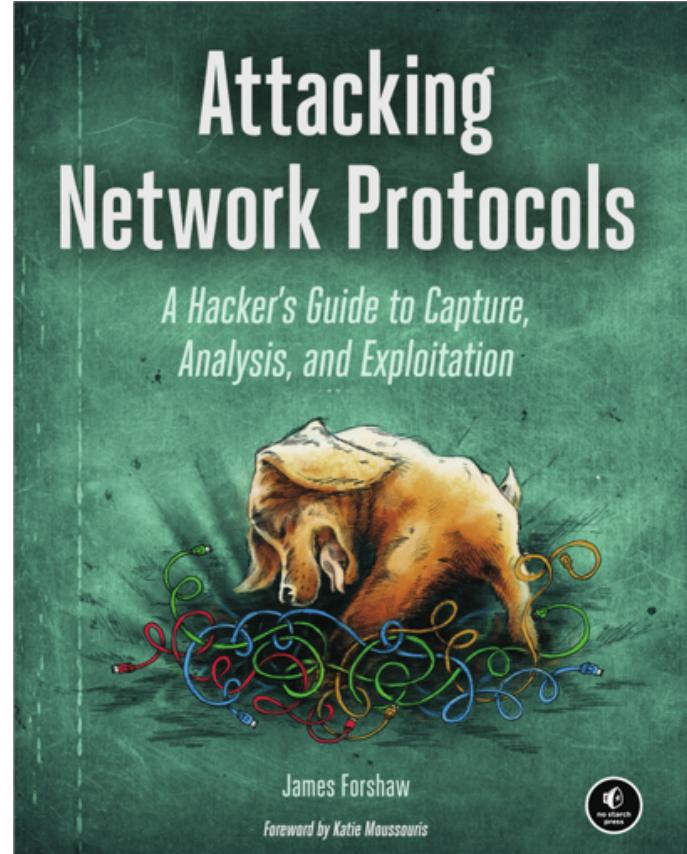
A photograph of a traditional Korean pavilion with a blue tiled roof, reflected in a pond. The pavilion is surrounded by lush green trees and rocks. The foreground shows a rocky path and some fallen leaves.

The Inner Workings of the Windows Runtime

James Forshaw @tiraniddo

Who am I?

- Researcher in Google's Project Zero
- Specialize in Windows
 - Especially local privilege escalation
 - Logical vulnerability specialist
- Author of a book on attacking network protocols
- @tiraniddo on Twitter.



Why Talk About Windows Runtime?

Understand the Technology

Aid to Reverse Engineering

Improve Security Research

Background Research



https://www.troopers.de/downloads/troopers17/TR17_Demystifying_%20COM.pdf

Windows RunTime
Hack In The Box 2012

Sébastien RENAUD srenaud@quarkslab.com
Kévin SZKUDLAPSKI kszkudlapski@quarkslab.com

QUARKSLAB
INNOVATIVE SECURITY

This Talk is based
on Windows 10
1803/1809

OleViewDotNet

[tyranid / oleviewdotnet](#) Unwatch 39 Star 312 Fork 59

[Code](#) [Issues 2](#) [Pull requests 0](#) [Projects 0](#) [Wiki](#) [Insights](#) [Settings](#)

A .net OLE/COM viewer and inspector to merge functionality of OleView and Test Container Edit

[Manage topics](#)

506 commits

2 branches

4 releases

2 contributors

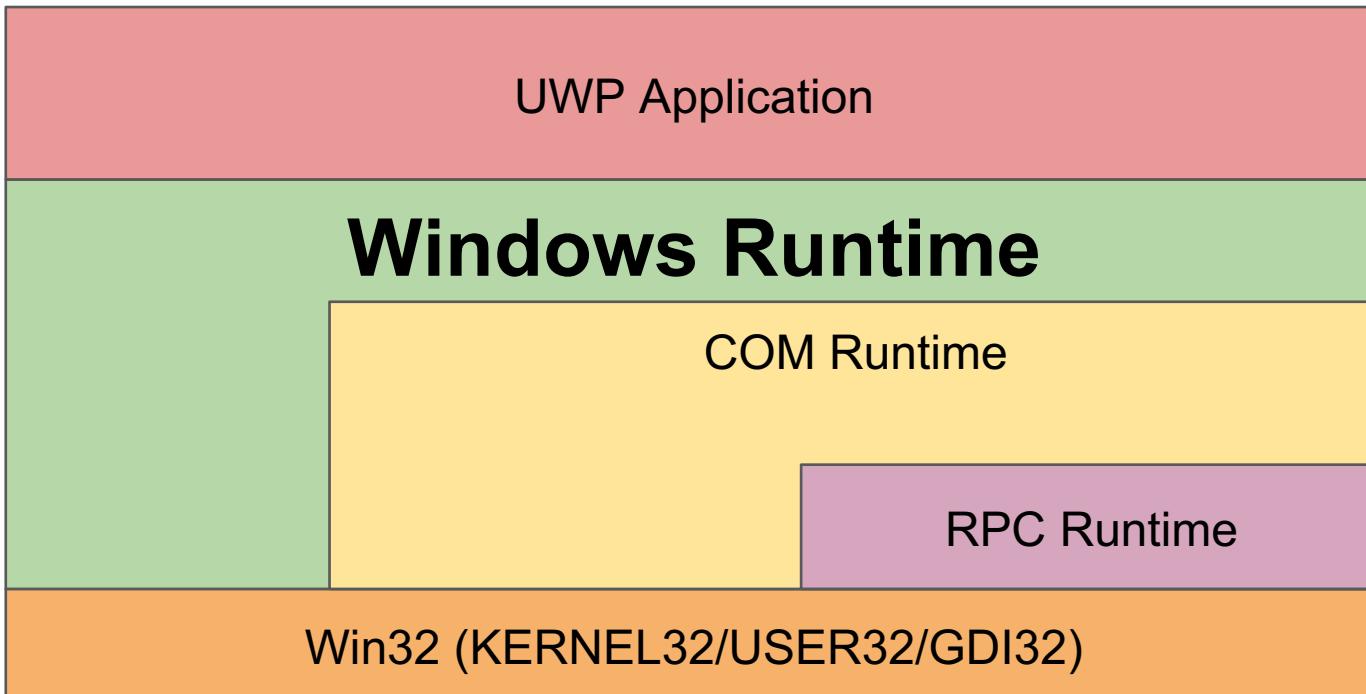
GPL-3.0

[Branch: master ▾](#) [New pull request](#) [Create new file](#) [Upload files](#) [Find file](#) [Clone or download ▾](#)

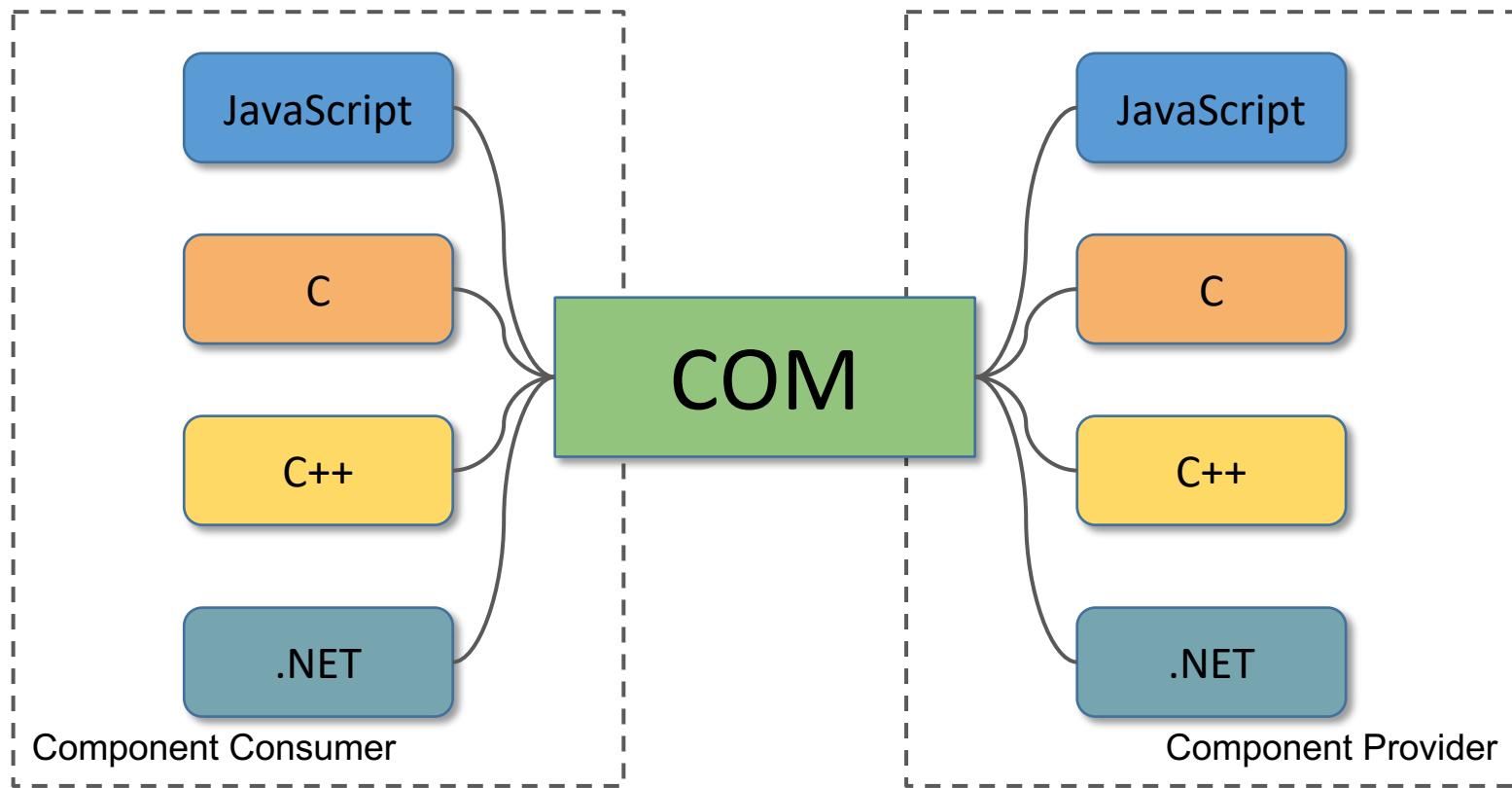
tyranid	Added method to create the Publisher ID from a Publisher Name.	Latest commit <code>acc9d1</code> 15 hours ago
OleViewDotNet.Main	Added method to create the Publisher ID from a Publisher Name.	15 hours ago
OleViewDotNet.PowerShell	Fix naming issues.	3 days ago
OleViewDotNet	Cleanup for assembly information.	a month ago
.gitignore	Updated gitignores and output file to a common location in release bu...	3 months ago

<https://github.com/tyranid/oleviewdotnet>

What's the Windows Runtime (WinRT)?



