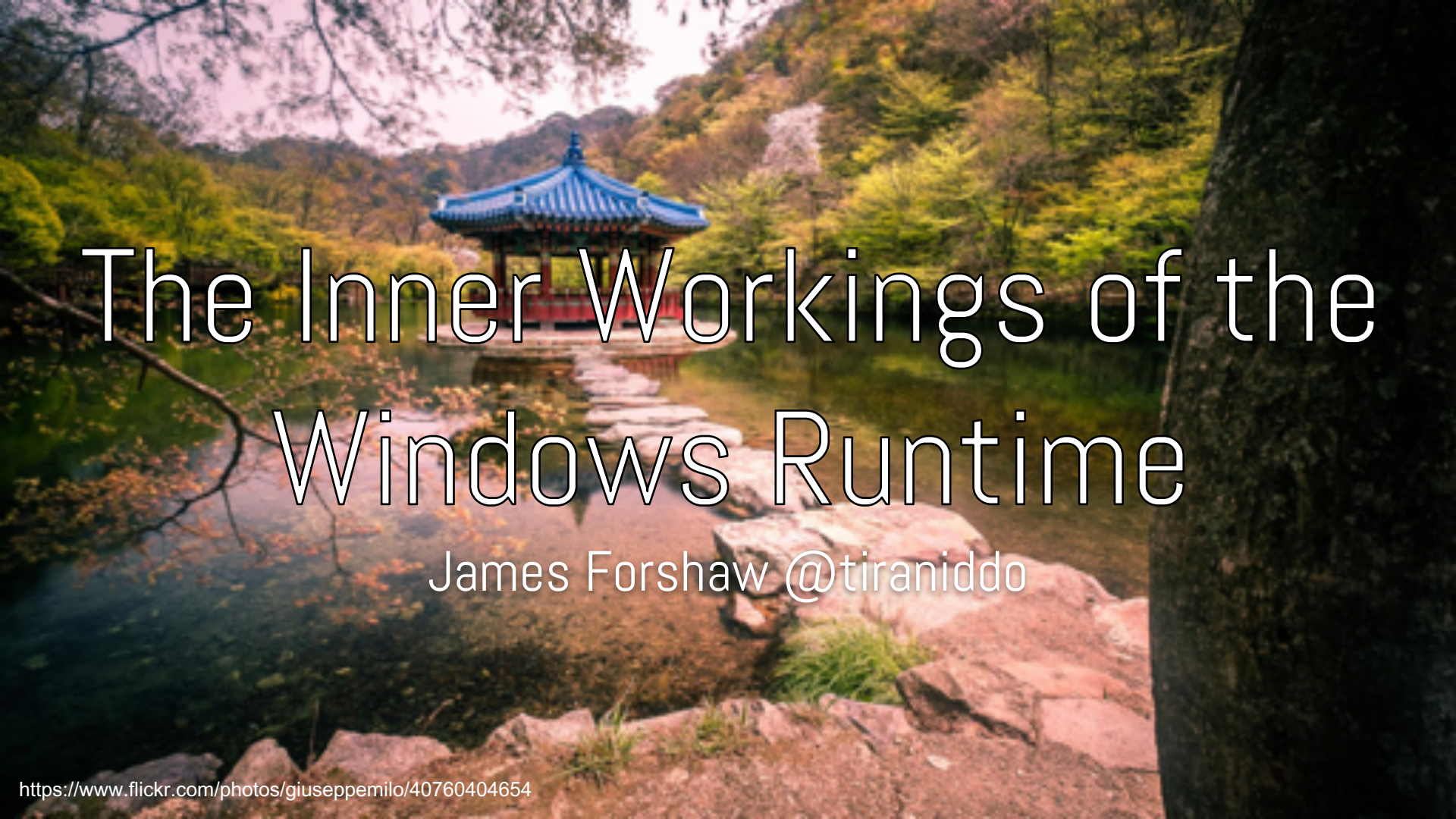


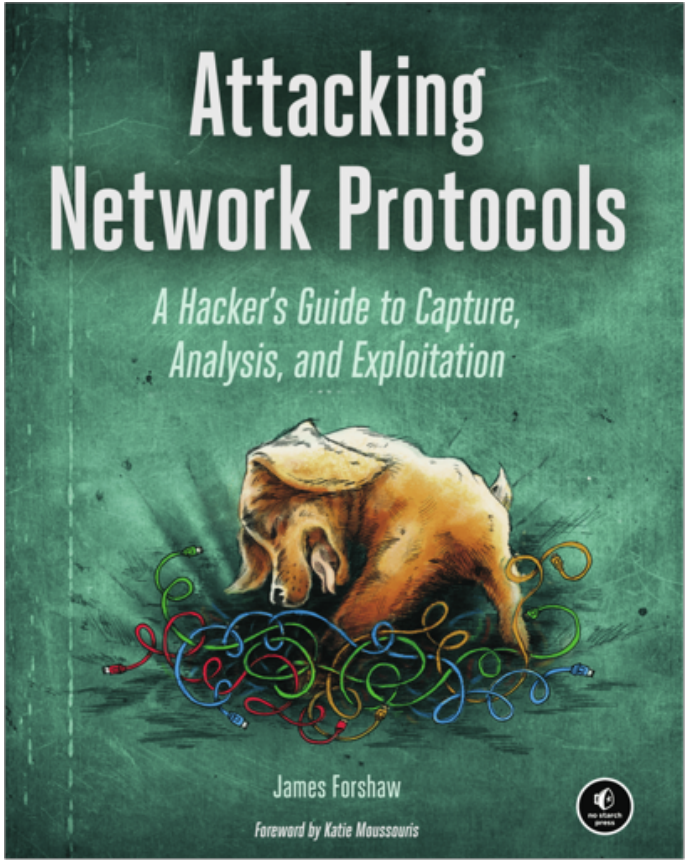
The Inner Workings of the Windows Runtime

James Forshaw @tiraniddo

https://www.flickr.com/photos/giuseppemilo/40760404654

