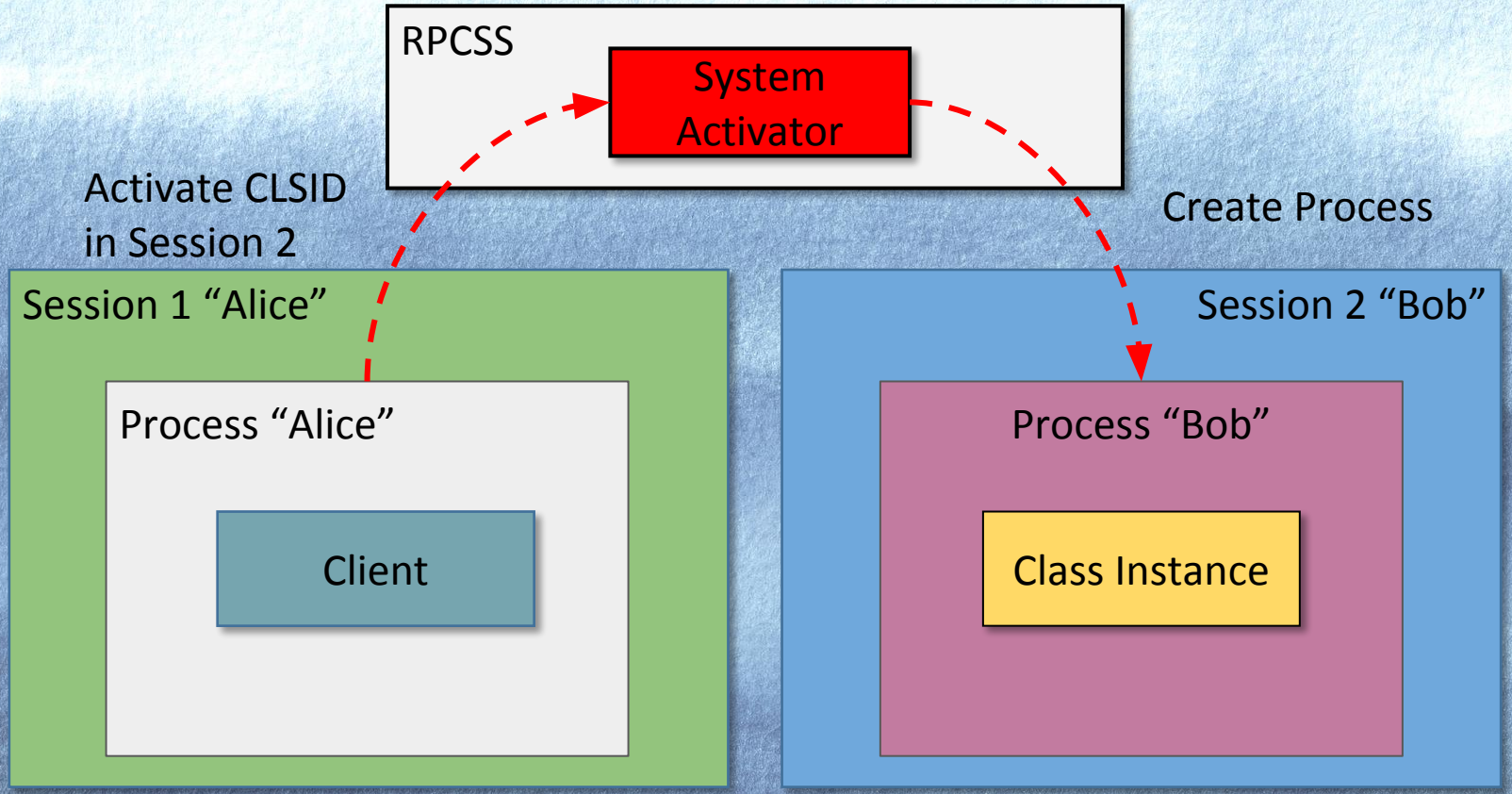
Only allow Side Loaded applications

https://bugs.chromium.org/p/project-zero/issues/detail?id=1688

Cross-Session Attacks

https://pixabay.com/photos/rose-blue-flower-bloom-romance-165819/

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.

https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2017-0100

Not Really!

Issue 1224: Windows: Bad Fix for COM Session Moniker EoP

Code ‹ Prev 90 of 139 Next › Back to list

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

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.

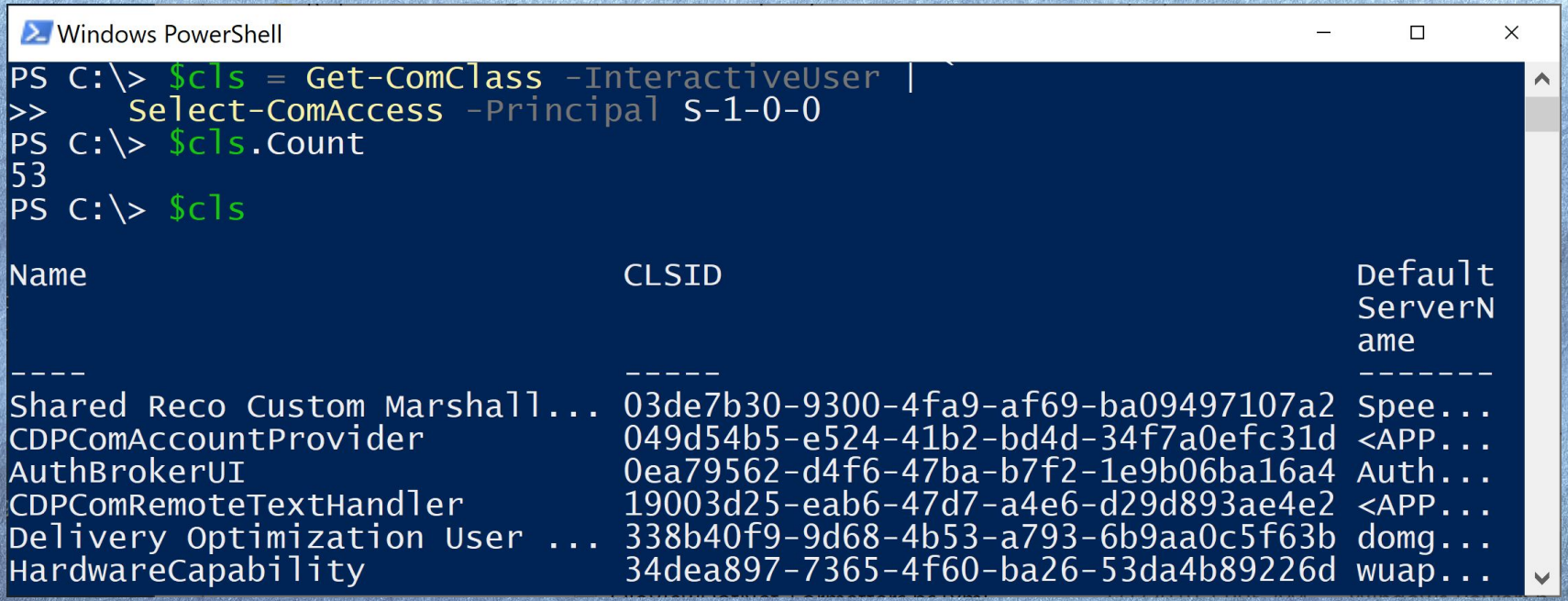
```
if ( imp_token_il >= process_token_il
  && (imp_token_il >= SECURITY_MANDATORY_HIGH_RID
   || EqualSid(process_token_user, imp_token_user)))
{
  ShellExecuteW(NULL, L"open", path, NULL, NULL, SW_SHOW);
}
```

Integrity levels are NOT a security boundary enforcement mechanism.

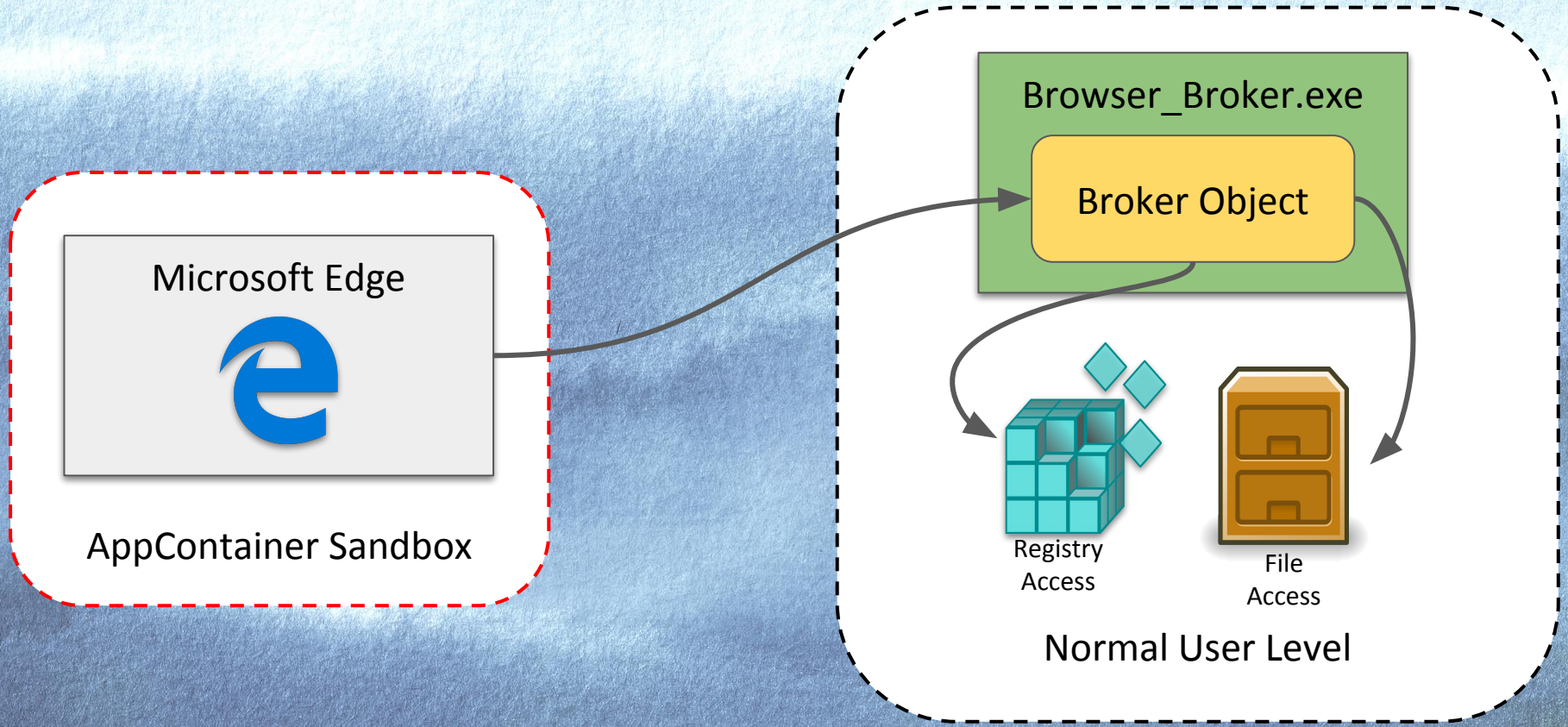
https://bugs.chromium.org/p/project-zero/issues/detail?id=1224