COM Joins Everything Together



IInspectable the New Root of Evil

```
MIDL_INTERFACE ("AF86E2E0-B12D-4c6a-9C5A-D7AA65101E90")
IInspectable : public IUnknown {
public:
    HRESULT GetIids (
        ULONG *iidCount,
        IID **iids);
    HRESULT GetRuntimeClassName (
        HSTRING *className);
    HRESULT GetTrustLevel (
        TrustLevel *trustLevel);
};
```

Get a list of interface IDs supported by class.

Get class name.

Get class trust level.

Activation Factories

- Component classes can't be directly 'newed' so WinRT defines a factory interface, *IActivationFactory*. Does not use *IClassFactory*.

```
DEFINE_GUID(IID_ActivationFactory,
    "00000035-0000-0000-C000-000000000046");
struct IActivationFactory : public IUnknown {
    HRESULT ActivateInstance(
        IInspectable **instance
    );
};
```

Activation Factories and Instances

```
HRESULT RoGetActivationFactory(  
    HSTRING             activatableClassId,  
    REFIID              iid,  
    LPVOID*             factory);
```

Abbreviated as
ACID

```
HRESULT RoActivateInstance(  
    HSTRING             activatableClassId,  
    IInspectable**      instance,  
);
```

Example ACID: “Windows.Foundation.Uri”

Runtime Class Registry Keys

System
Windows Runtime
Classes

HKEY_LOCAL_MACHINE\Software\Classes

Per-App Runtime
Extension Classes

HKEY_CURRENT_USER\Software\Classes

Per-App Runtime Classes

%ProgramData%\Package\ActivationStore.dat

Runtime Extension Classes

<i>Contract ID</i>	<i>Description</i>
Windows.Launch	Default Application Launch
Windows.Protocol	URI Protocol Handler
Windows.BackgroundTasks	Background Task
Windows.File	Launch and pass a file object
Windows.Search	Search request