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

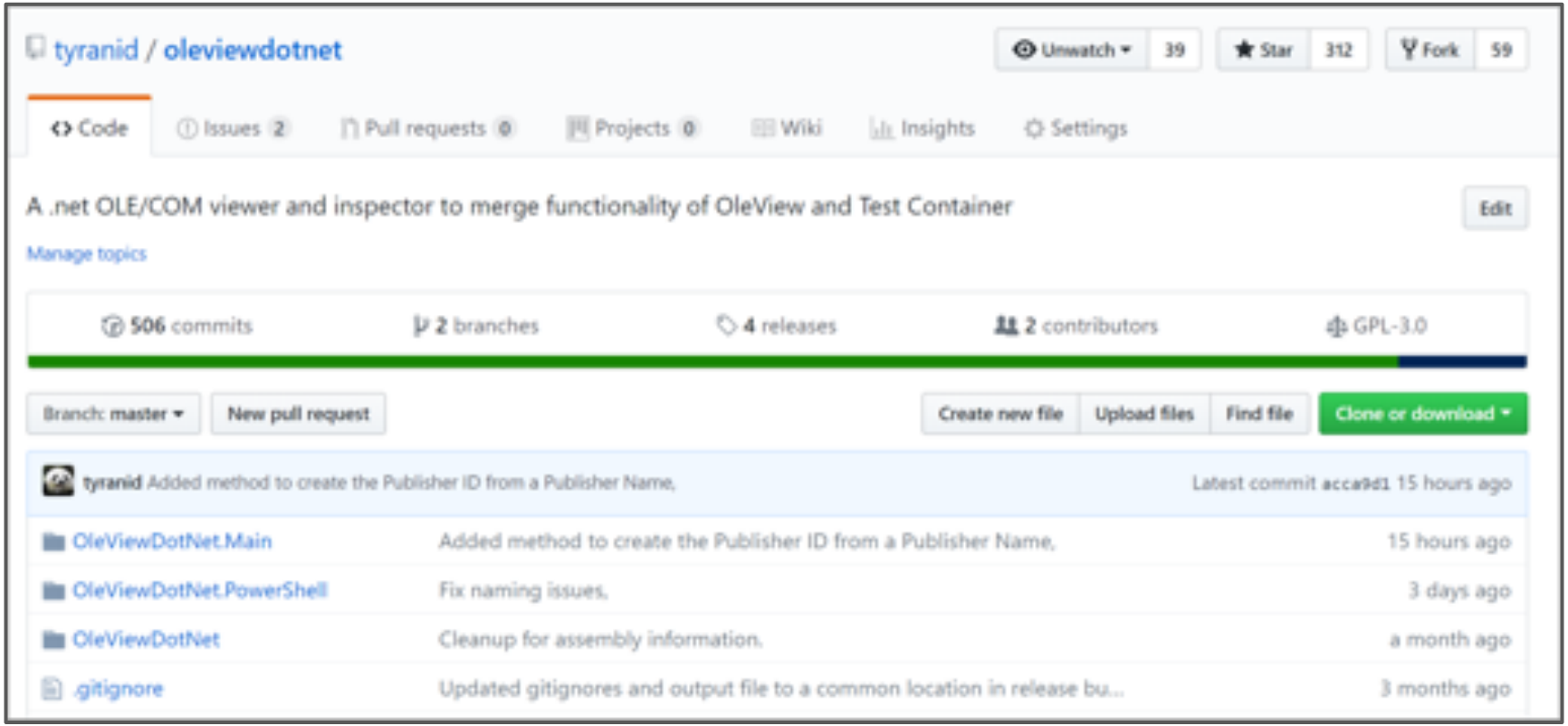
Background Research

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

https://conference.hitb.org/hitbsecconf2012ams/materials/D1T2%20-