Find Potential Cross-Session Objects

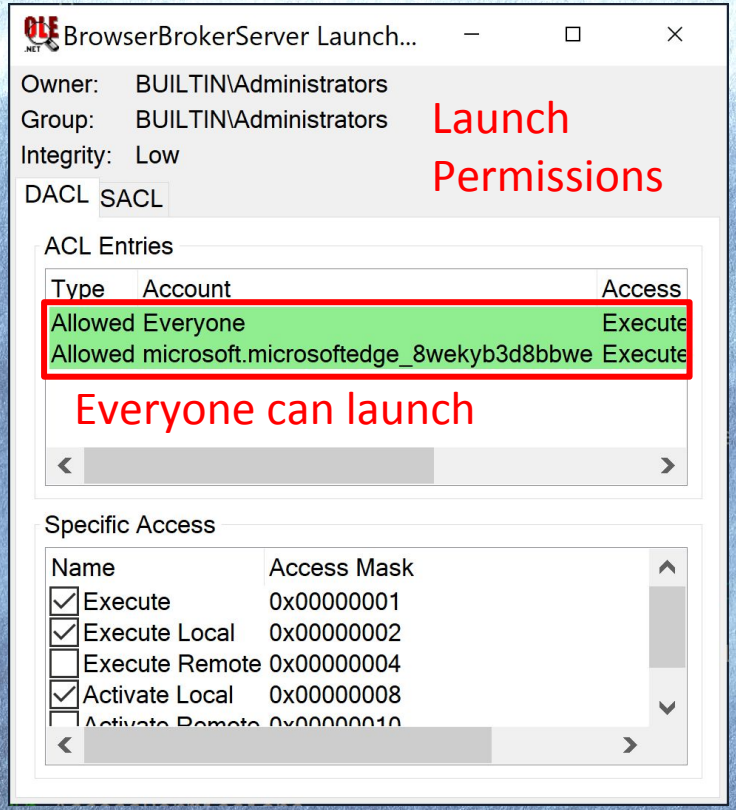
```
PS> Get-ComClass -InteractiveUser | `
    Select-ComAccess -Principal S-1-0-0
```

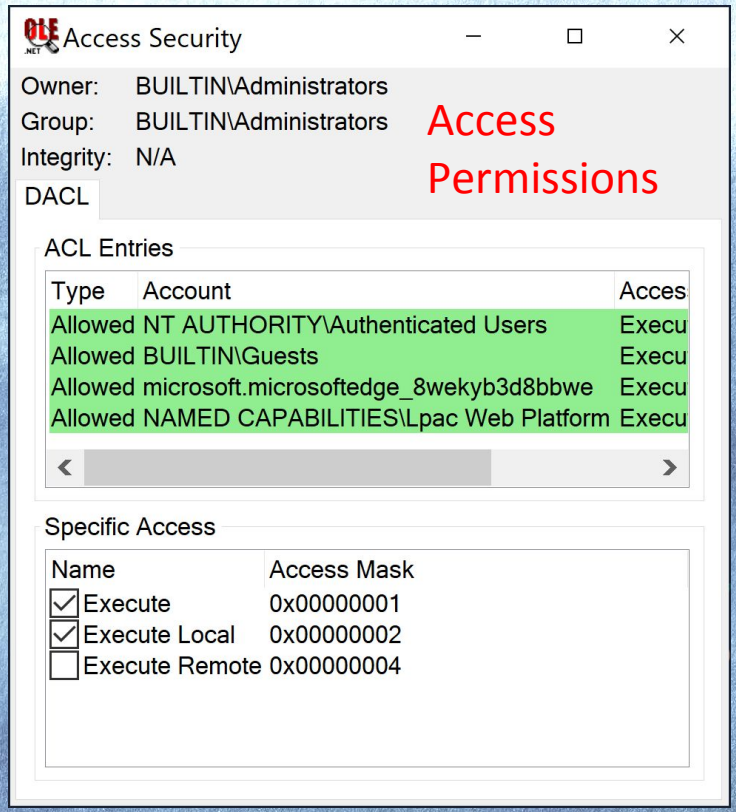


Browser Broker



Browser Broker Security





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\cleviewdotnet\1.7\OleV
iewDotNet.psm1:1448 char:17
+ ...                 $obj = $Class.Cr
$Remo ...
+                     ~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotS
   ion
    + FullyQualifiedErrorId : COMB

PS C:\>
```

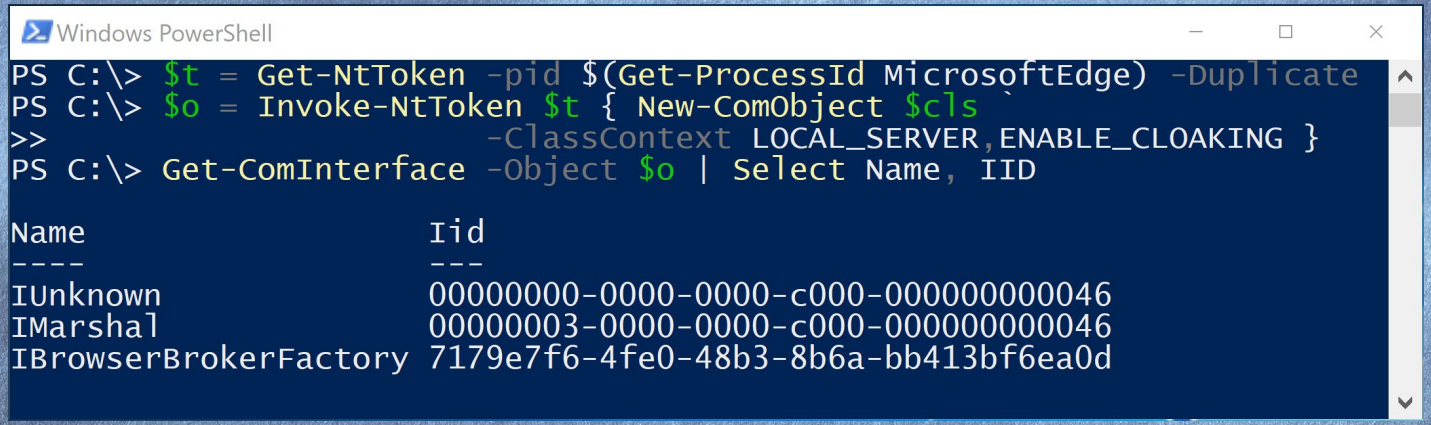
```
HRESULT BrokerAuthenticateCOMCaller() {
  HANDLE TokenHandle;
  LPWSTR FamilyName;

  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

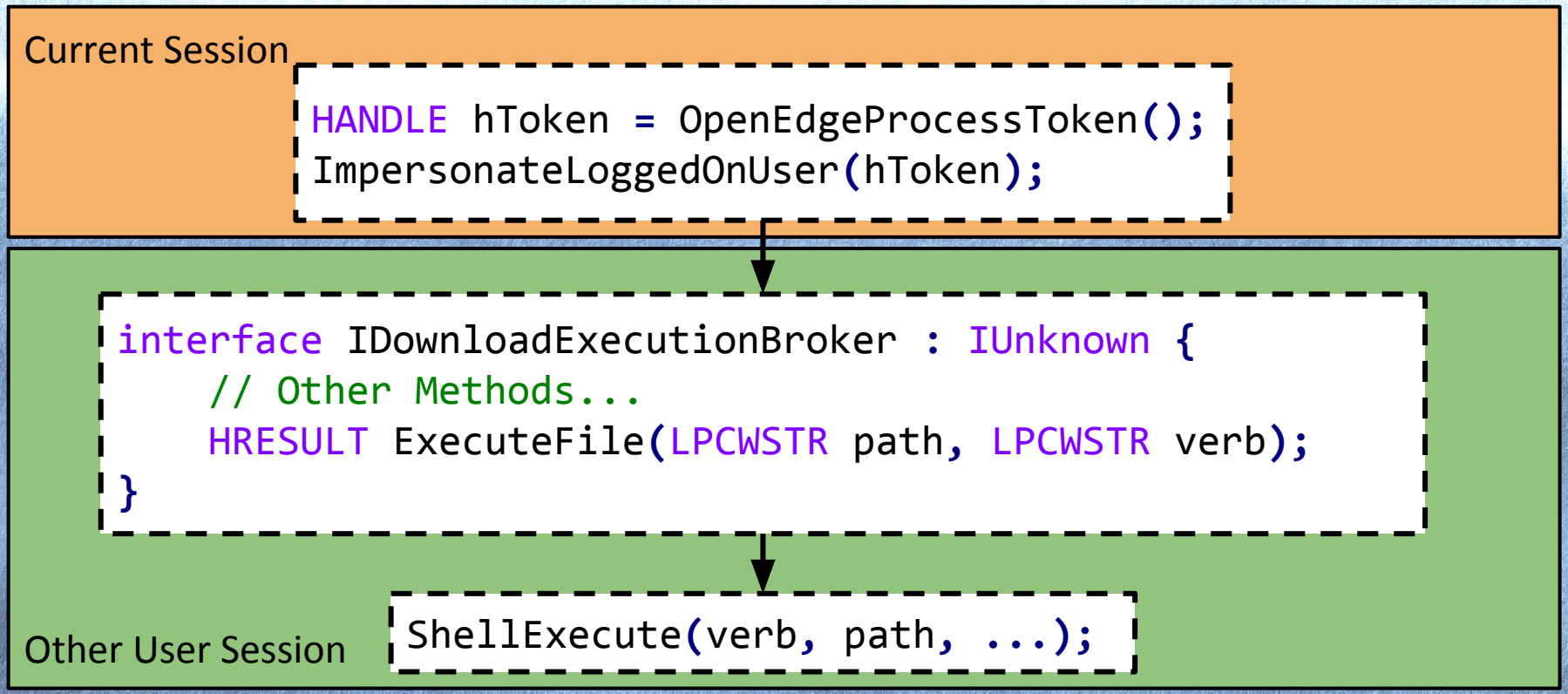
```
PS> $token = Get-NtToken -Duplicate `
      -pid  $(Get-Process MicrosoftEdge).Id
```