Class Trust Levels

```
HRESULT GetTrustLevel(TrustLevel *trustLevel);
```

Full Trust

Can only be created in a fully trusted context

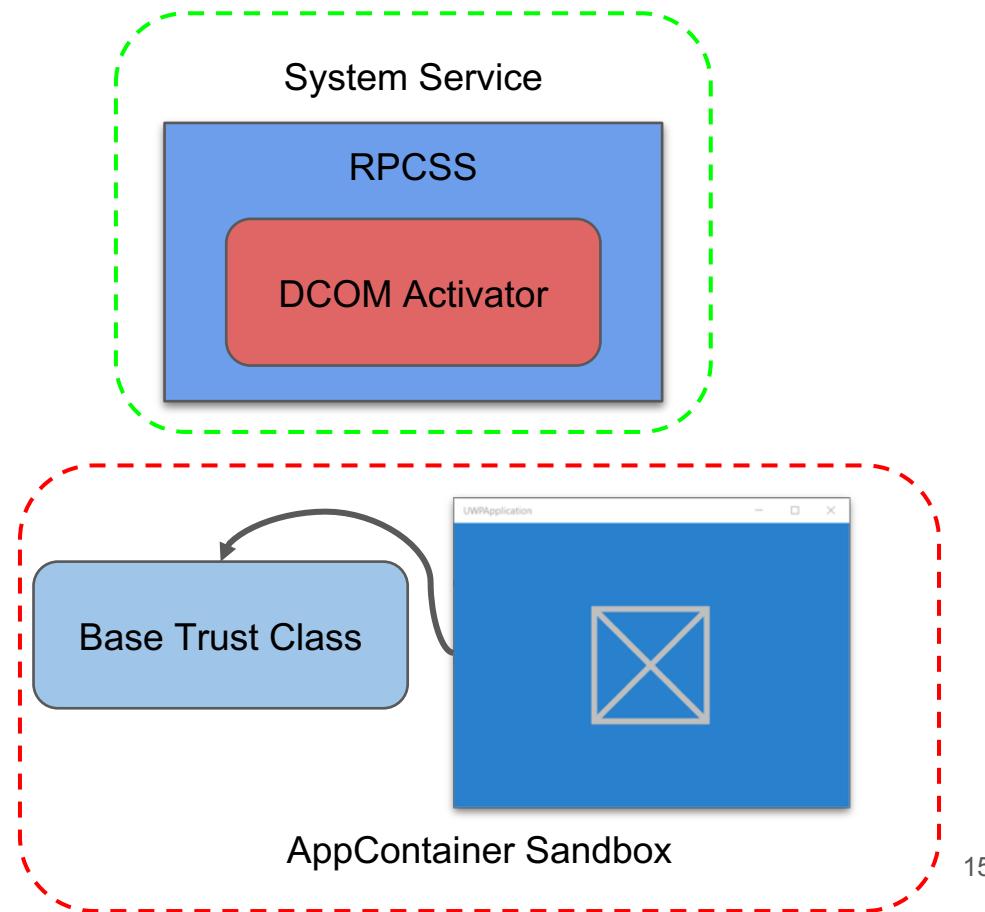
Partial Trust

Can be created in a sandbox context through a broker

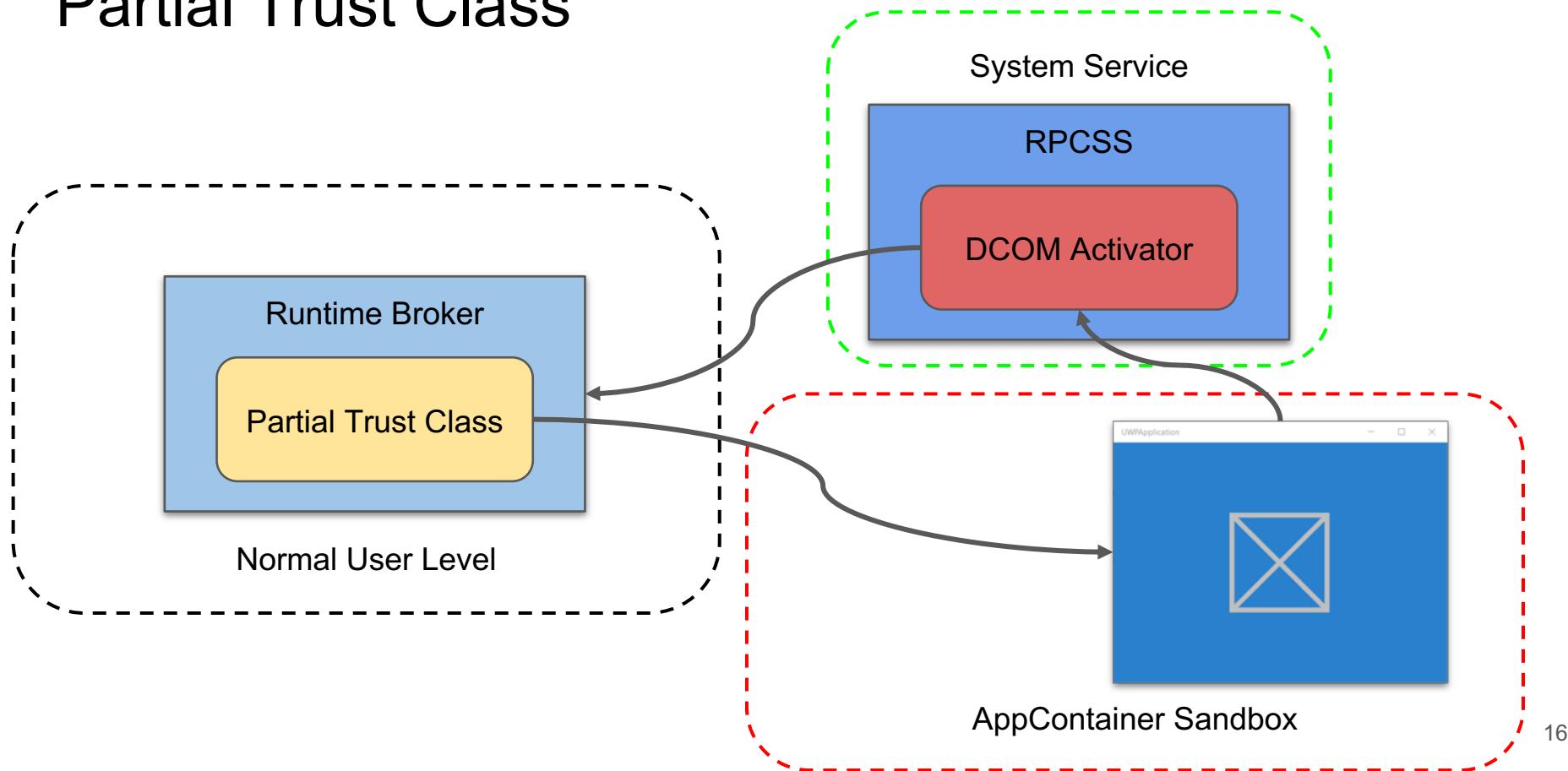
Base Trust

Can be created in any context

Base Trust Class



Partial Trust Class



DEMO 1

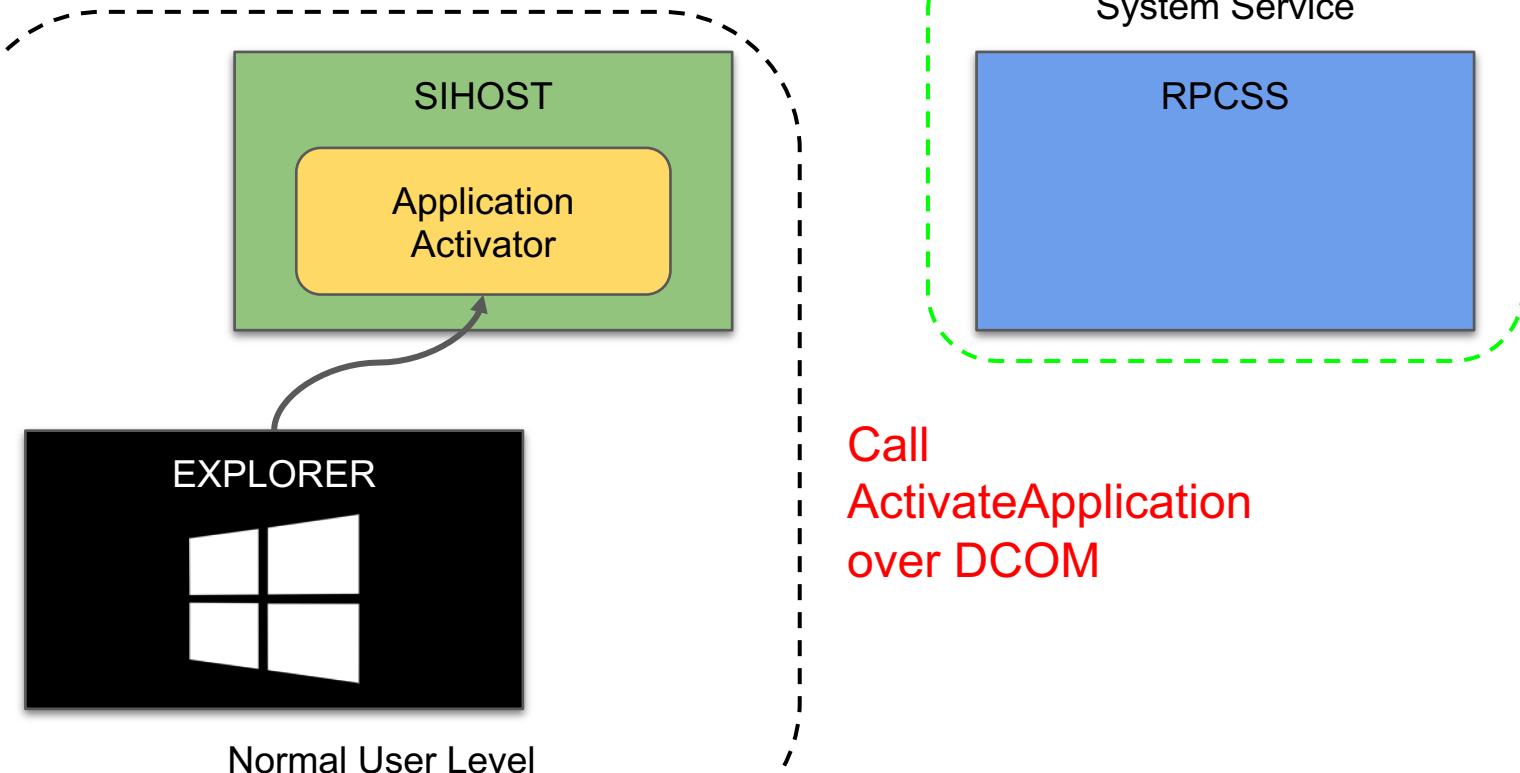
Application Manifest XML

```
<Package>
  <Identity Name="Microsoft.MicrosoftEdge"
            Publisher="CN=Microsoft Corporation, . . ."
            Version="44.17763.1.0"
            ProcessorArchitecture="neutral"/>
<Applications>
  <Application Id="MicrosoftEdge"
               Executable="MicrosoftEdge.exe"
               EntryPoint="MicrosoftEdge.App">
    ...
  </Application>
</Applications>
</Package>
```