%20Sebastien%20Renaud%20and%20Kevin%20Szkudlapski%20-%20WinRT.pdf

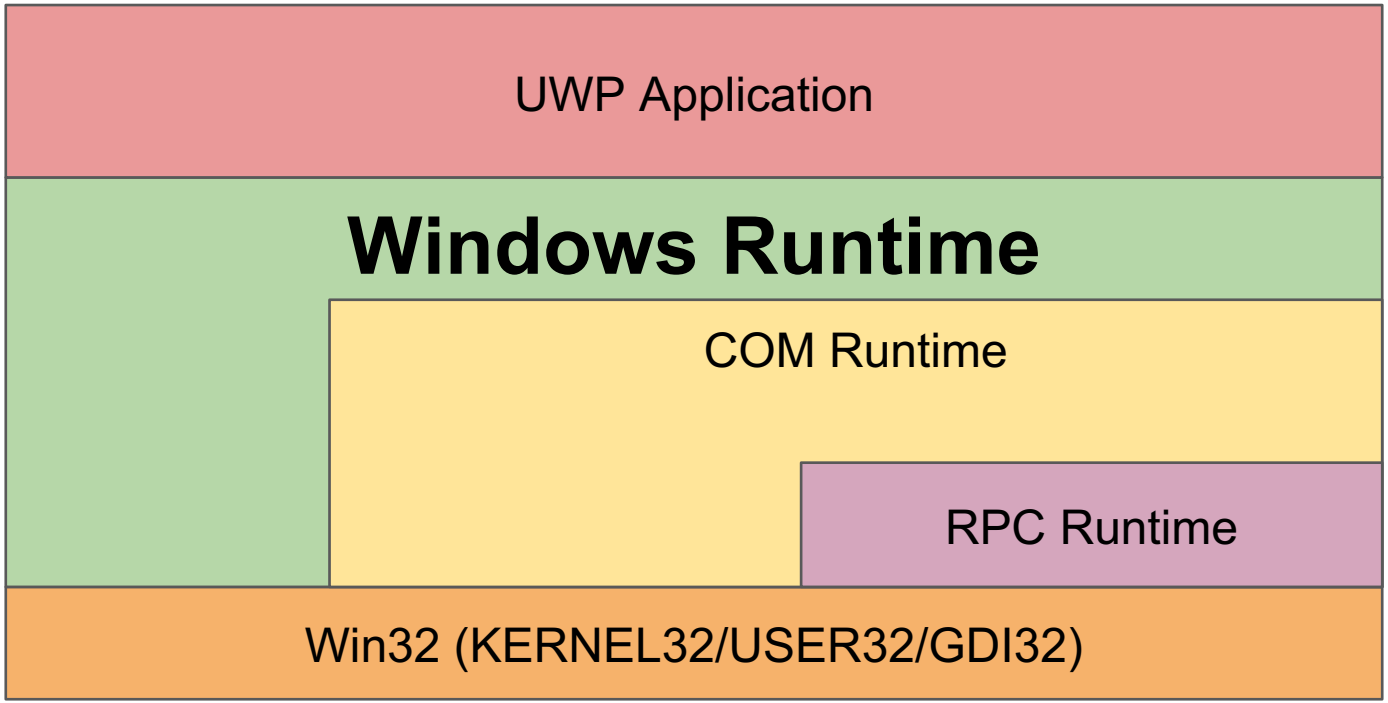
This Talk is based on Windows 10 **1803/1809**

OleViewDotNet

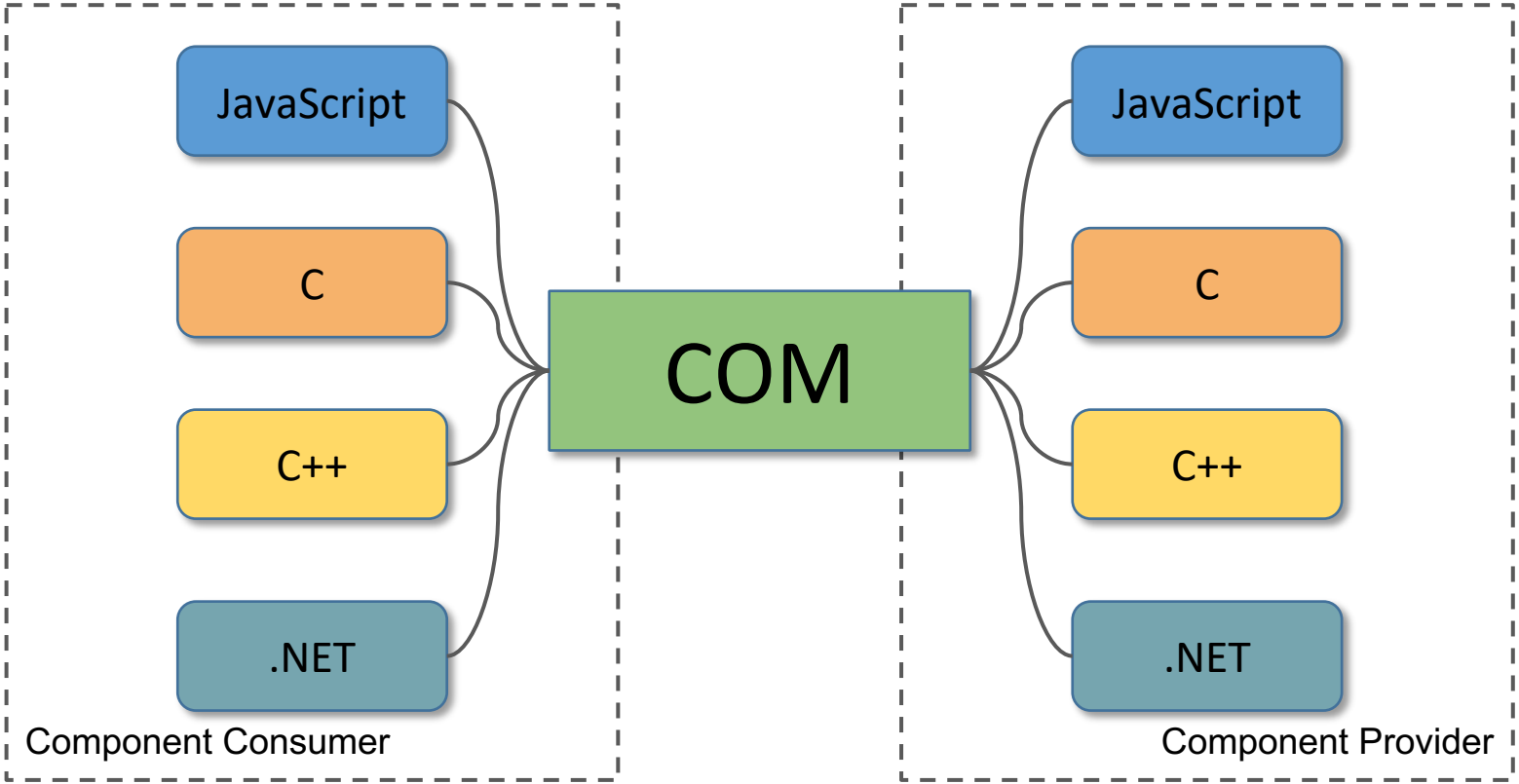