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



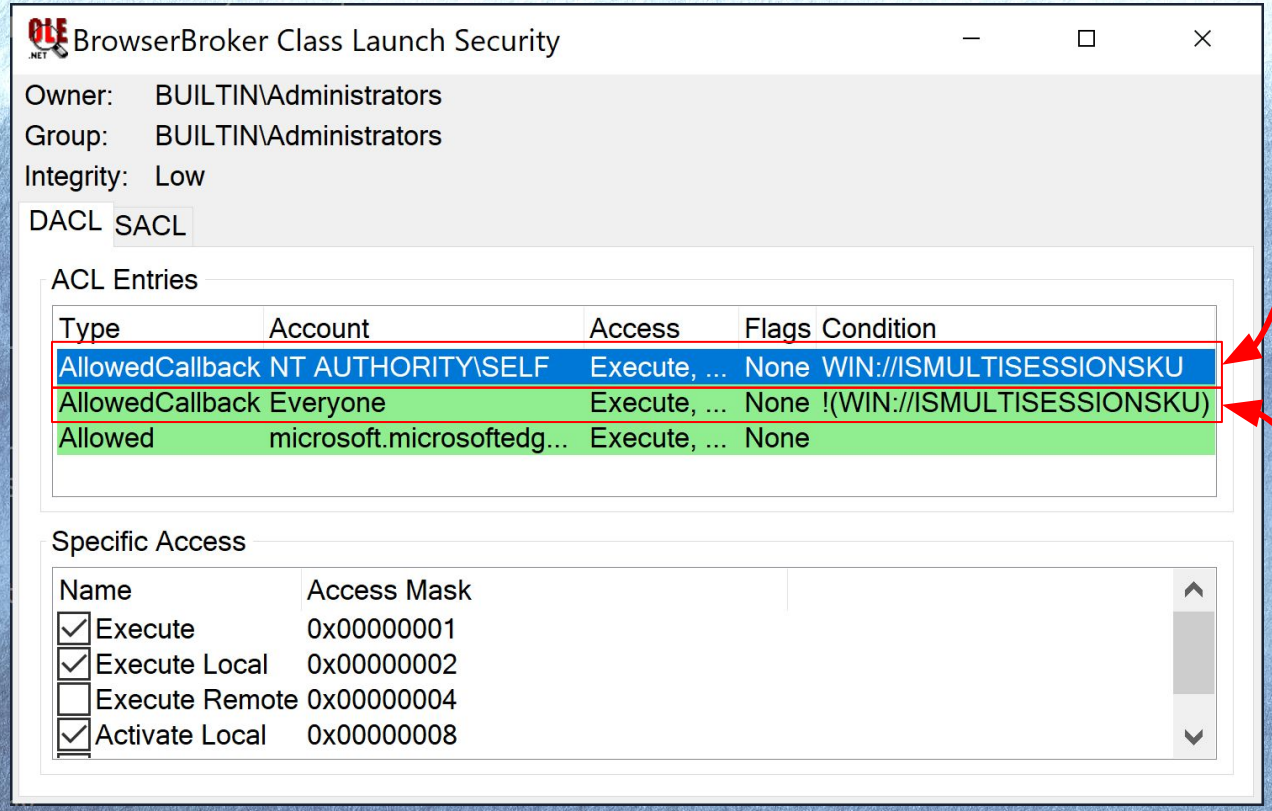
Create with impersonation token

Chaining to ShellExecute



https://bugs.chromium.org/p/project-zero/issues/detail?id=1683

Fixed as CVE-2019-0566



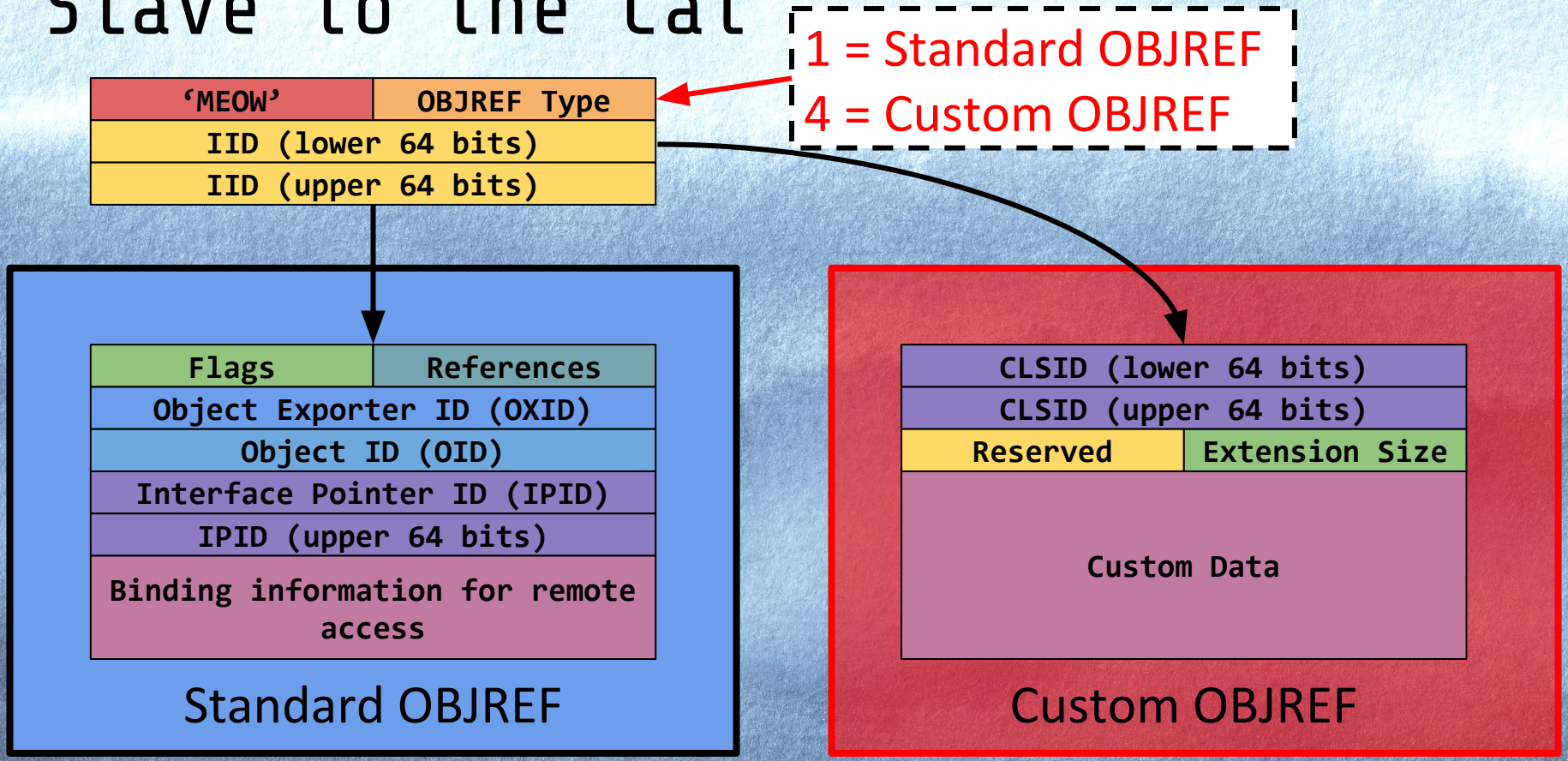
Added “SELF” SID access on multi-session SKUs.