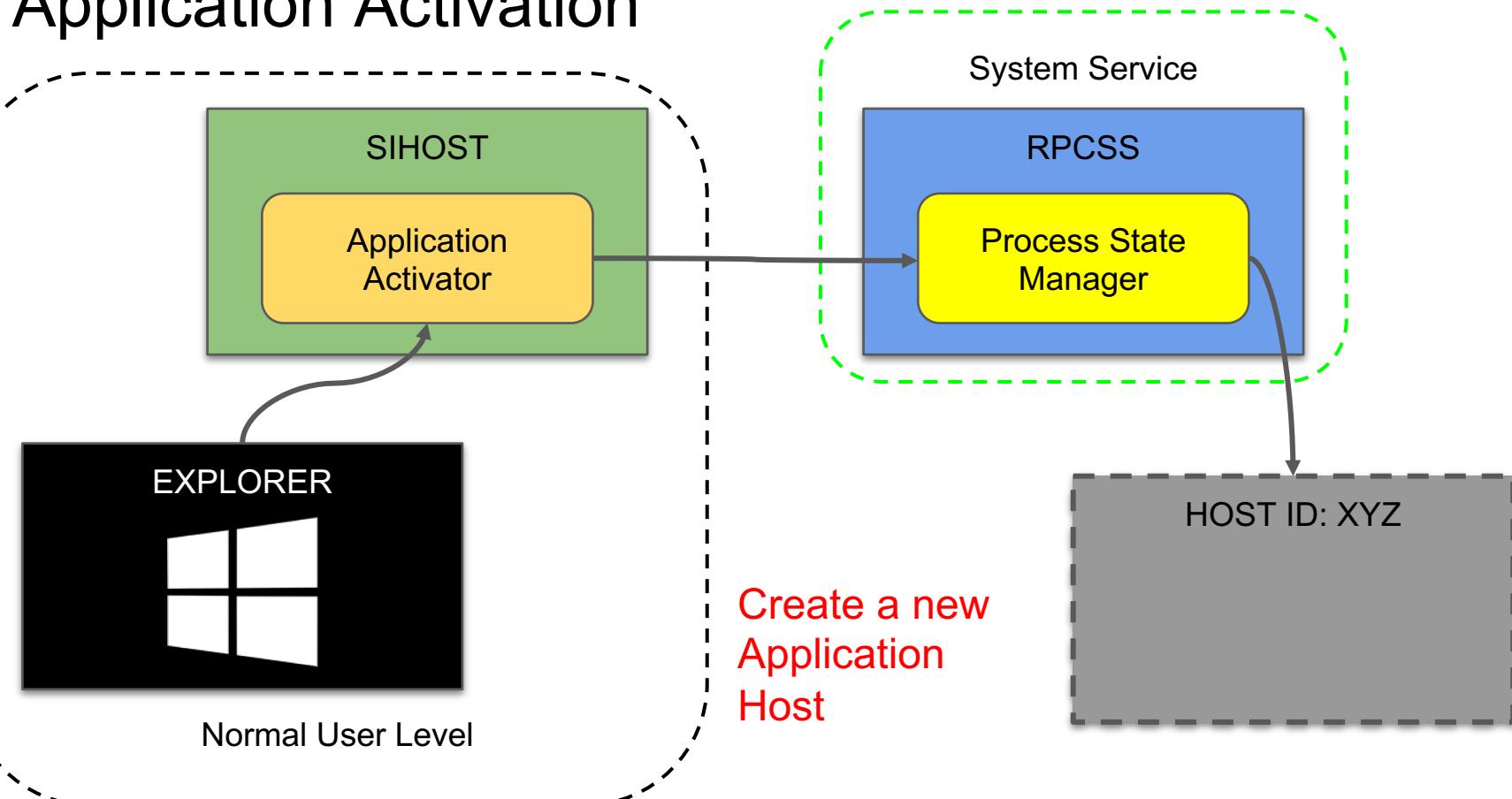
Package Identity

Application Launch

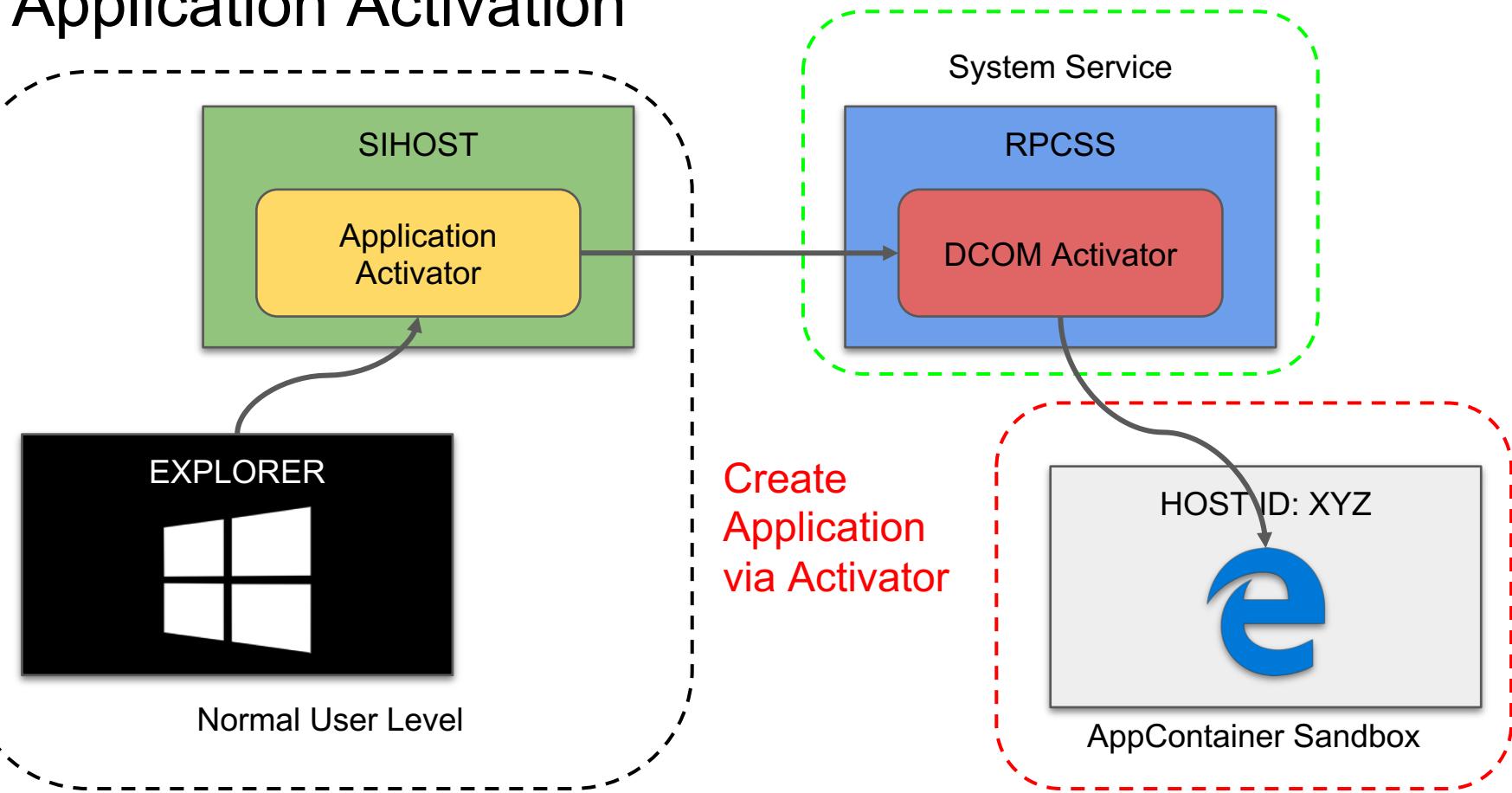
Application Activation



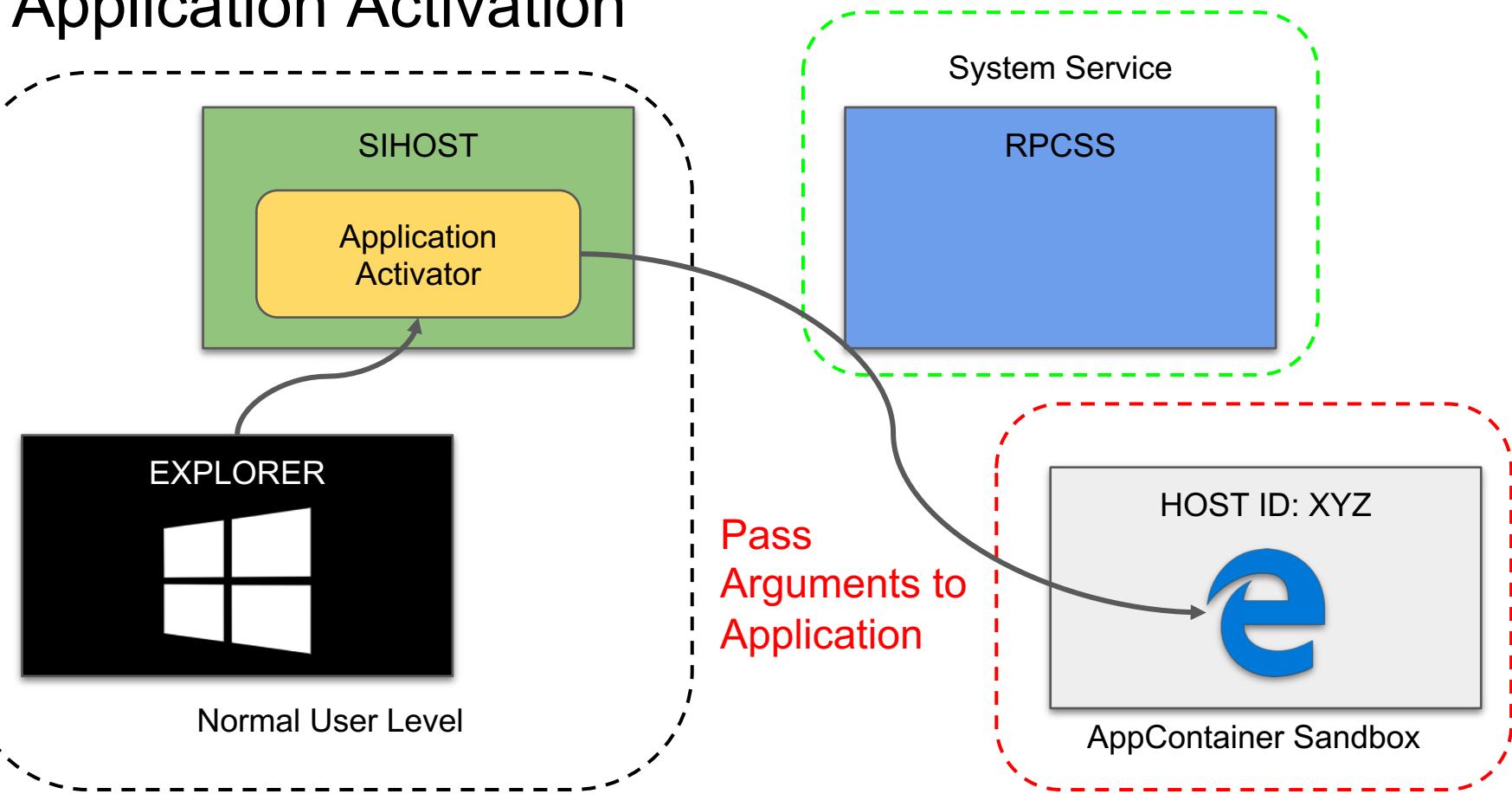
Application Activation



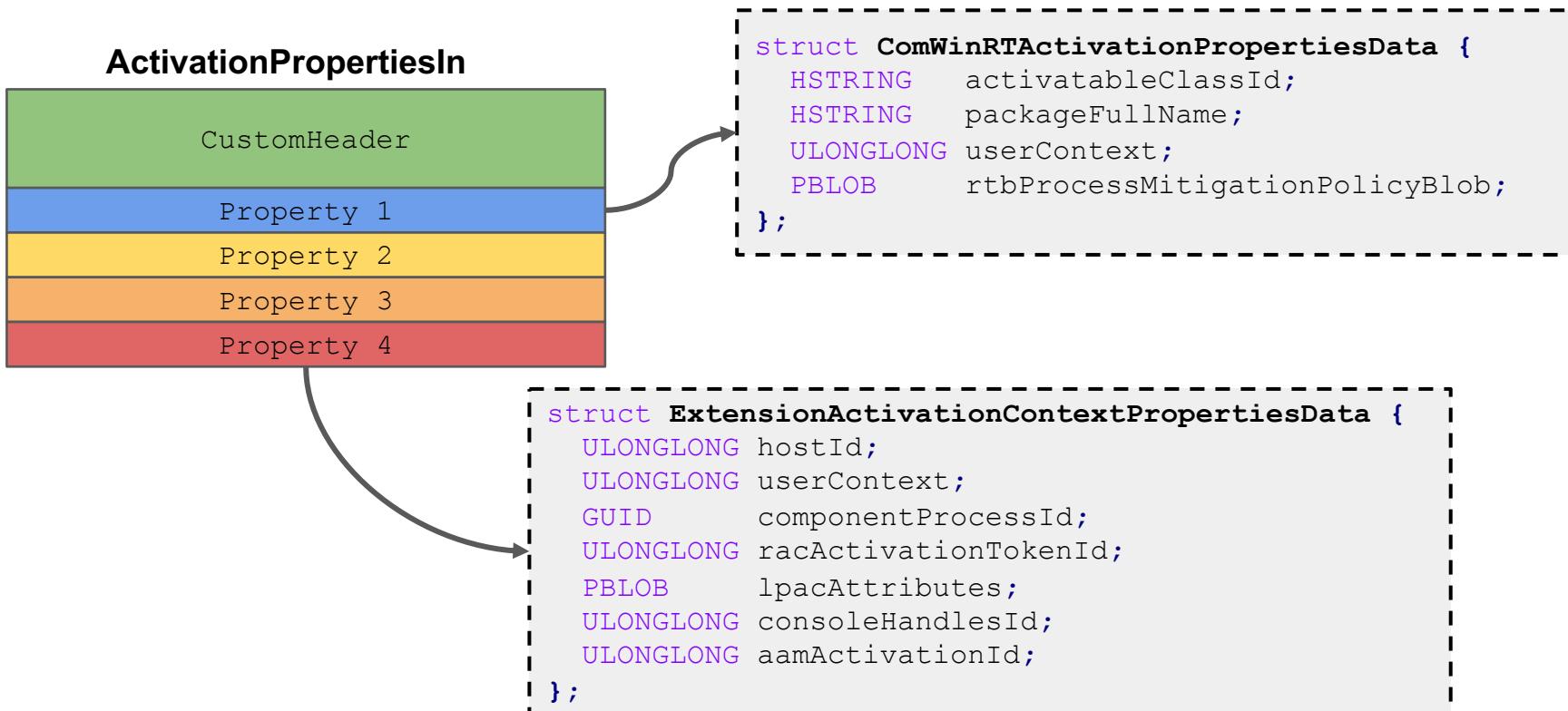
Application Activation



Application Activation



WinRT Activation Properties

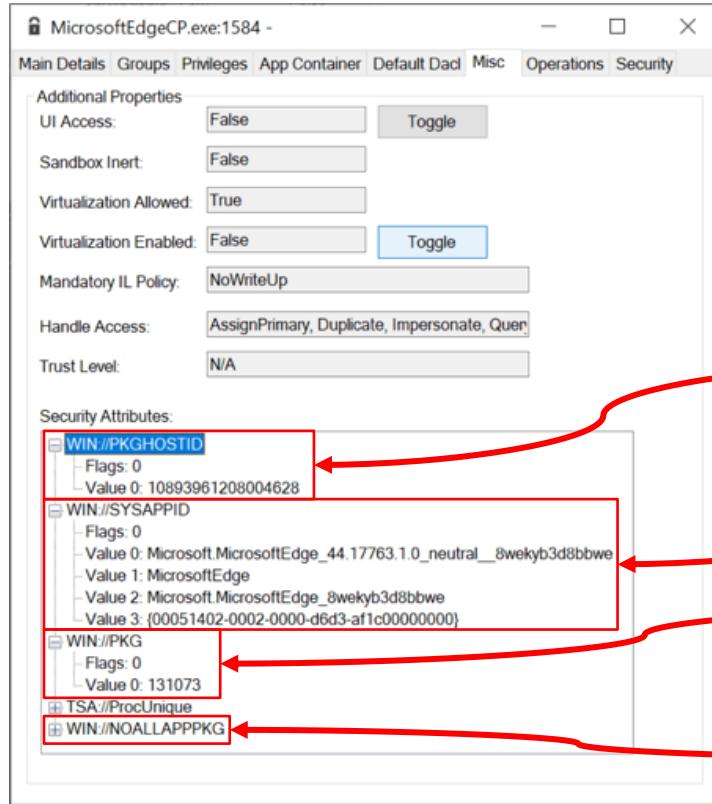


Extension Activation

```
// Exported as Ordinal #65
HRESULT RoGetExtensionRegistration(
    HSTRING contractId,
    HSTRING packageId,
    HSTRING activatableClassId,
    IExtensionRegistration **extensionRegistration);
```

```
IExtensionRegistration* reg =;
RoGetExtensionRegistration("Windows.Launch",
    "Pkg_1.0.0.0_xxxxxxxxxx", "App", &reg);
reg->set_HostId(12345678);
IInspectable* obj;
reg->Activate(&obj);
```

AppContainer Access Token Attributes



Caller needs SeTcbPrivilege to add or modify security attributes.

Application Host ID

System Application ID

Package Flags

Low Privilege App Container

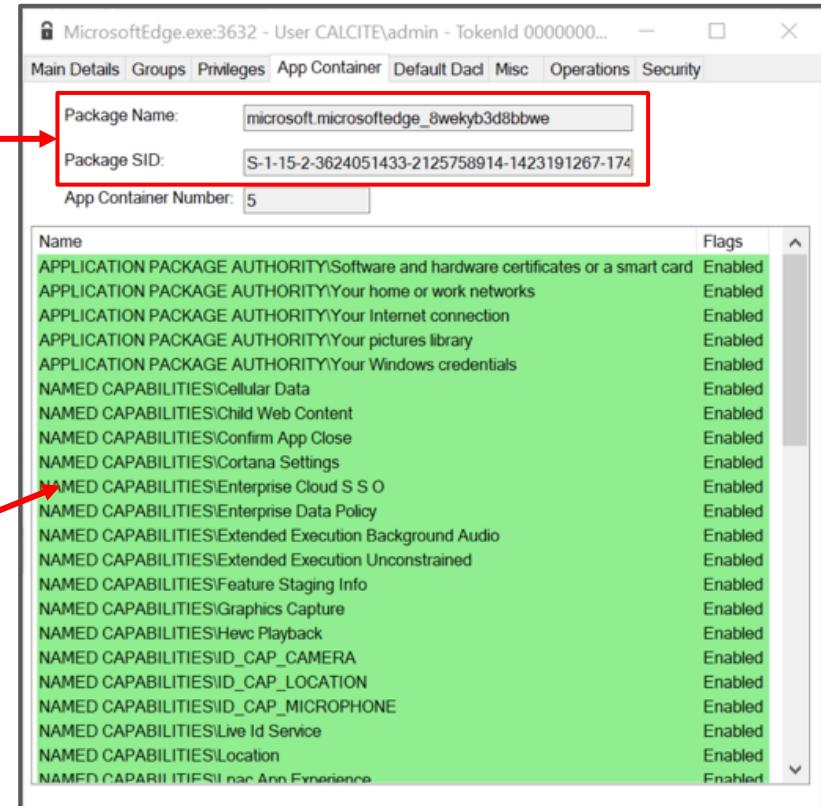
Building the System Application ID

<i>Component</i>	<i>Example</i>
Package Name	<i>Microsoft.MicrosoftEdge</i>