https://github.com/tyranid/oleviewdotnet

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
  );
};
```

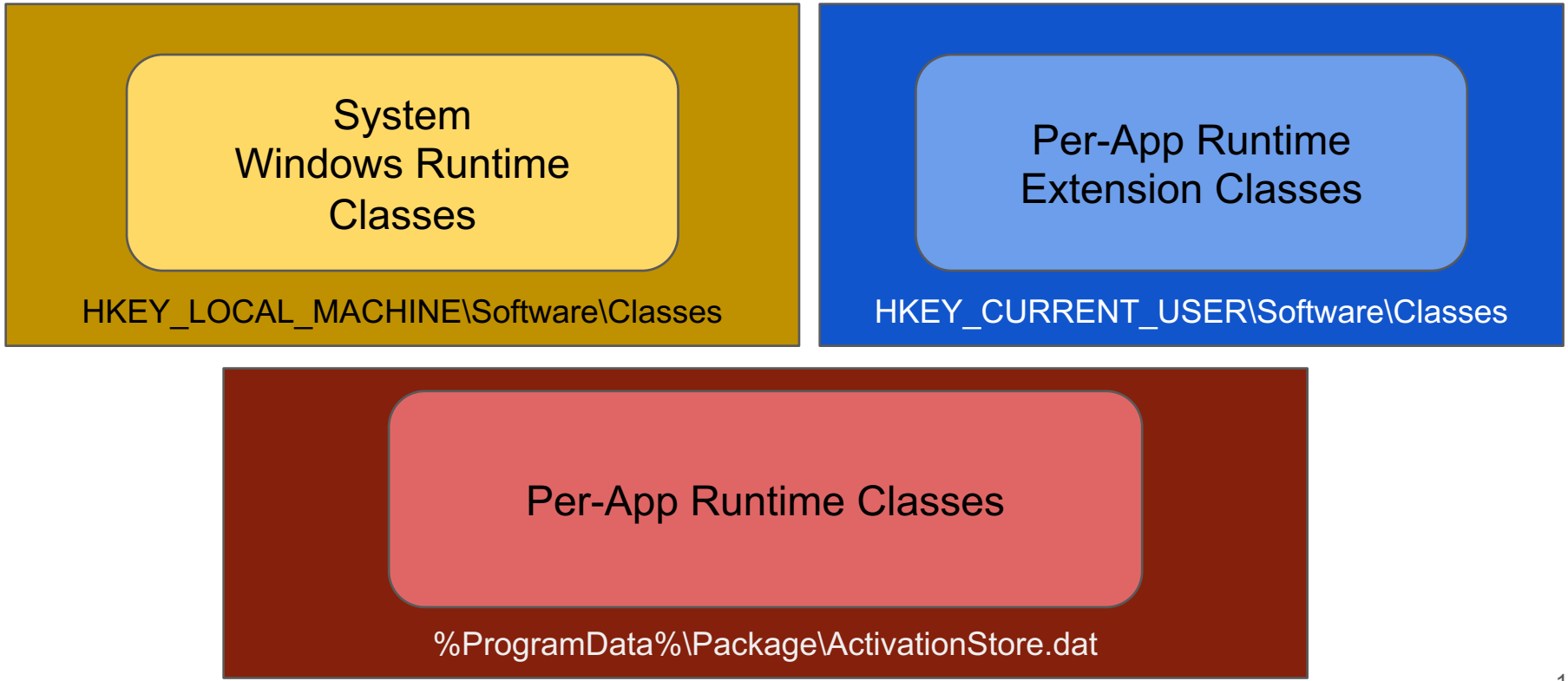
Activation Factories and Instances

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