Left Everyone group for non-multi-session SKUs

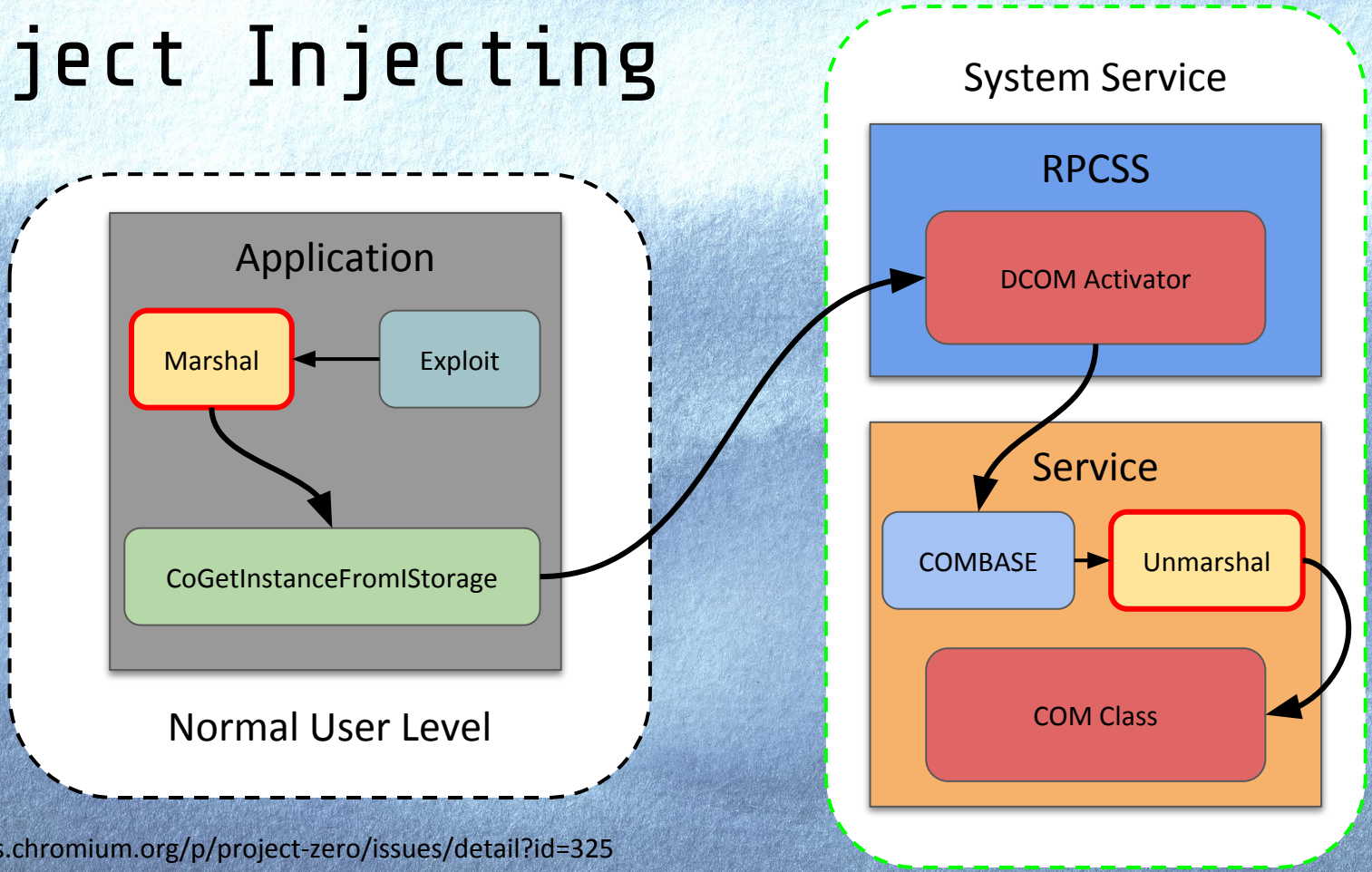
https://bugs.chromium.org/p/project-zero/issues/detail?id=1683

https://pixabay.com/photos/railway-rails-track-train-2439189/

Slave to the Cat



Object Injecting



https://bugs.chromium.org/p/project-zero/issues/detail?id=325

“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

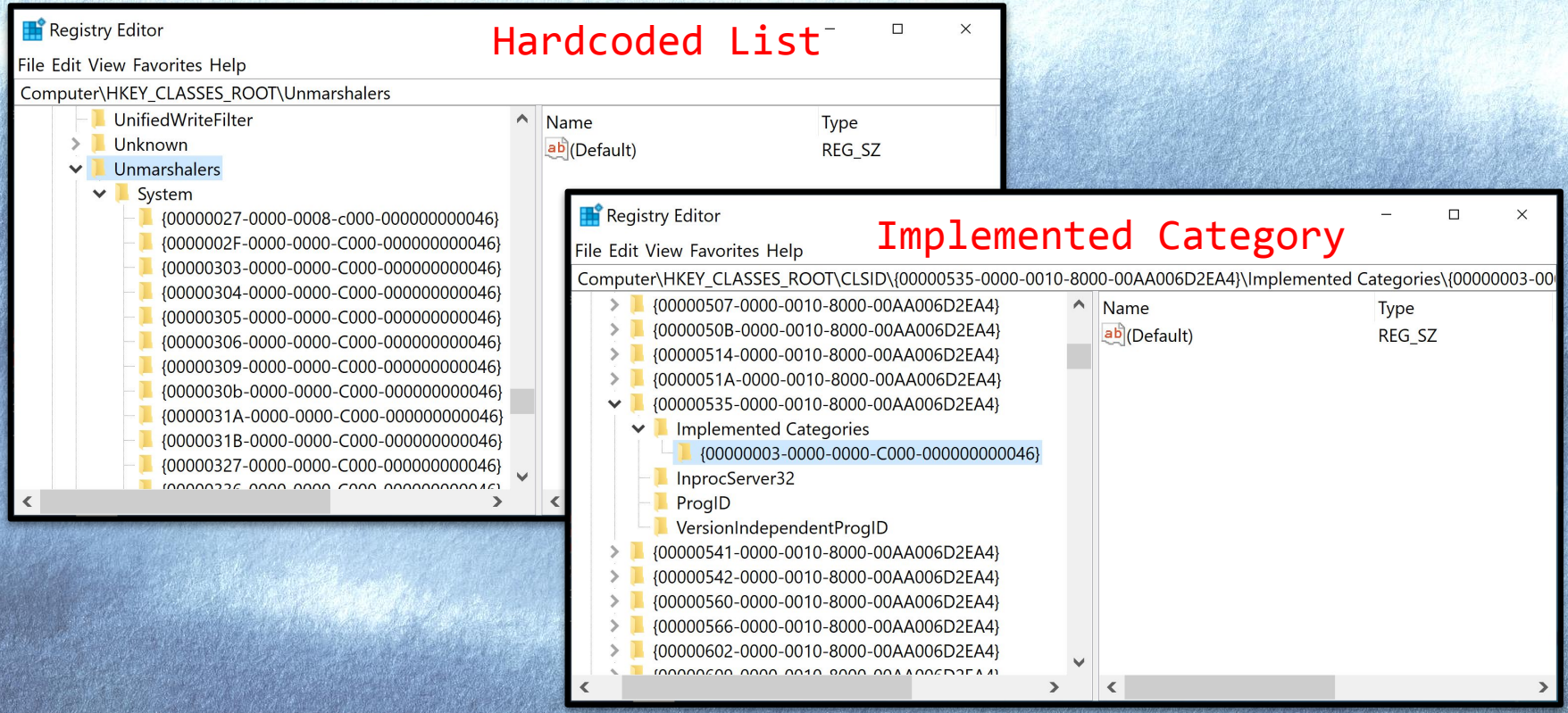
Restriction on Custom Marshaling

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, ComSvcs.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 unmarshalers and unmarshalers allowed per-process by the CoAllowUnmarshalerCLSID 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 unmarshalers and unmarshalers allowed per-process by the CoAllowUnmarshalerCLSID function. Unmarshaling behavior for data with a source that's not app container is unchanged from previous versions.

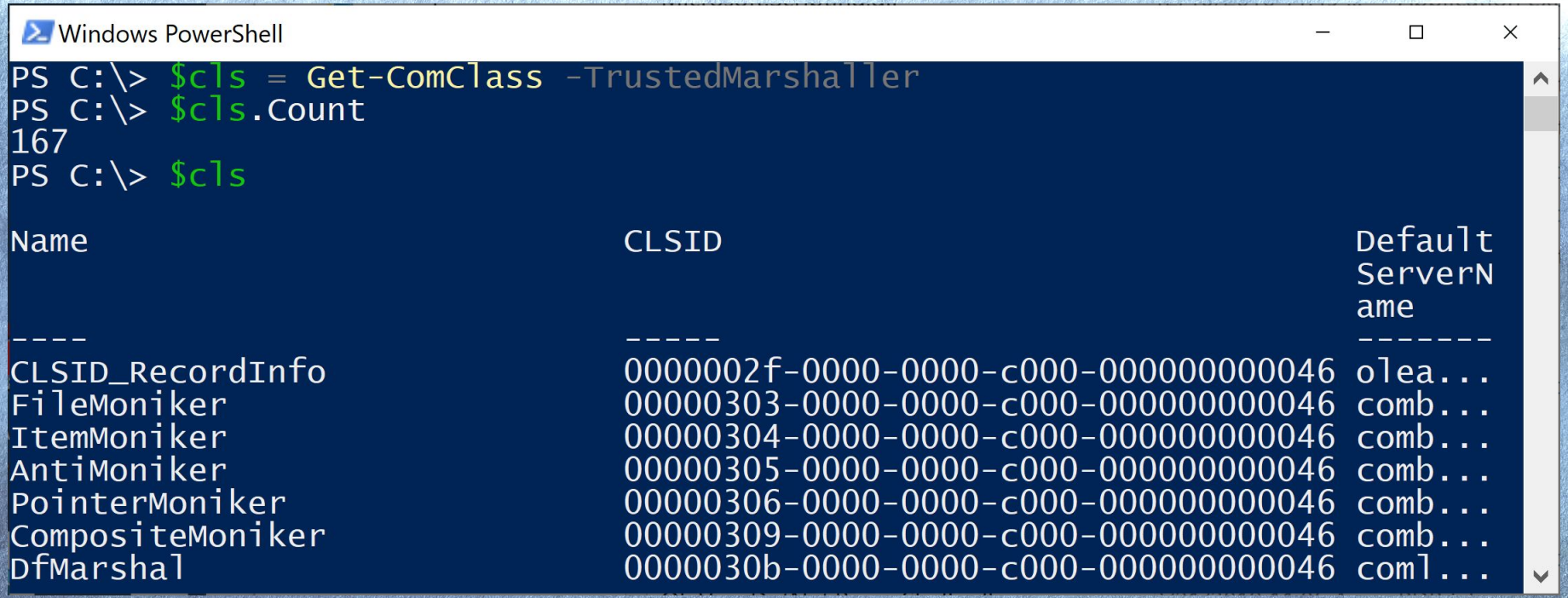
Bypassed if a “System Trusted” marshaler is used.