Publisher ID	<i>8wekyb3d8bbwe</i>
Package Family Name	<i>Microsoft.MicrosoftEdge_8wekyb3d8bbwe</i>
Package Full Name	<i>Microsoft.MicrosoftEdge_44.17763.1.0_neutral__8wekyb3d8bbwe</i>
Package Moniker	Same as Package Full Name
Package-Relative App ID	<i>App</i>
Application User Model ID	<i>Microsoft.MicrosoftEdge_8wekyb3d8bbwe!App</i>

AppContainer SID and Capabilities

Package Family
Name and
Package SID

```
<Capabilities>
<Capability Name="internetClient"/>
<Capability Name="privateNetworkClientServer"/>
<rescap:Capability Name="childWebContent"/>
<rescap:Capability Name="confirmAppClose"/>
<rescap:Capability Name="lpacCom"/>
...
<DeviceCapability Name="location"/>
<DeviceCapability Name="microphone"/>
<DeviceCapability Name="webcam"/>
</Capabilities>
```



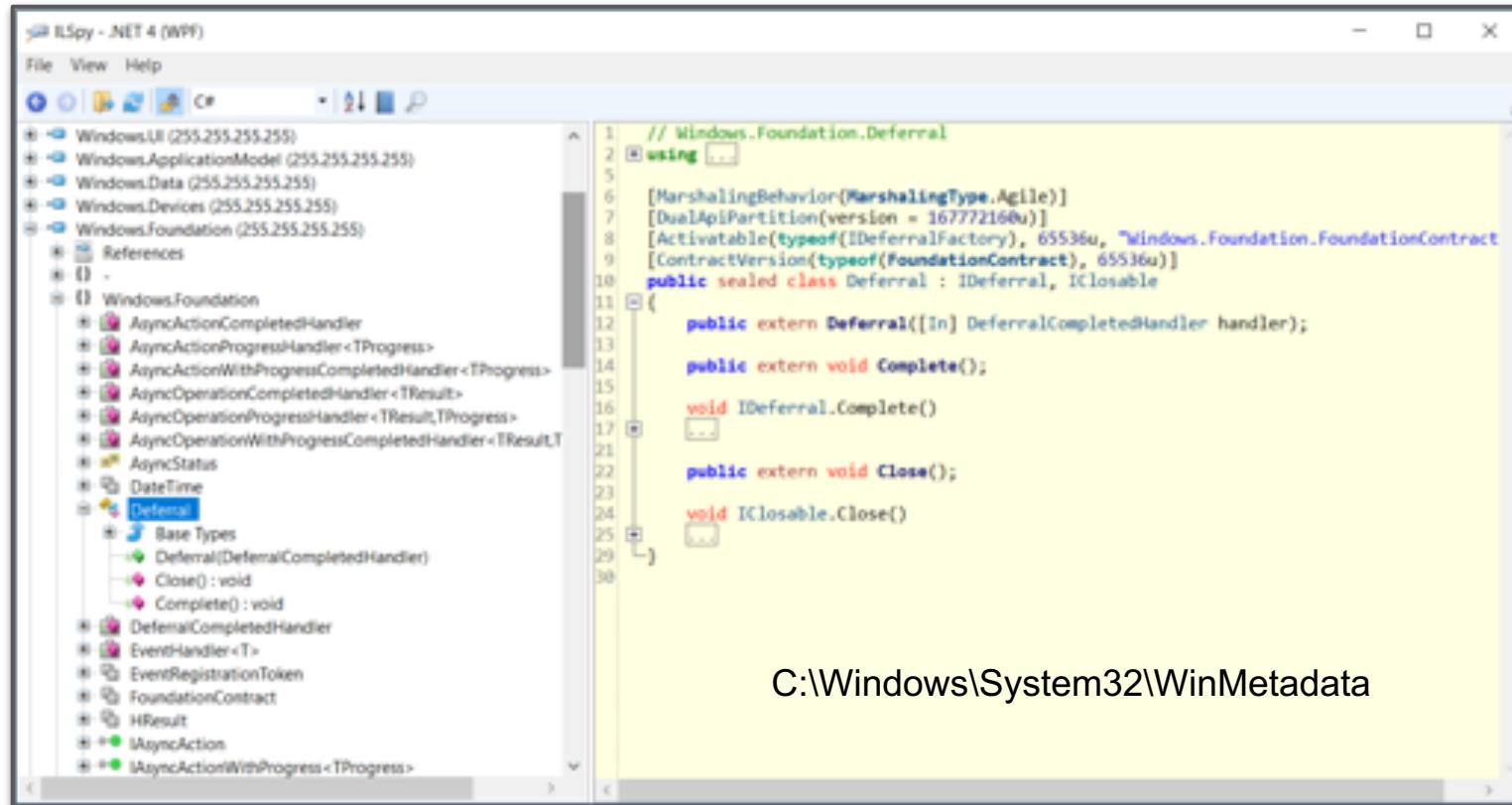
The screenshot shows the Windows Task Manager for MicrosoftEdge.exe. A red arrow points from the 'Package Family Name and Package SID' text above to the 'Package Name' and 'Package SID' fields in the 'Main Details' tab. Another red arrow points from the 'Device Capability' section of the XML code to the 'NAMED CAPABILITIES' table below.