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

Example ACID: **“Windows.Foundation.Uri”**

Runtime Class Registry Keys



Runtime Extension Classes

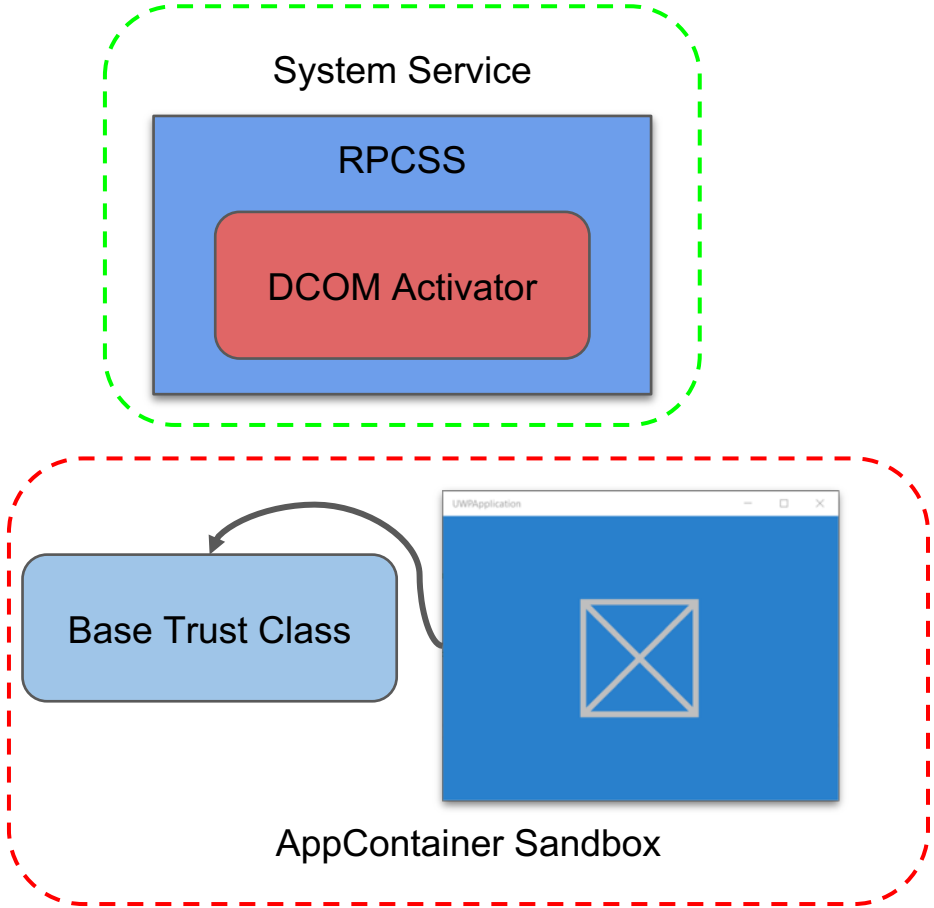