System Trusted Marshalers

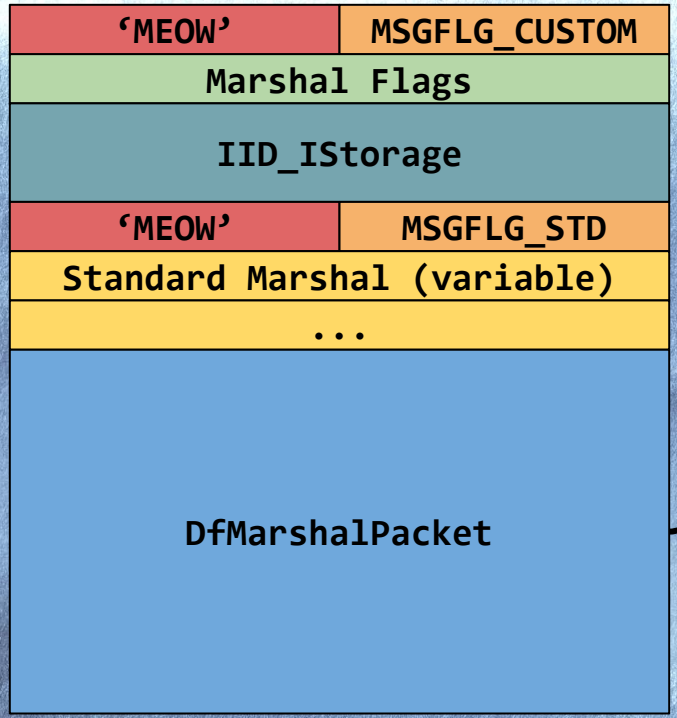


Find Trusted Marshallers

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



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

Issue Number	*Description*
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

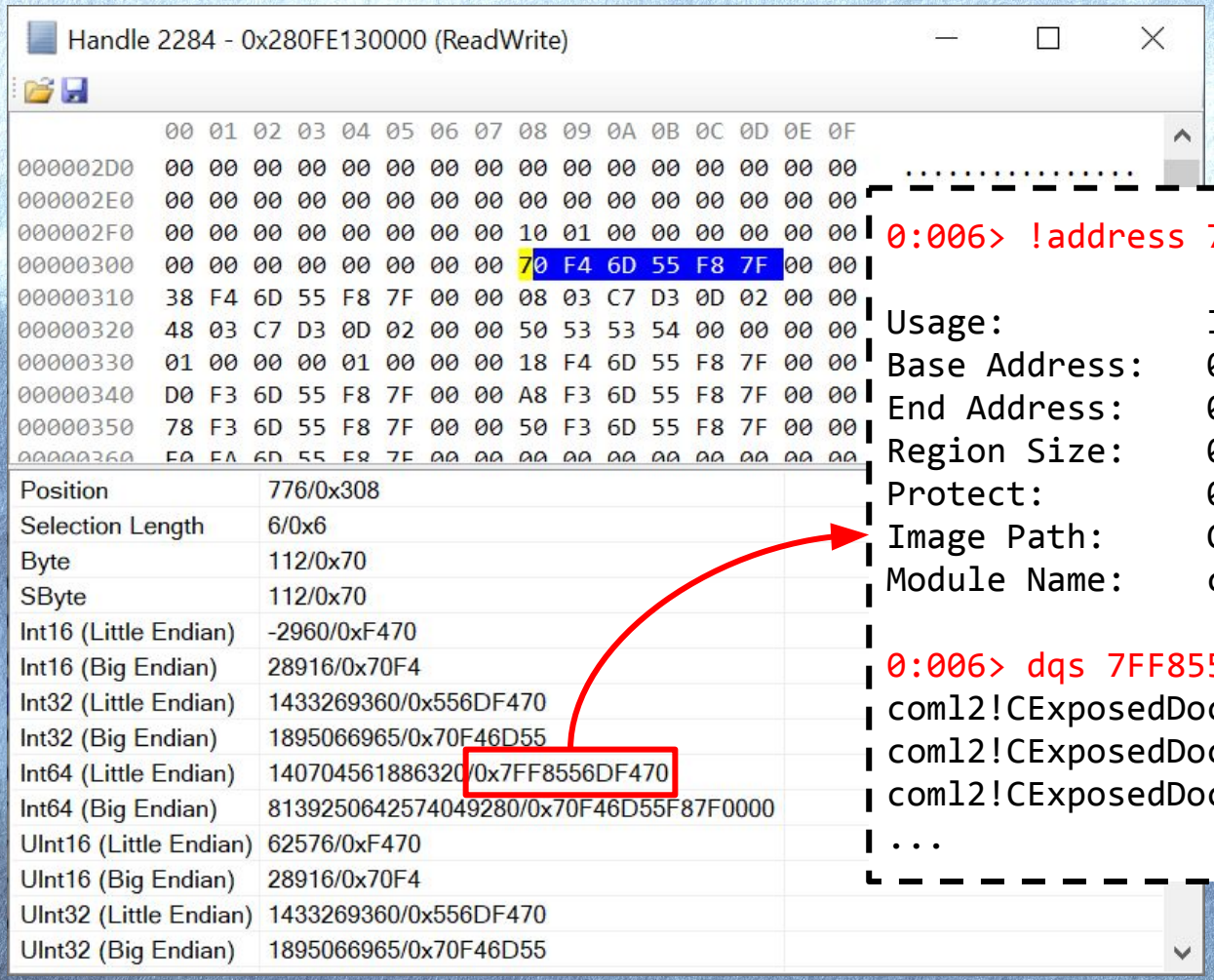
Shared Allocator (Issue 1646)

```
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;
  // ...
}
```

https://bugs.chromium.org/p/project-zero/issues/detail?id=1646



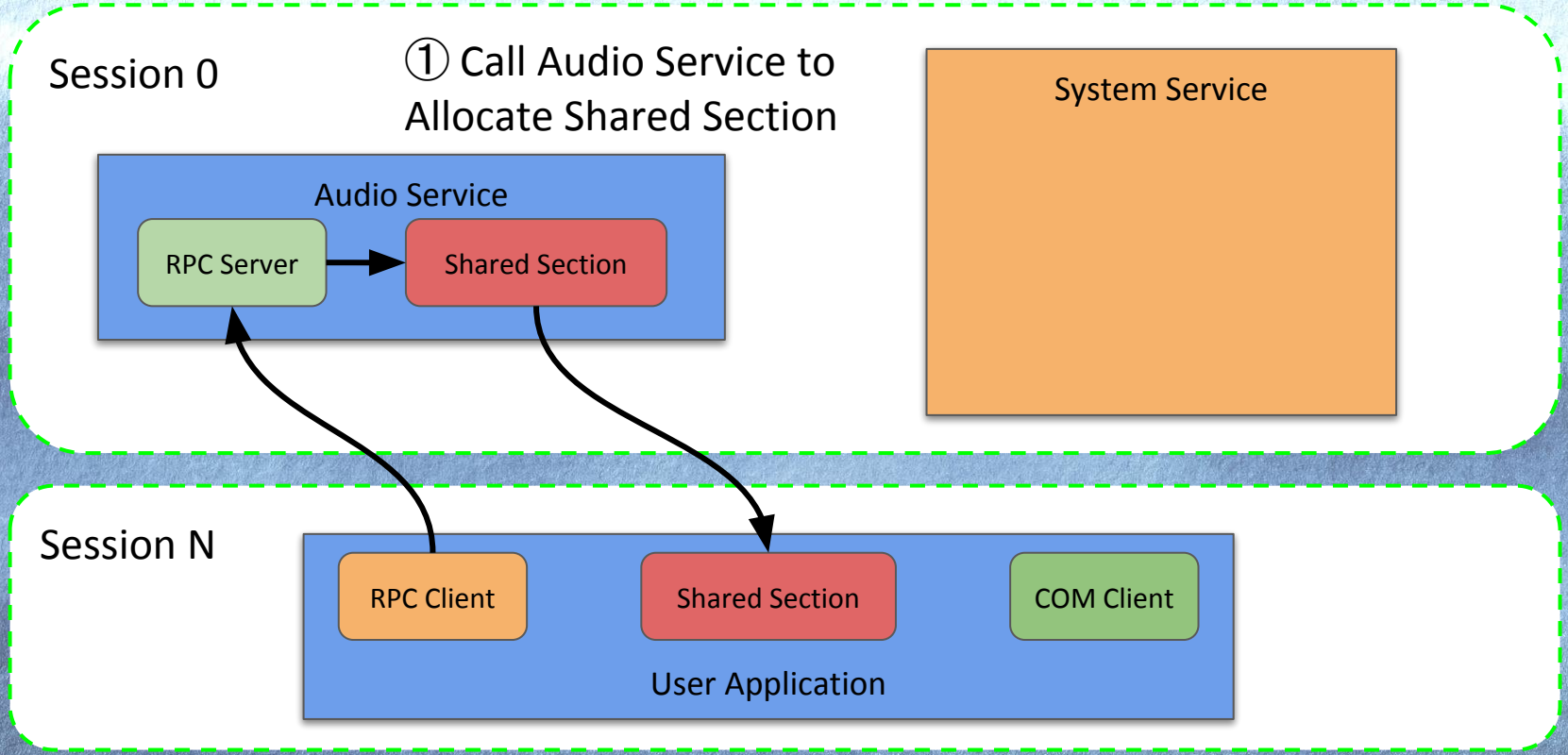

```
0:006> !address 7FF8556DF470

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

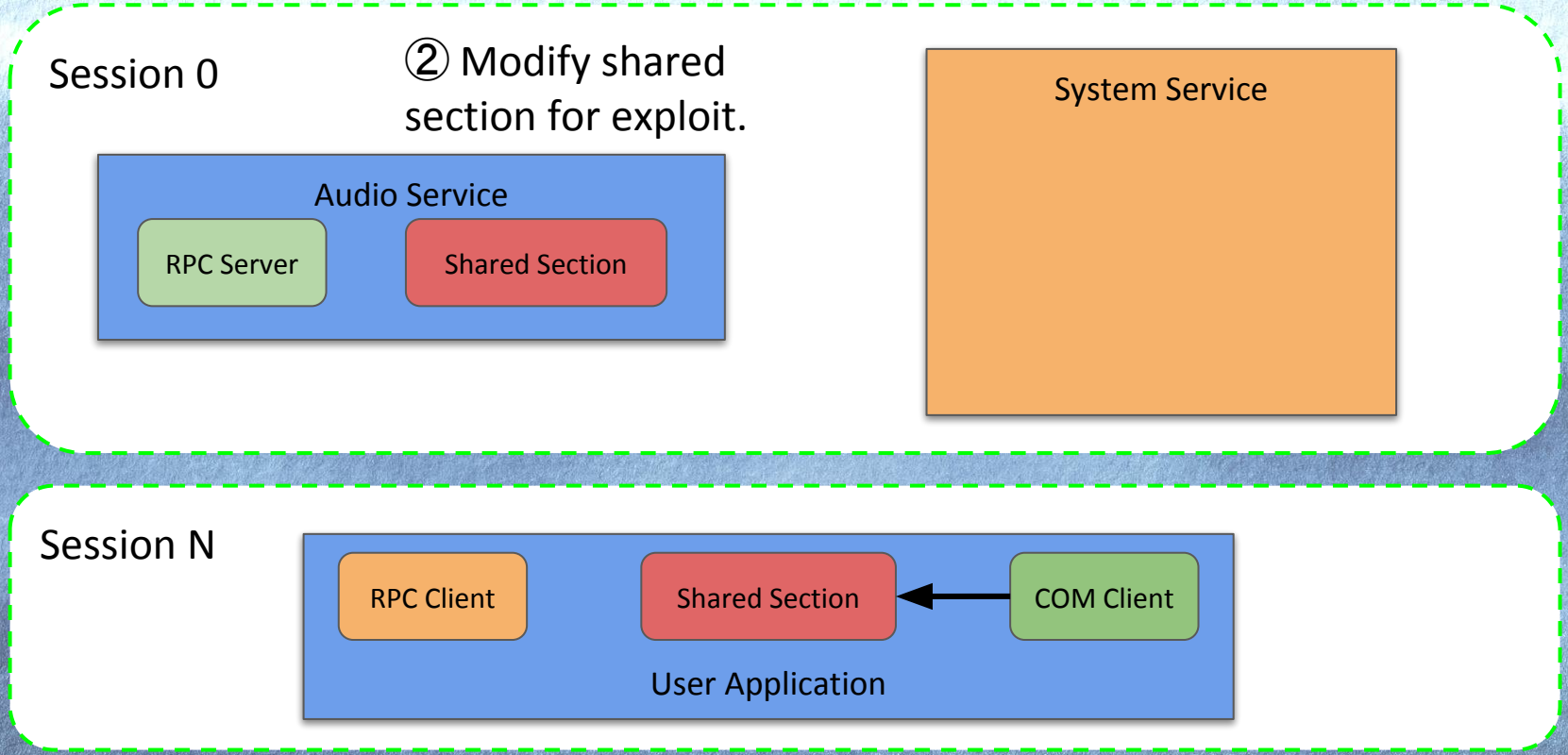
```
HRESULT CSharedMemoryBlock::InitUnMarshal(void *hMem,
                                          unsigned int dwProcessId) {
  unsigned int dwCurrentSession;
  unsigned int dwSourceSession;