Name	Flags
APPLICATION PACKAGE AUTHORITY\Software and hardware certificates or a smart card	Enabled
APPLICATION PACKAGE AUTHORITY\Your home or work networks	Enabled
APPLICATION PACKAGE AUTHORITY\Your Internet connection	Enabled
APPLICATION PACKAGE AUTHORITY\Your pictures library	Enabled
APPLICATION PACKAGE AUTHORITY\Your Windows credentials	Enabled
NAMED CAPABILITIES\Cellular Data	Enabled
NAMED CAPABILITIES\Child Web Content	Enabled
NAMED CAPABILITIES\Confirm App Close	Enabled
NAMED CAPABILITIES\Cortana Settings	Enabled
NAMED CAPABILITIES\Enterprise Cloud S S O	Enabled
NAMED CAPABILITIES\Enterprise Data Policy	Enabled
NAMED CAPABILITIES\Extended Execution Background Audio	Enabled
NAMED CAPABILITIES\Extended Execution Unconstrained	Enabled
NAMED CAPABILITIES\Feature Staging Info	Enabled
NAMED CAPABILITIES\Graphics Capture	Enabled
NAMED CAPABILITIES\Hvc Playback	Enabled
NAMED CAPABILITIES\ID_CAP_CAMERA	Enabled
NAMED CAPABILITIES\ID_CAP_LOCATION	Enabled
NAMED CAPABILITIES\ID_CAP_MICROPHONE	Enabled
NAMED CAPABILITIES\Live Id Service	Enabled
NAMED CAPABILITIES\Location	Enabled
NAMED CAPABILITIES\Microsoft App Experience	Enabled

DEMO 2

Reverse Engineering Native Components

Windows Metadata



The screenshot shows the IL Spy tool interface with the title "ILSpy - .NET 4 (WPF)". The left pane displays a tree view of assembly references, including Windows.UI, Windows.ApplicationModel, Windows.Data, Windows.Devices, Windows.Foundation, References, and several Windows.* assemblies. The Windows.Foundation assembly is expanded, showing its internal types like Deferral, AsyncActionCompletedHandler, etc. The right pane shows the source code for the `Deferral` class:

```
// Windows.Foundation.Deferral
using ...;

[MarshalingBehavior(MarshalingType.Agile)]
[DualApiPartition(version = 167772160u)]
[Activatable(typeof(IDeferralFactory), 65536u, "Windows.Foundation.FoundationContract")]
[ContractVersion(typeof(FoundationContract), 65536u)]
public sealed class Deferral : IDeferral, IClosable
{
    public extern Deferral([In] DeferralCompletedHandler handler);

    public extern void Complete();

    void IDeferral.Complete();
}

public extern void Close();

void IClosable.Close();
}
```

Below the code pane, the path `C:\Windows\System32\WinMetadata` is displayed.

Combining Interfaces

```
class RuntimeClass {  
    // Default constructor.  
    public RuntimeClass();  
    // Constructor with parameter.  
    public RuntimeClass(int p);  
    // Static method.  
    public static int A();  
    // Instance method.  
    public int B();  
}
```

Factory Object

Instance Object

Combining Interfaces

```
class RuntimeClass {  
    // Default constructor.  
    public RuntimeClass();  
    // Constructor with parameter.  
    public RuntimeClass(int p);  
    // Static method.  
    public static int A();  
    // Instance method.  
    public int B();  
}
```

Factory Object

```
interface IActivationFactory {  
    HRESULT ActivateInstance(  
        IInspectable **instance  
    );  
}
```

Instance Object

```
interface IRuntimeClass {  
    HRESULT B(int* retval);  
}
```

Combining Interfaces

```
class RuntimeClass {  
    // Default constructor.  
    public RuntimeClass();  
    // Constructor with parameter.  
    public RuntimeClass(int p);  
    // Static method.  
    public static int A();  
    // Instance method.  
    public int B();  
}
```

Factory Object

```
interface IActivationFactory {  
    HRESULT ActivateInstance(  
        IInspectable **instance  
    );  
}
```

```
interface IRuntimeClassFactory {  
    HRESULT ActivateInstanceWithParam(  
        int p,  
        IRuntimeClass** instance);  
}
```

```
interface IRuntimeClassStatics {  
    HRESULT A(int* retval);  
}
```

Instance Object

```
interface IRuntimeClass {  
    HRESULT B(int* retval);  
}
```

Finding the Implementation Binary

Get object for the class

```
PS> $cls = Get-ComRuntimeClass -Name "Class.Name"
```

If In-Process get DLL path

```
PS> $cls.DllPath
```

If OOP NormalExe get Server Exe Path

```
PS> $cls.ServerEntry.ExePath
```

If OOP service get Service name

```
PS> $cls.ServerEntry.ServiceName
```

Activation Entry Points

Exported from a DLL

```
HRESULT DllGetActivationFactory(  
    HSTRING activatableClassId,  
    IActivationFactory **factory  
) ;
```

Called in an EXE

```
HRESULT RoRegisterActivationFactories(  
    HSTRING *activatableClassIds,  
    PFNGETACTIVATIONFACTORY *activationFactoryCallbacks,  
    UINT32 count,  
    RO_REGISTRATION_COOKIE *cookie  
) ;
```

C++ Application Frameworks

C++/CX (Custom C++ dialect)

```
void App::OnLaunched(LaunchActivatedEventArgs^ e) {
    Handler^ handler = ref new Handler();
    handler->HandleLaunch("Launched");
}
```

C++/WRL (C++ 11)

```
HRESULT App::OnLauncher(ILaunchActivatedEventArgs* e) {
    ComPtr<IHandler> handler;
    HRESULT hr = Make<Handler>(&handler)
    if (FAILED(hr))
        return hr;
    HStringReference str(L"OnLaunched");
    return handler->HandleLaunch(str.Get());
}
```

C++/WINRT (C++ 17)

```
void App::OnLaunched(LaunchActivatedEventArgs const& e) {
    Handler handler = Handler();
    handler.HandleLaunch(hstring(L"Launched"));
}
```

IDL File

```
namespace WRLClass {
    [uuid(E74F1CF0-59C7-4CA6-BDE5-0F9DED9B4EF7),
     version(1.0), exclusiveto(WinRTClass)]
    interface IWinRTClass : IInspectable {
        HRESULT Add([in] int a, [in] int b,
                    [out, retval] int* value);
    }

    [version(1.0), activatable(1.0)]
    runtimeclass WinRTClass {
        [default] interface IWinRTClass;
    }
}
```

C++/WRL Implementation

```
class WinRTClass : public RuntimeClass<IWinRTClass> {
   InspectableClass(L"WRLClass.WinRTClass", BaseTrust)
public:
    HRESULT STDMETHODCALLTYPE Add(
        /* [in] */ int a,
        /* [in] */ int b,
        /* [retval, out] */ int * value
    ) override {
        *value = a + b;
        return S_OK;
    }
};

ActivatableClass(WinRTClass);
```

Define base implementation of IInspectable

Interface Implementation

Define ActivationFactory

Finding Implemented Interfaces

```
HRESULT QueryInterface(REFIID riid, void** ppv) {
    bool handled = false;
    HRESULT hr = CustomQueryInterface(riid, ppv, &handled);
    if (FAILED(hr) || handled)
        return hr;
    return Super::AsIID(this, riid, ppv);
}
```

Overridable
Custom QI

Call AsIID
helper
method

AsIID Helper

```
HRESULT AsIID(RuntimeClass<IT...*>* implements,
               REFIID riid, void **ppv) {
    HRESULT hr = E_NOINTERFACE;
    if (riid == __uuidof(IUnknown)
        || riid == __uuidof(IInspectable)) {
        *ppv = implements->CastToUnknown();
        hr = S_OK;
    } else {
        hr = implements->CanCastTo(riid, ppv);
    }
    if (SUCCEEDED(hr))
        static_cast<IUnknown*>(*ppv)->AddRef();
    return hr;
}
```

Variadic
Template

Handle Base
Case

Specific
CanCastTo

CanCastTo Helper

Variadic
Template
Expanded

```
HRESULT RuntimeClass<I1, I2, I3> CanCastTo(REFIID riid,  
                                         void* ppv) {  
  
    if (riid == __uuidof(I1)) {  
        ppv = static_cast<I1*>(this);  
    } else if (riid == __uuidof(I2)) {  
        ppv = static_cast<I2*>(this);  
    } else if (riid == __uuidof(I3)) {  
        ppv = static_cast<I2*>(this);  
    } else {  
        return E_NOINTERFACE;  
    }  
    return S_OK;  
}
```

Test Each
Interface

String Handles (HSTRING)

```
typedef struct HSTRING__{
    int unused;
} HSTRING__;
```

Opaque string handle structure.

```
// Declare the HSTRING handle for C/C++
typedef HSTRING__ * HSTRING;
```

```
WindowsCreateString(
    PCNZWCH sourceString,
    UINT32 length,
    HSTRING *string
);
```

Reference counted on the heap.

```
WindowsCreateStringReference(
    PCWSTR sourceString,
    UINT32 length,
    HSTRING_HEADER *hstringHeader,
    HSTRING *string
);
```

Scoped on the stack.