Contract ID	***Description***
Windows.Launch	Default Application Launch
Windows.Protocol	URI Protocol Handler
Windows.BackgroundTasks	Background Task
Windows.File	Launch and pass a file object
Windows.Search	Search request

Class Trust Levels

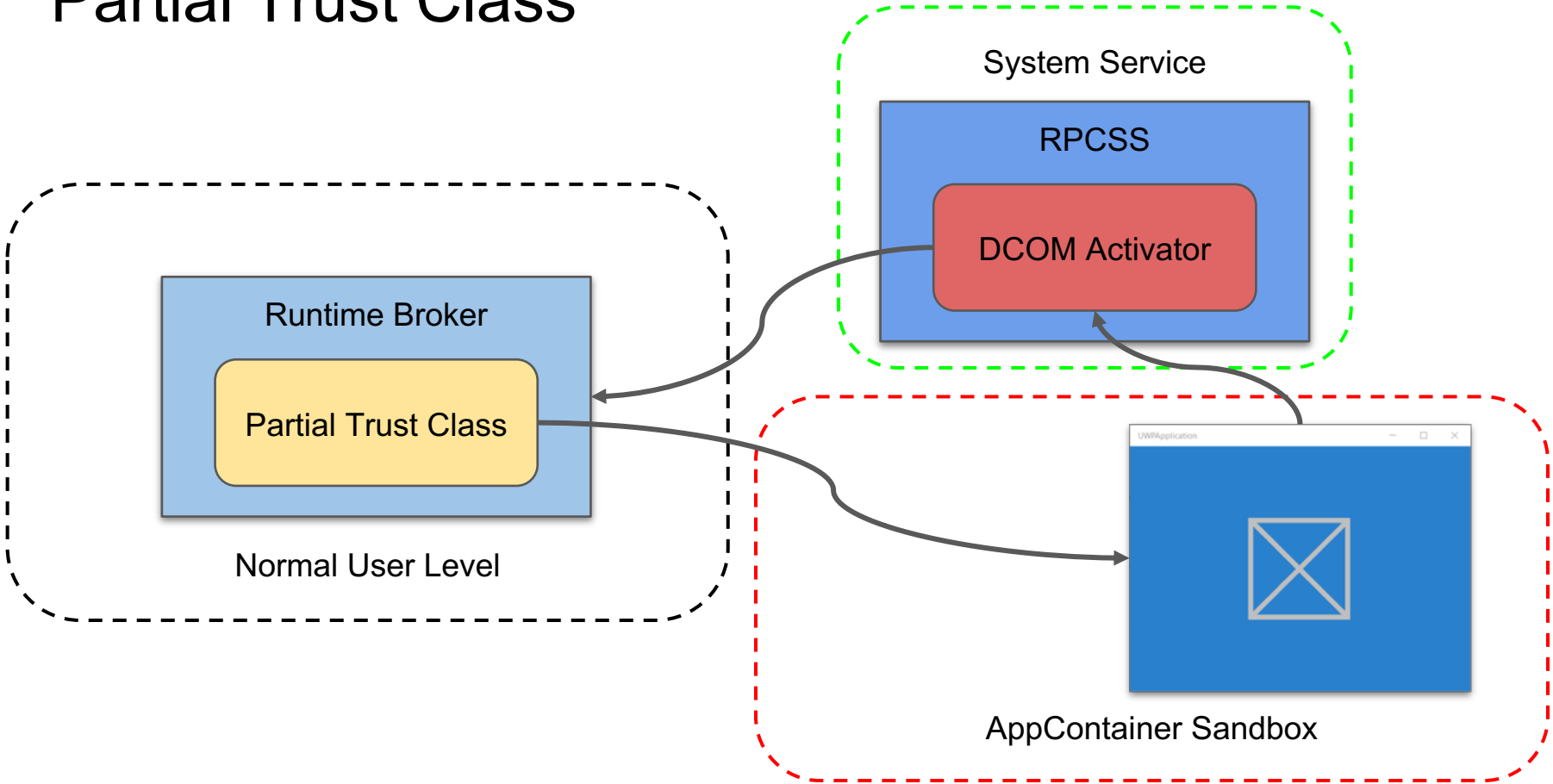
```
HRESULT GetTrustLevel(TrustLevel *trustLevel);
```

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

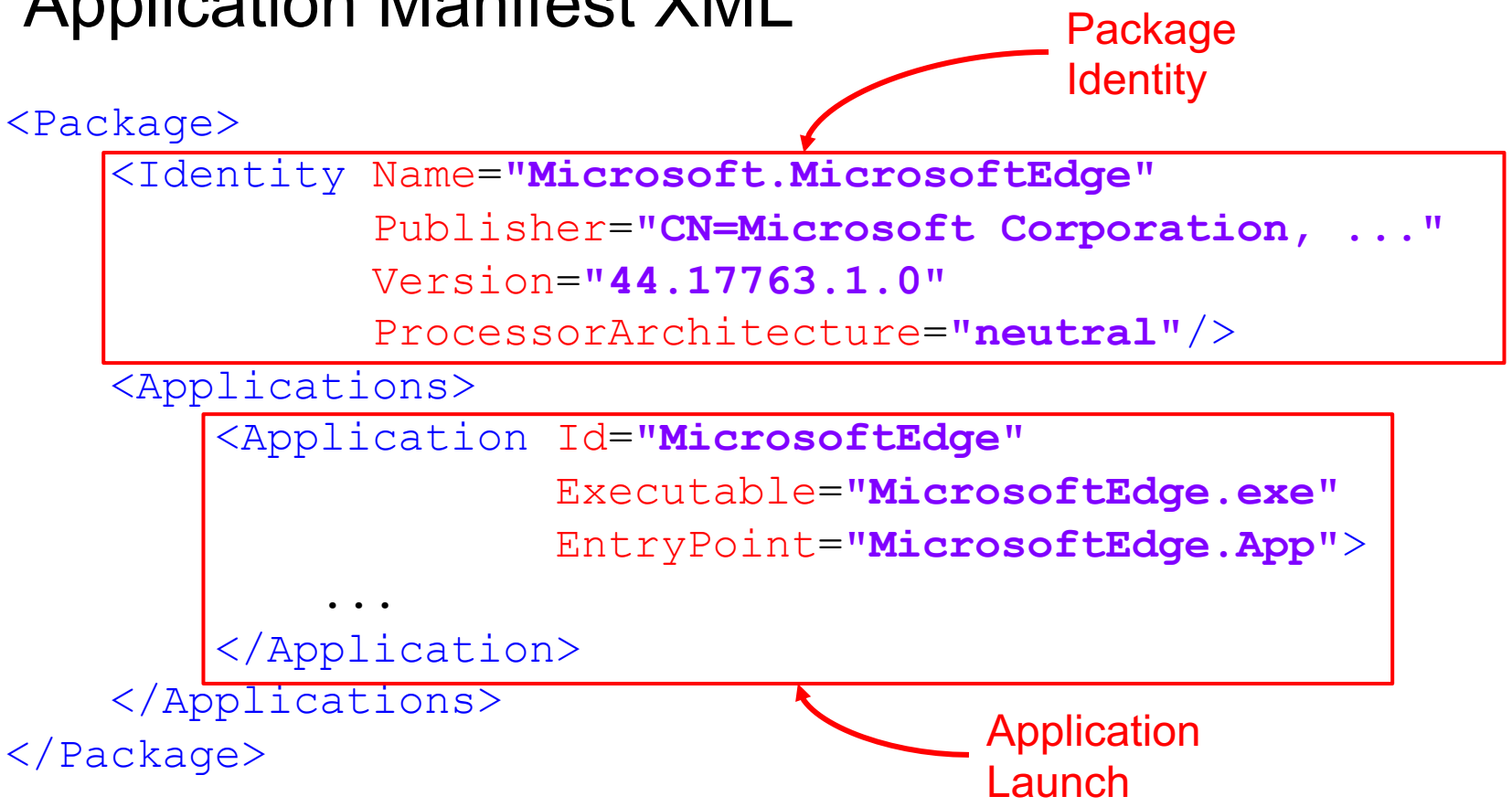


Partial Trust Class