  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

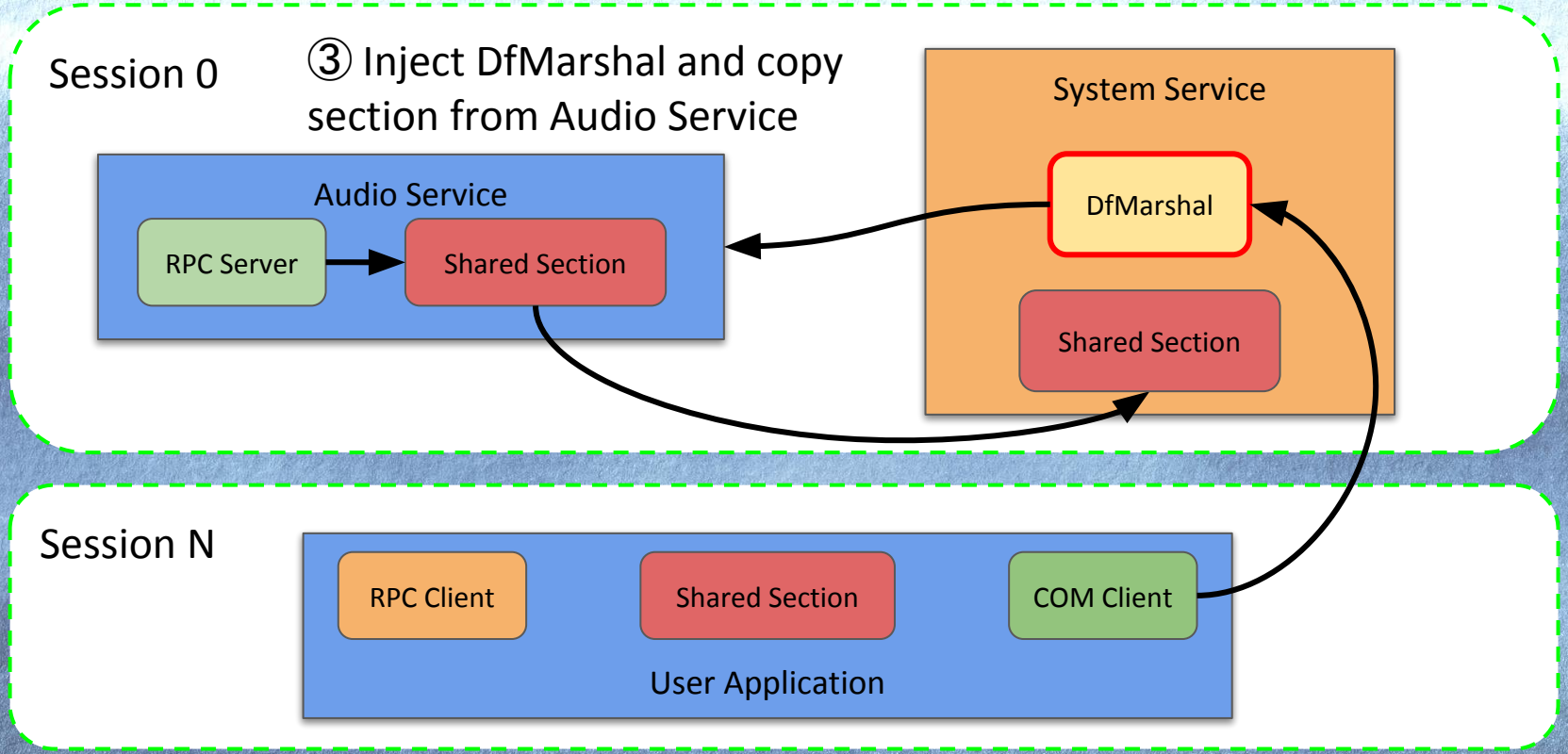
Audio Server to the Rescue



Audio Server to the Rescue



Audio Server to the Rescue



Discovering the Audio Server

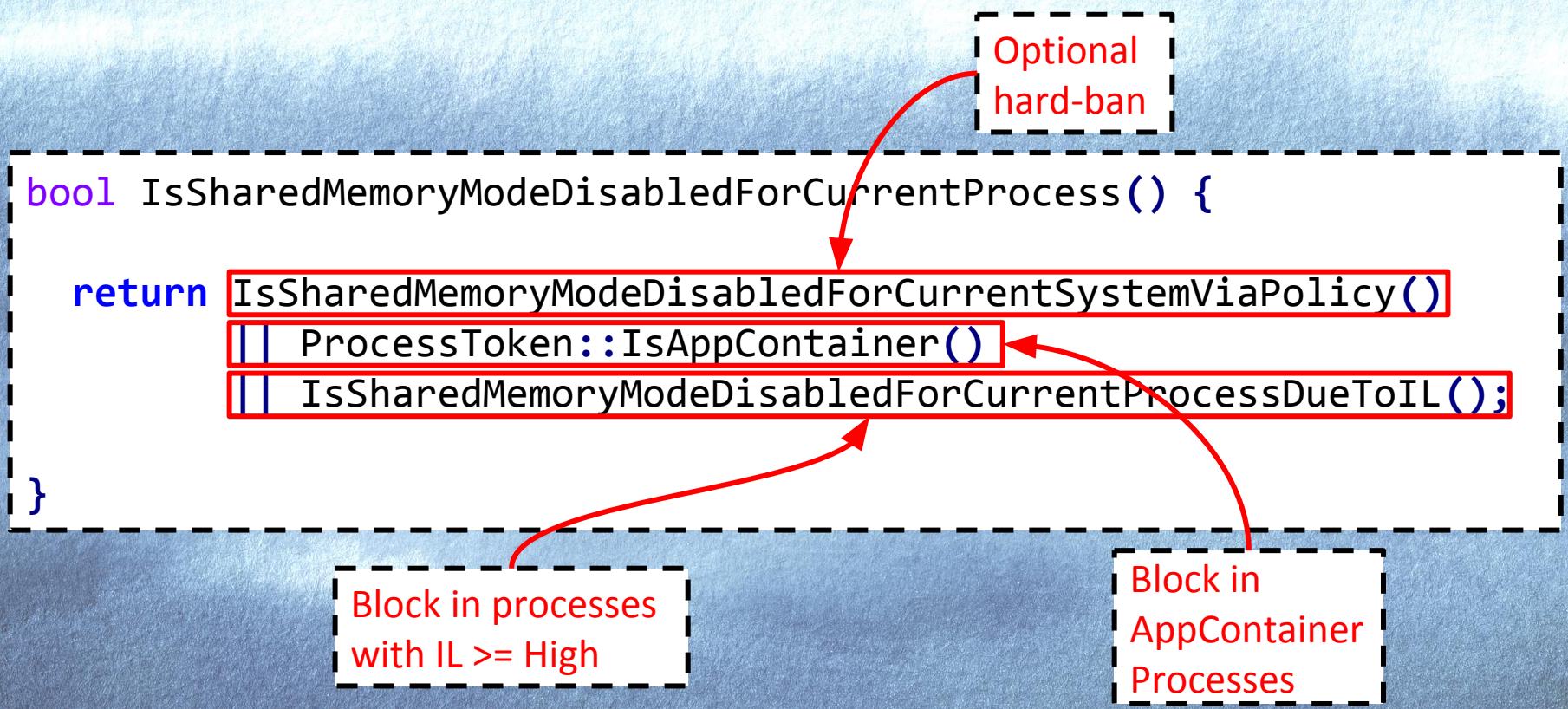
```
PS> $r = ls \windows\system32\*.dll | Get-RpcServer
```