The Real HSTRING

```
struct HSTRING_HEADER_INTERNAL {
    WINDOWS_RUNTIME_HSTRING_FLAGS flags;
    unsigned int length;           Used for stack scoped
    unsigned int padding1;         "reference" strings.
    unsigned int padding2;
    const wchar_t *StringRef;
};
```

```
struct STRING_OPAQUE {
    HSTRING_HEADER_INTERNAL header;
    volatile int refcount;
    wchar_t string[1];           Inline string data and
                                reference count for use
                                on the heap
```

```
PCWSTR WindowsGetStringRawBuffer(
    HSTRING string,
    UINT32 *length
);
```

Call to get raw
buffer and length.

Create a new instance of a COM object.

```
PS> $obj = New-ComObject -Class $cls
```

Get all known interfaces for a class.

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

Get proxy information for a list of interfaces.

```
PS> $prx = $intf | Get-ComProxy
```

Format the COM proxies as text.

```
PS> $prx | Format-ComProxy
```

Debugging Applications

Get all registered Windows.Launch Extensions

```
PS> Get-ComRuntimeExtension -Launch | `  
      Select PackageId, AppId
```

Start a package and debug it.

```
PS> windbg.exe -plmPackage PKGID -plmApp APPID
```

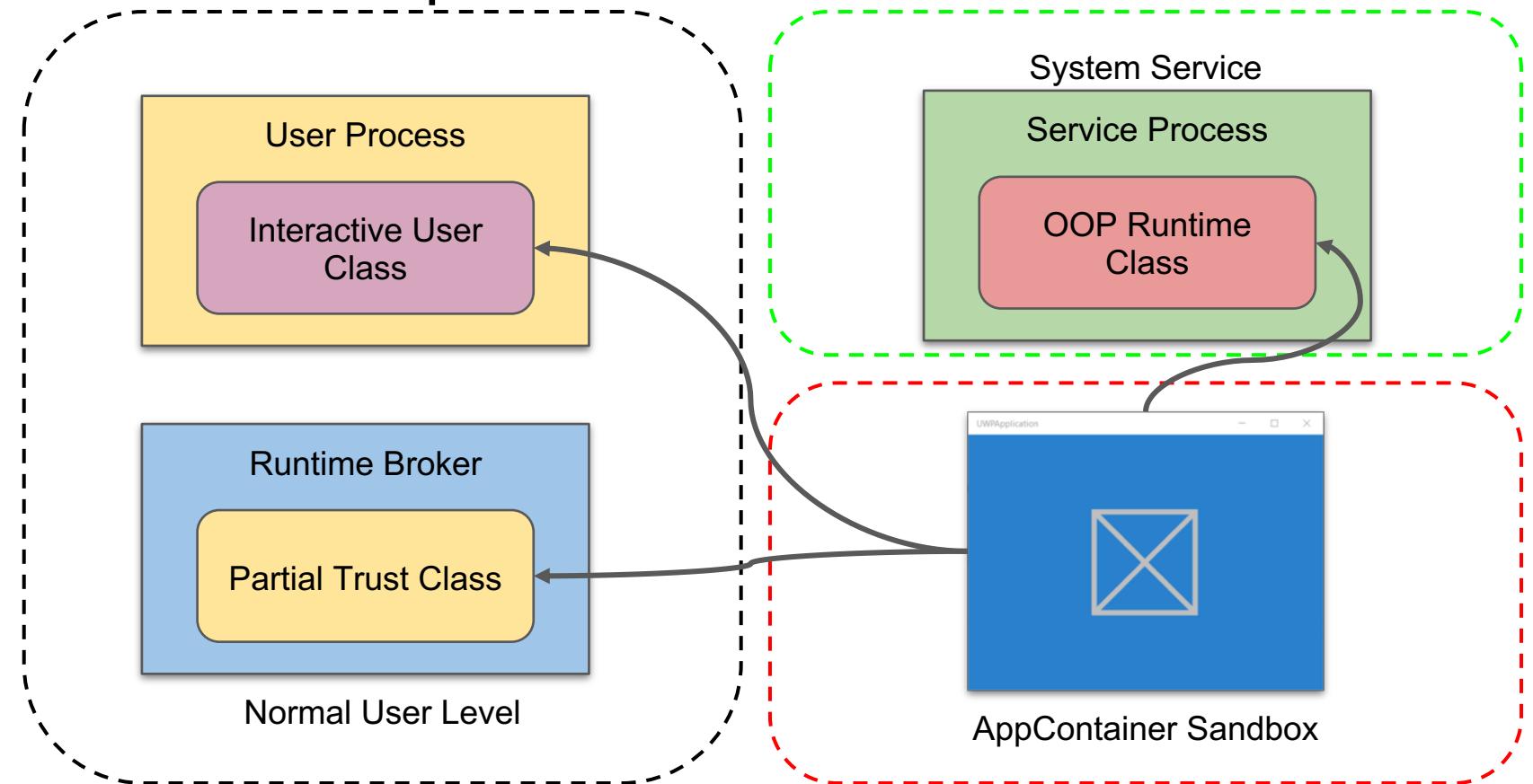
Enable debugging for a package

```
PS> plmdebug.exe /enableDebug PKGID DBGPATH.EXE
```

DEMO 3

Windows Runtime Security

Sandbox Escape OOP Attack Surface



Get list of partial trust classes

```
PS> Get-ComRuntimeClass -TrustLevel PartialTrust
```

Get Interactive User classes

```
PS> Get-ComRuntimeServer -IdentityType SessionUser `| Select -ExpandProperty Classes
```

Get svchost hosted classes

```
PS> Get-ComRuntimeServer -ServerType SvchostService `| Select -ExpandProperty Classes
```

Get EXE hosted classes

```
PS> Get-ComRuntimeServer -ServerType ExeService `| Select -ExpandProperty Classes
```

Partial Trust Class Default Permissions

```
PS> Show-ComSecurityDescriptor -RuntimeDefault
```

The screenshot shows the Windows Security Properties dialog for a COM class. It displays the following information:

Owner: NT AUTHORITY\SYSTEM
Group: NT AUTHORITY\SYSTEM
Integrity: Low

DACL **SACL**

ACL Entries

Type	Account	Access	Flags	Condition
Allowed	APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES	Execute, ExecuteLocal, ActivateLocal	None	
Allowed	NT AUTHORITY\SELF	Execute, ExecuteLocal, ActivateLocal	None	
Allowed	NT AUTHORITY\SYSTEM	Execute, ExecuteLocal, ActivateLocal	None	
Allowed	NT AUTHORITY\LOCAL SERVICE	Execute, ExecuteLocal, ActivateLocal	None	
Allowed	NT AUTHORITY\NETWORK SERVICE	Execute, ExecuteLocal, ActivateLocal	None	
AllowedCallback	NT AUTHORITY\INTERACTIVE	Execute, ExecuteLocal, ActivateLocal	None	!(WIN:/!IS)

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

Allows all AC at the same user to access the class.

Class Specific Permissions

```
PS> Show-ComSecurityDescriptor $cls
```

The screenshot shows the Windows Application Model App Extensions App Extension Catalog Launch Security dialog box. It displays the following information:

Owner: NT AUTHORITY\SYSTEM
Group: NT AUTHORITY\SYSTEM
Integrity: Low
DACL SACL

ACL Entries:

Type	Account	Access	Flags	Condit
Allowed	NAMED CAPABILITIES\!pac App Experience	Execute, ExecuteLocal, ActivateLocal	None	
Allowed	APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES	Execute, ExecuteLocal, ActivateLocal	None	
Allowed	NT AUTHORITY\SELF	Execute, ExecuteLocal, ActivateLocal	None	
Allowed	NT AUTHORITY\SYSTEM	Execute, ExecuteLocal, ActivateLocal	None	
Allowed	NT AUTHORITY\LOCAL SERVICE	Execute, ExecuteLocal, ActivateLocal	None	
Allowed	NT AUTHORITY\NETWORK SERVICE	Execute, ExecuteLocal, ActivateLocal	None	