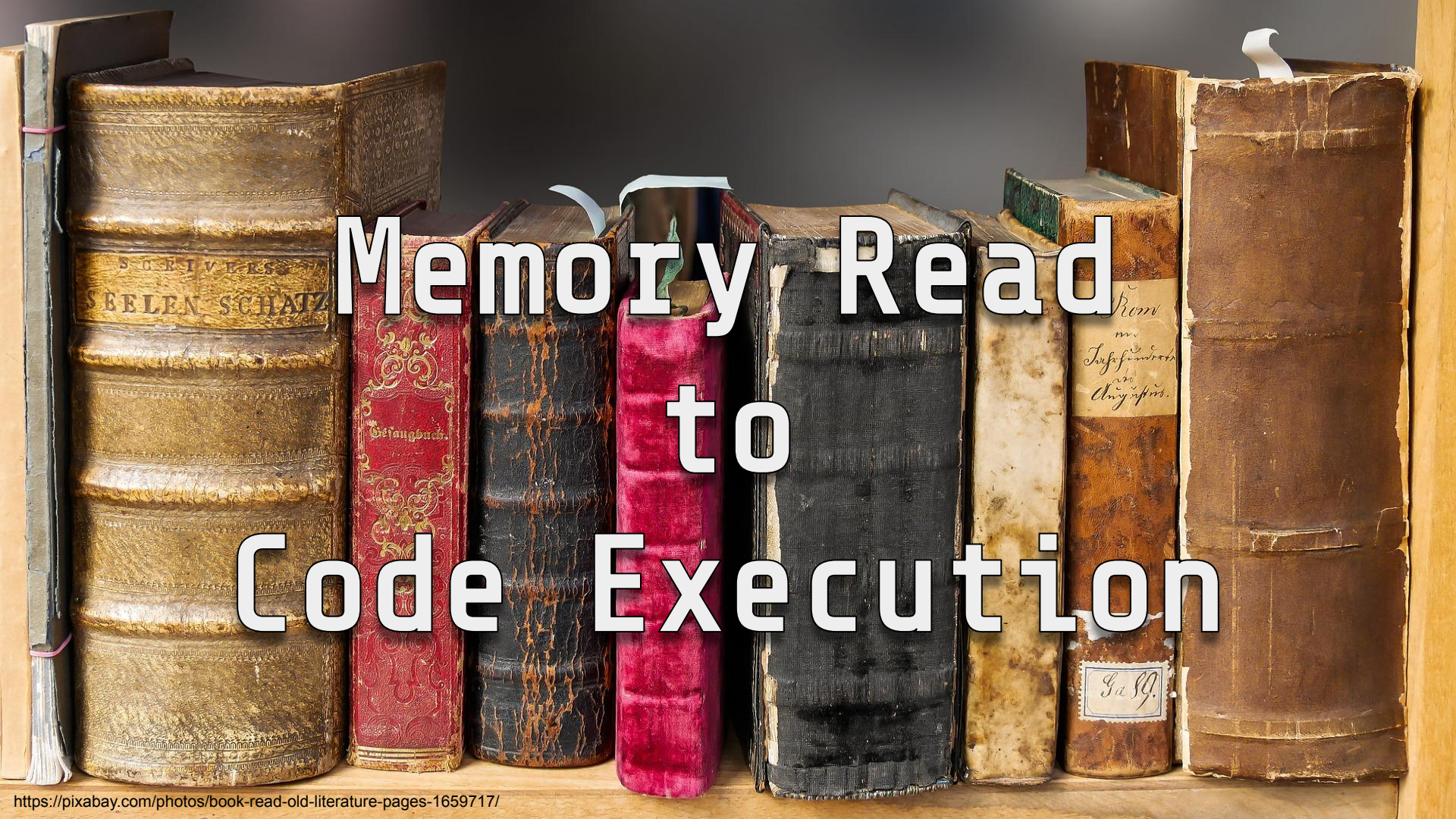
```
function Select-Procedure {
  Param([NtApiDotNet.Ndr.NdrProcedureParameter]$proc)

  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.

https://pixabay.com/photos/panda-cub-wildlife-zoo-cute-china-1203101/

Fixed in CVE-2018-8550

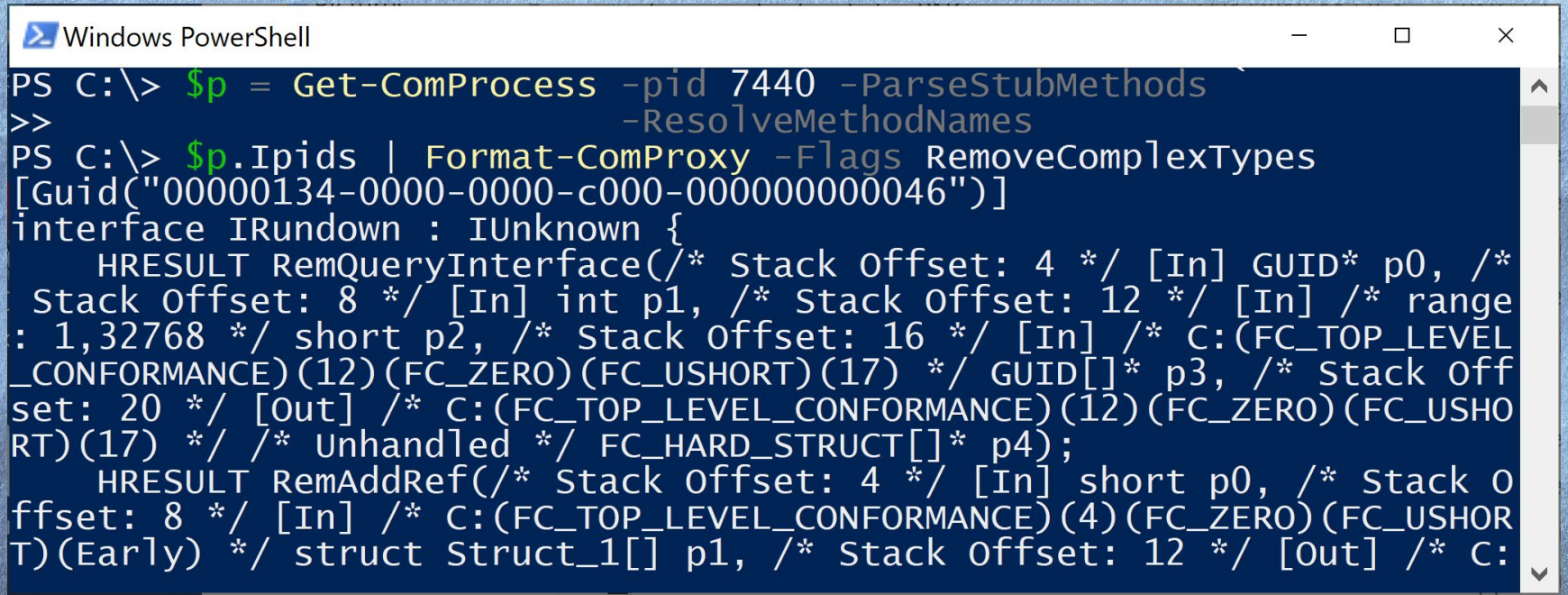




https://pixabay.com/photos/book-read-old-literature-pages-1659717/

Finding Code Injection

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



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

Criteria	***How?***
Process Secret GUID	Read fixed memory location
Context Pointer	Read fixed *g_pMTAEmptyCtx* 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 Host EoP

Code

‹ Prev 4 of 4

Back to list

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:

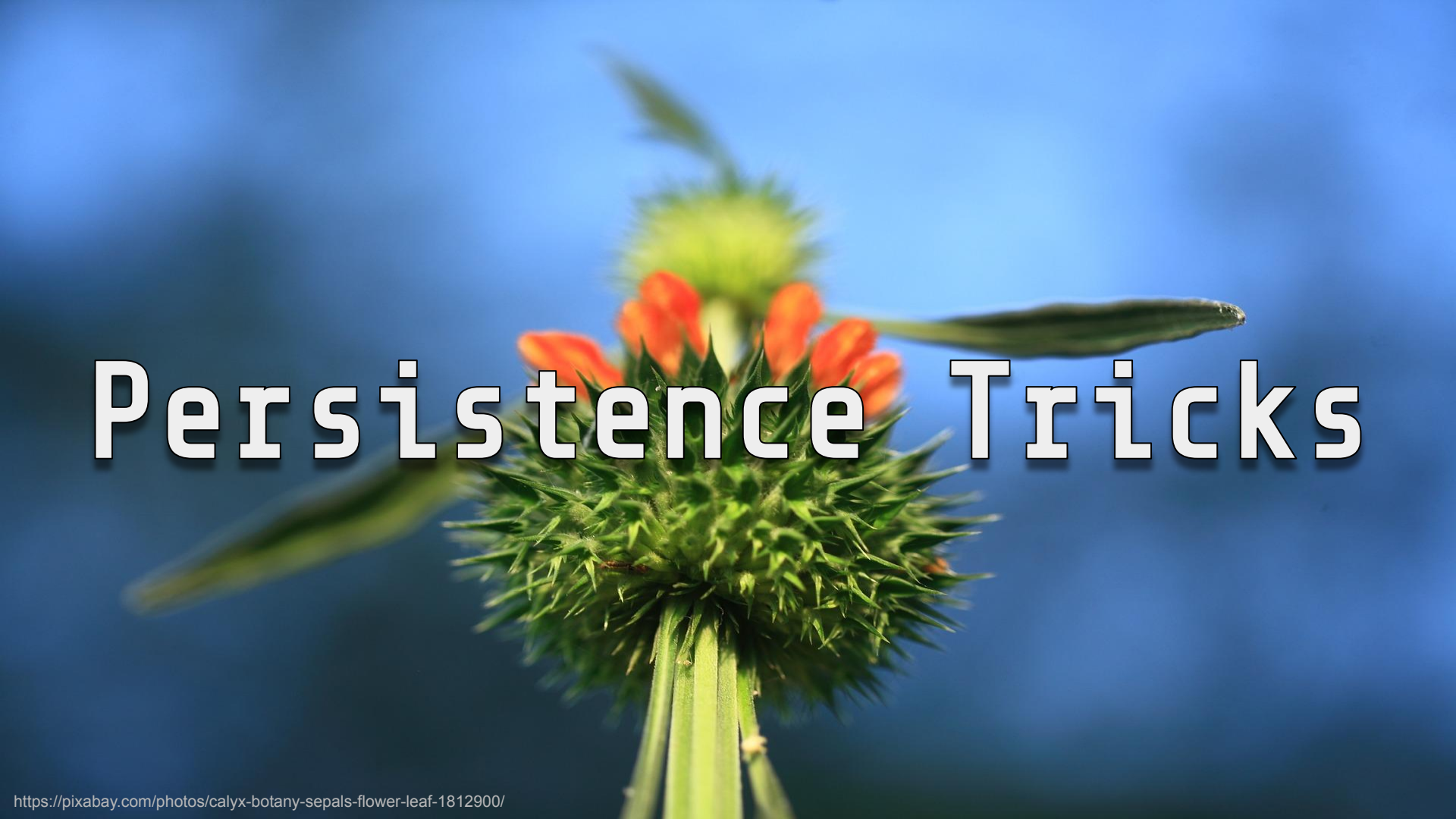
The hardened VirtualBox process on a Windows host doesn't secure its COM interface leading to arbitrary code injection and EoP.

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?

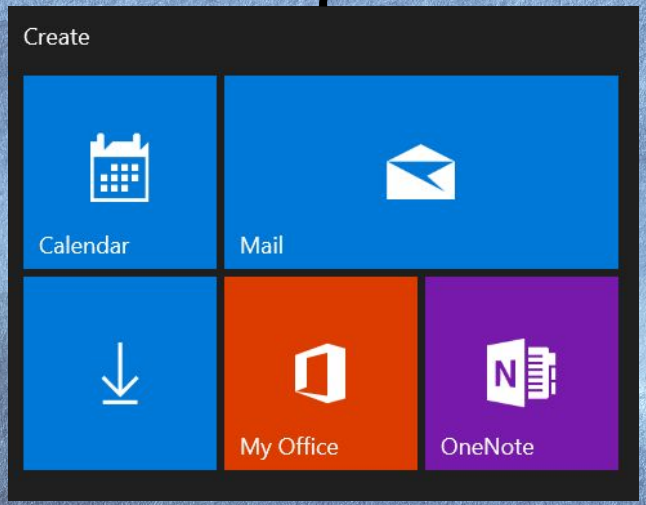
https://bugs.chromium.org/p/project-zero/issues/detail?id=1811

https://pixabay.com/photos/wolf-predator-eurasian-wolf-1975823/



https://pixabay.com/photos/calyx-botany-sepals-flower-leaf-1812900/

Class Lookup



HKCU\Software\Classes**CLSID**

HKLM\Software\Classes**CLSID**

COM Server

ENTERPRISE

TECHNIQUES

All

Initial Access

Execution

Persistence

.bash_profile and .bashrc

Accessibility Features

Account Manipulation

AppCert DLLs

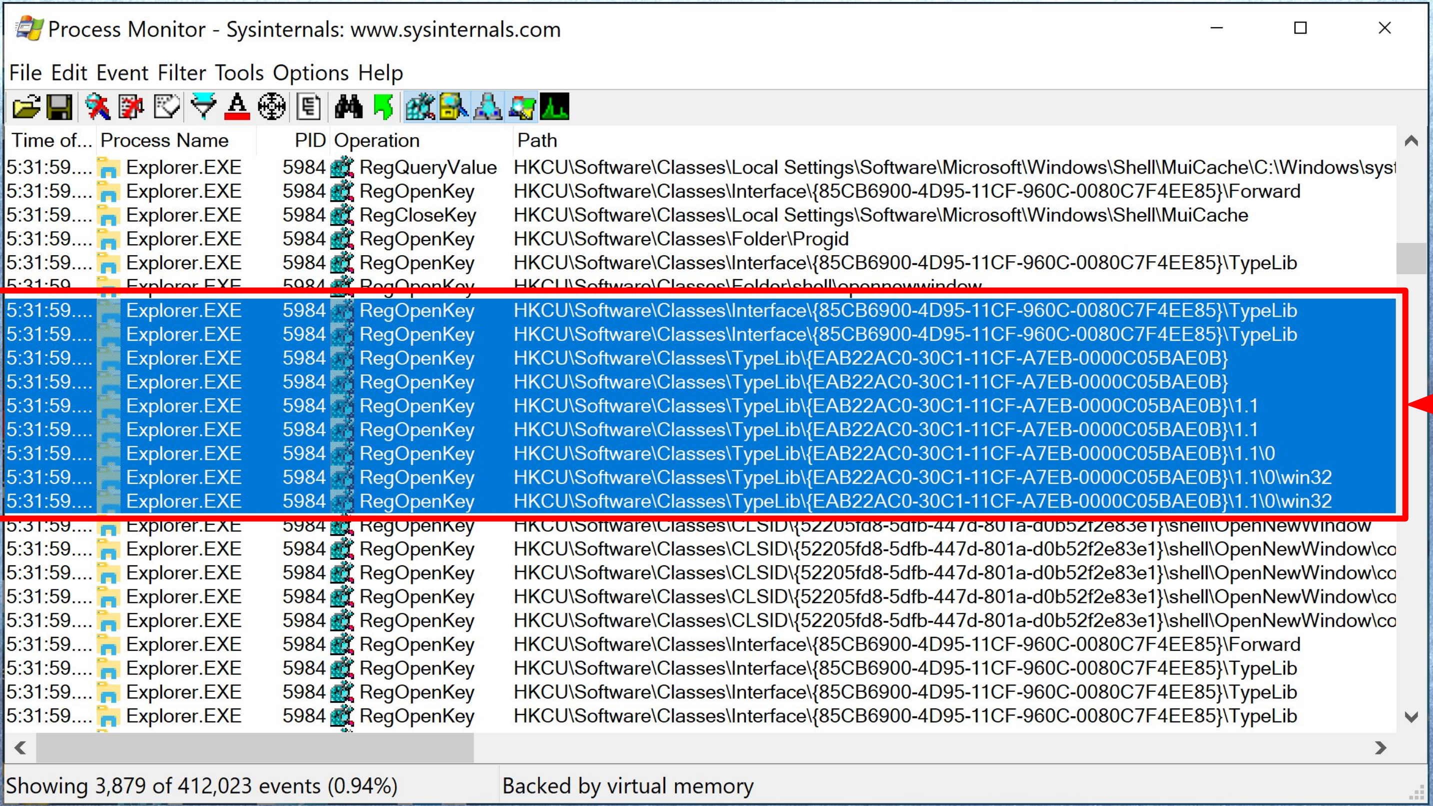
AppInit DLLs

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?



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

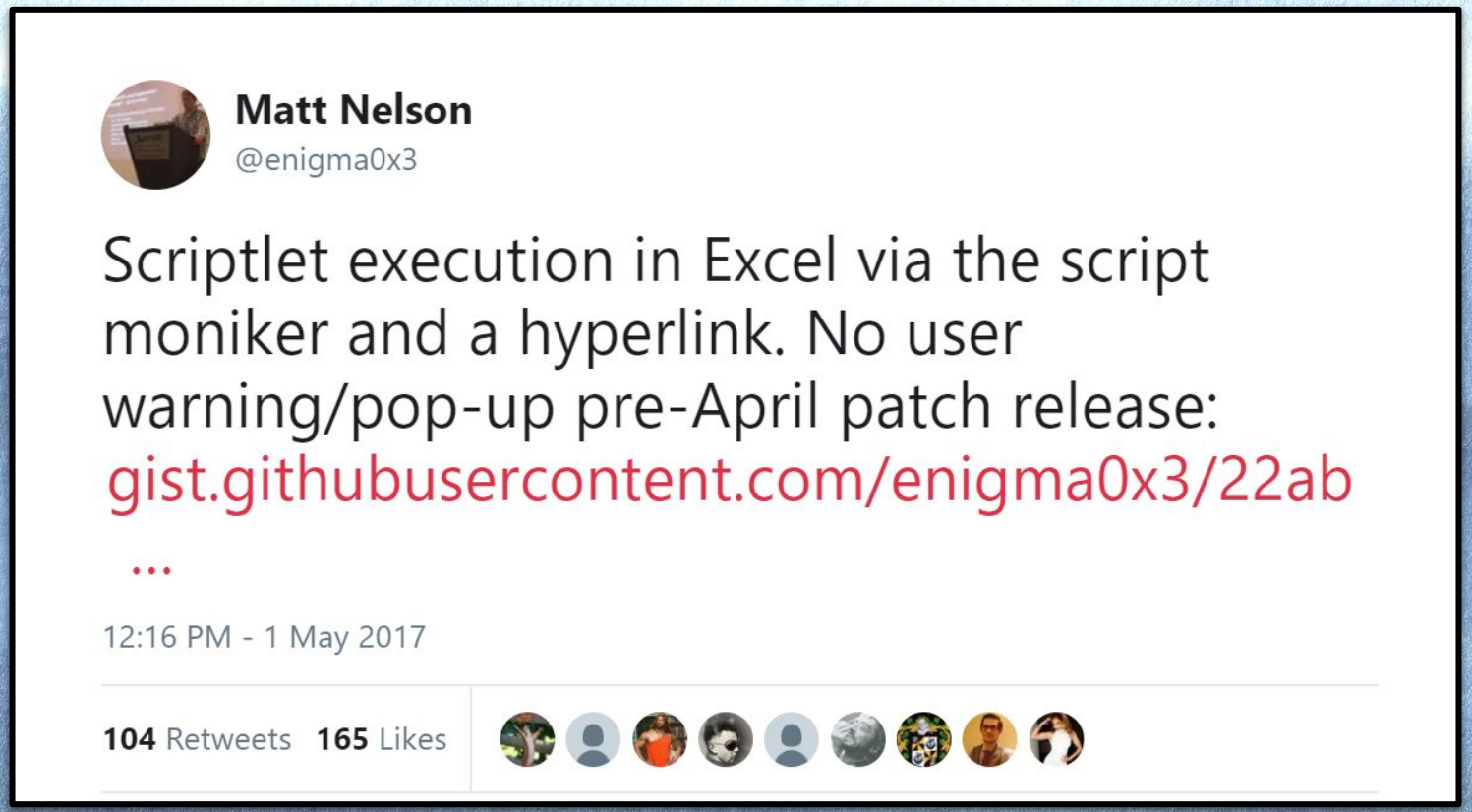
Loads and registers a type library.

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



https://pixabay.com/photos/tiger-predator-fur-beautiful-2320819/

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

https://pixabay.com/photos/fox-red-jumping-hunter-hunting-3822455/