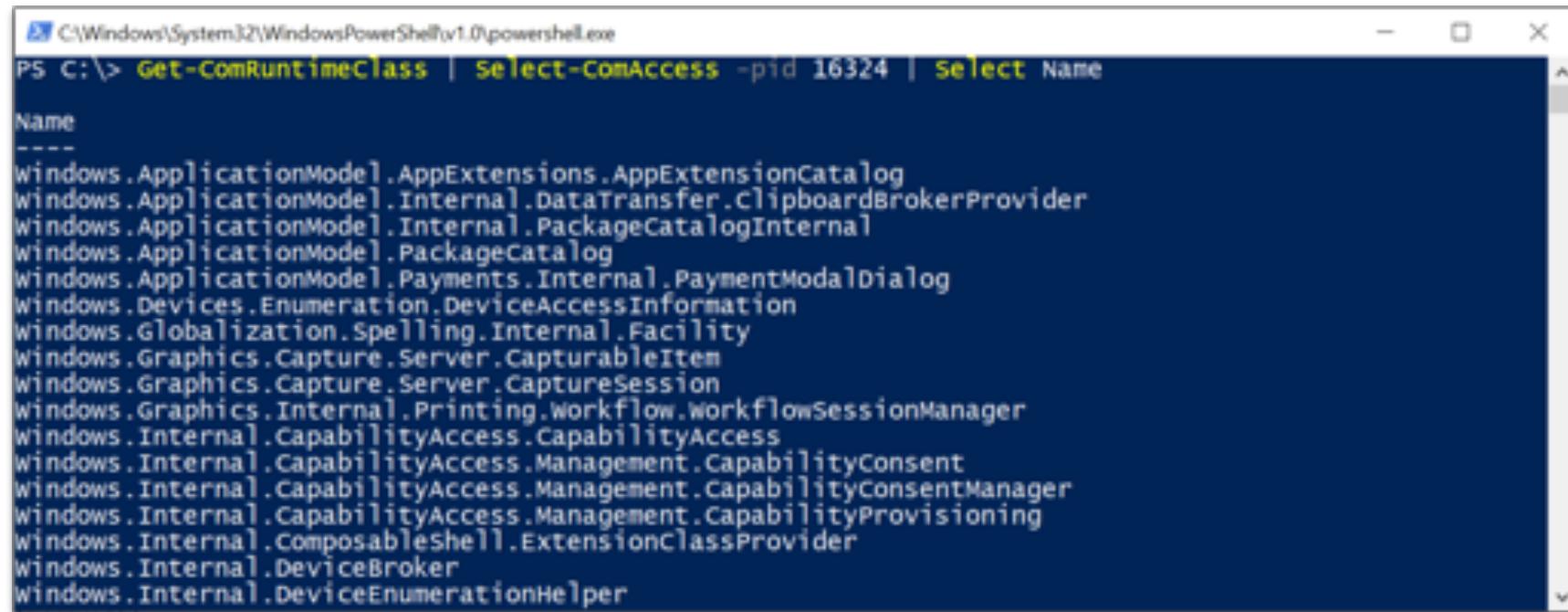
Specific Access:

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

Adds the
IpacAppExperience
capability

Finding Accessible Classes

```
PS> Get-ComRuntimeClass | Select-ComAccess -pid X
```



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

```
PS C:\> Get-ComRuntimeClass | Select-ComAccess -pid 16324 | Select Name
```

The output lists various Windows COM classes, all of which are currently marked as accessible (indicated by the 'A' character in the status bar of the PowerShell window). The classes listed are:

- Windows.ApplicationModel.AppExtensions.AppExtensionCatalog
- Windows.ApplicationModel.Internal.DataTransfer.ClipboardBrokerProvider
- Windows.ApplicationModel.Internal.PackageCatalogInternal
- Windows.ApplicationModel.PackageCatalog
- Windows.ApplicationModel.Payments.Internal.PaymentModalDialog
- Windows.Devices.Enumeration.DeviceAccessInformation
- Windows.Globalization.Spelling.Internal.Facility
- Windows.Graphics.Capture.Server.CapturableItem
- Windows.Graphics.Capture.Server.CaptureSession
- Windows.Graphics.Internal.Printing.Workflow.WorkflowSessionManager
- Windows.Internal.CapabilityAccess.CapabilityAccess
- Windows.Internal.CapabilityAccess.Management.CapabilityConsent
- Windows.Internal.CapabilityAccess.Management.CapabilityConsentManager
- Windows.Internal.CapabilityAccess.Management.CapabilityProvisioning
- Windows.Internal.ComposableShell.ExtensionClassProvider
- Windows.Internal.DeviceBroker
- Windows.Internal.DeviceEnumerationHelper

Package Name Checks

```
BOOL BrokerAuthenticateCOMCaller() {
    HANDLE token;
    CoImpersonateClient();
    OpenThreadToken(GetCurrentThread(), TOKEN_QUERY, &token);
    WCHAR family_name[255];
    ULONG family_name_length = 255;
    NTSTATUS status = RtlQueryPackageClaims(token,
                                             family_name, &family_name_length);
    if (NT_SUCCESS(status))
        return wcsicmp(package_name, L"MicrosoftEdge") == 0;
    return FALSE;
}
```

Reads from
WIN://SYSAPPID

Incorrect Capability or Missing Security Checks

```
HANDLE CheckedCreateFile(string path) {  
    // Get client token.  
    HANDLE token;  
    CoImpersonateClient();  
    OpenThreadToken(GetCurrentThread(), &token);  
  
    HANDLE ret = INVALID_HANDLE_VALUE;  
    if (CapabilityCheck(token, L"internetClient")) {  
        ret = CreateFile(path, ...);  
    }  
    return ret;  
}
```

But opening a file.

Checking for
internetClient capability

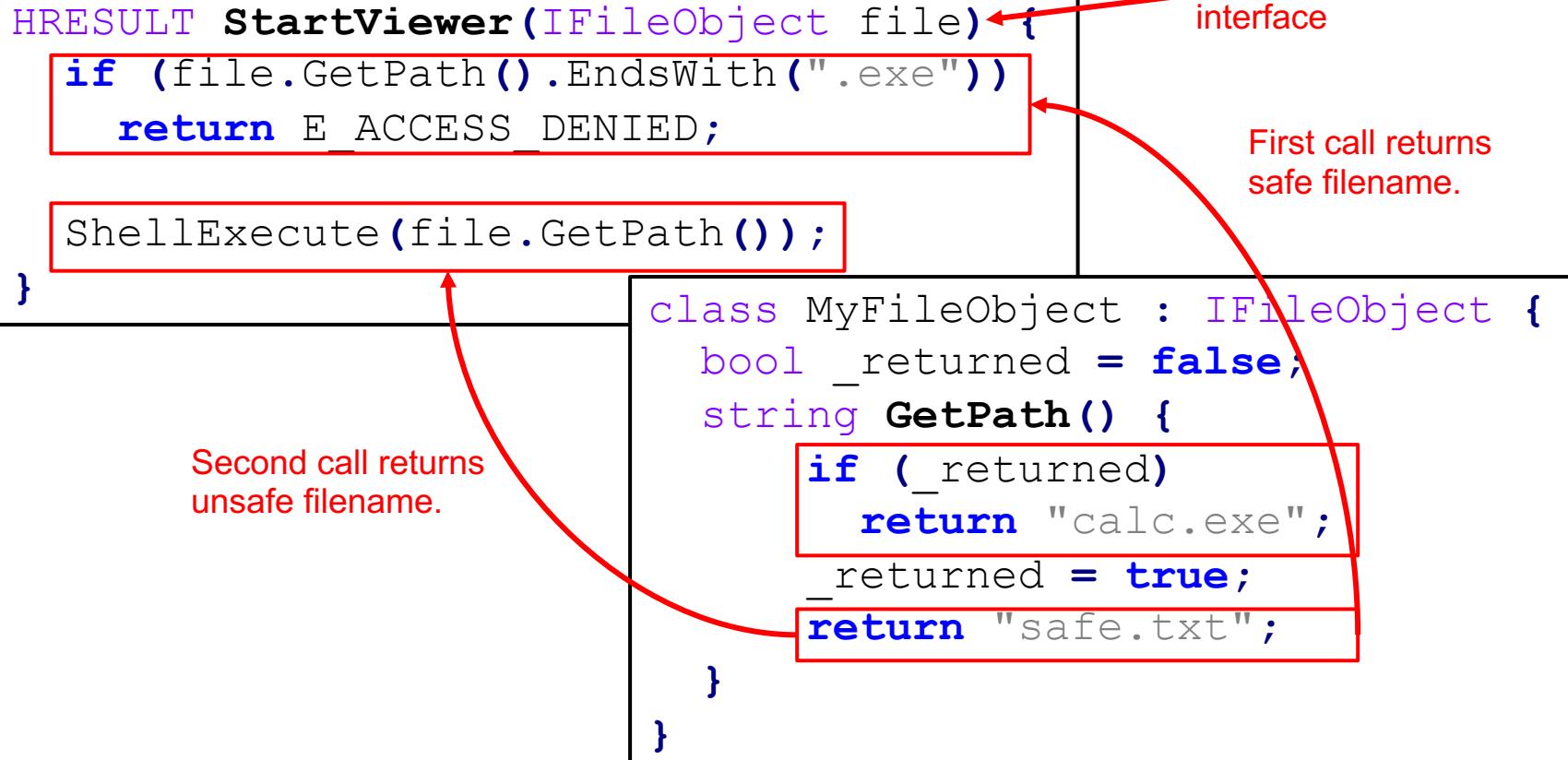
HSTRING is a Counted String

```
UINT32 length;  
PCWSTR str = WindowsGetStringRawBuffer(hString,  
                                         &length);  
// Might not be equal.  
assert(wcslen(str) == length);
```



```
HRESULT WindowsStringHasEmbeddedNull(  
    HSTRING string,  
    BOOL     *hasEmbedNull  
);
```

TOCTOU in Marshaled Interfaces



Inject a DLL Into Running Process

```
c:\windows\system32\WindowsPowerShell\v1.0\powershell.exe
PS D:\> Get-NtProcessMitigations -Name 'MicrosoftEdge.exe' | Select *Signed*
MicrosoftSignedOnly      : False
StoreSignedOnly           : True
SignedMitigationOptIn    : True
AuditMicrosoftSignedOnly : False
AuditStoreSignedOnly     : False

PS D:\> Get-NtProcessMitigations -Name 'Calculator.exe' | Select *Signed*
MicrosoftSignedOnly      : False
StoreSignedOnly           : True
SignedMitigationOptIn    : False
AuditMicrosoftSignedOnly : False
AuditStoreSignedOnly     : False
```

Only Store signed DLLs can be loaded

DEMO 4

Conclusions

- All based on familiar COM programming paradigms
- The Windows Runtime has many interesting attack surfaces
 - Attack surface which might be accessible remotely
 - Plenty of Sandbox to User and User to System privilege escalation routes
- Tooling is not quite there, making an effort with OleViewDotNet

謝謝