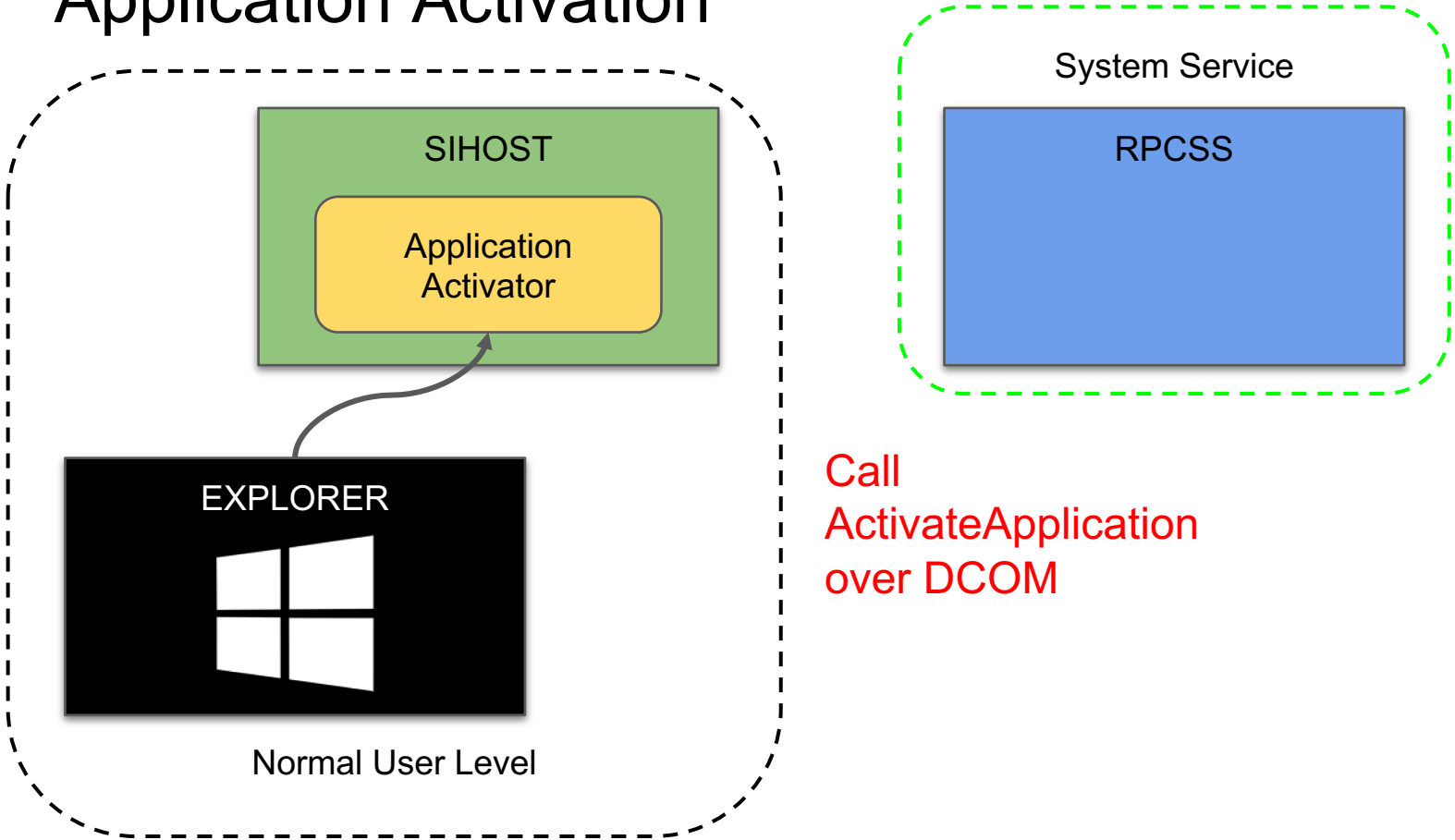


DEMO 1

Application Manifest XML

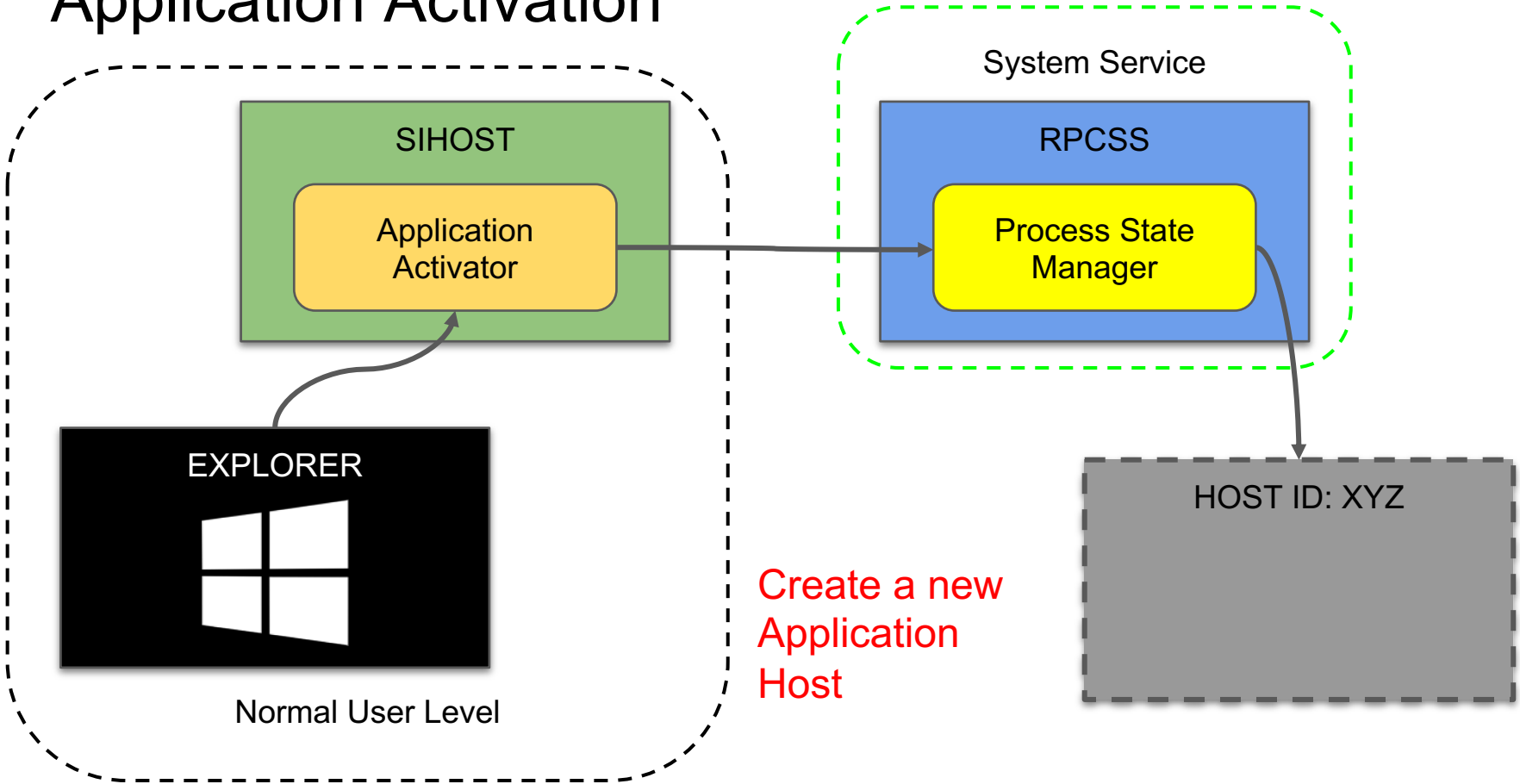


Application Activation

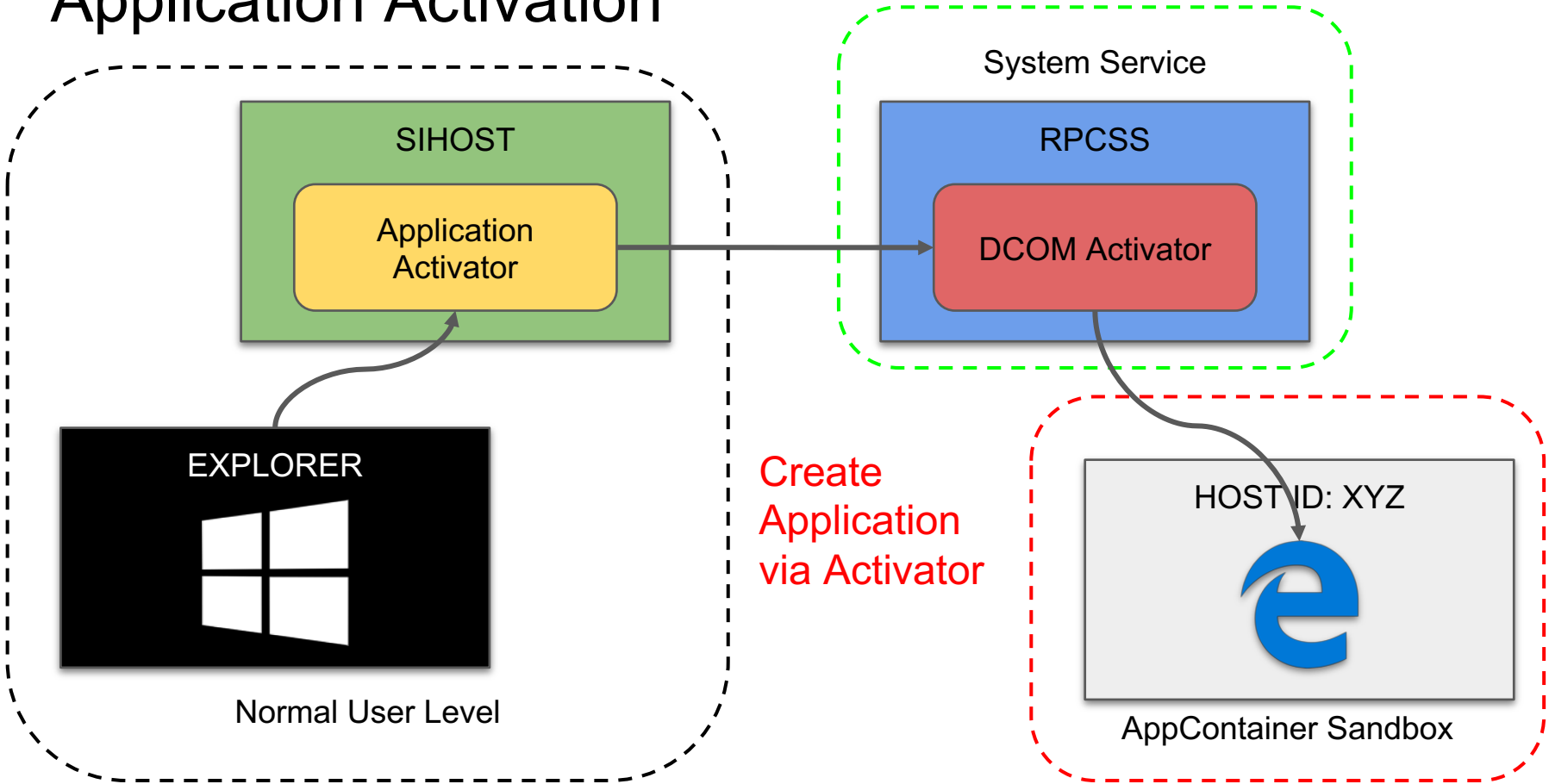


Call ActivateApplication over DCOM

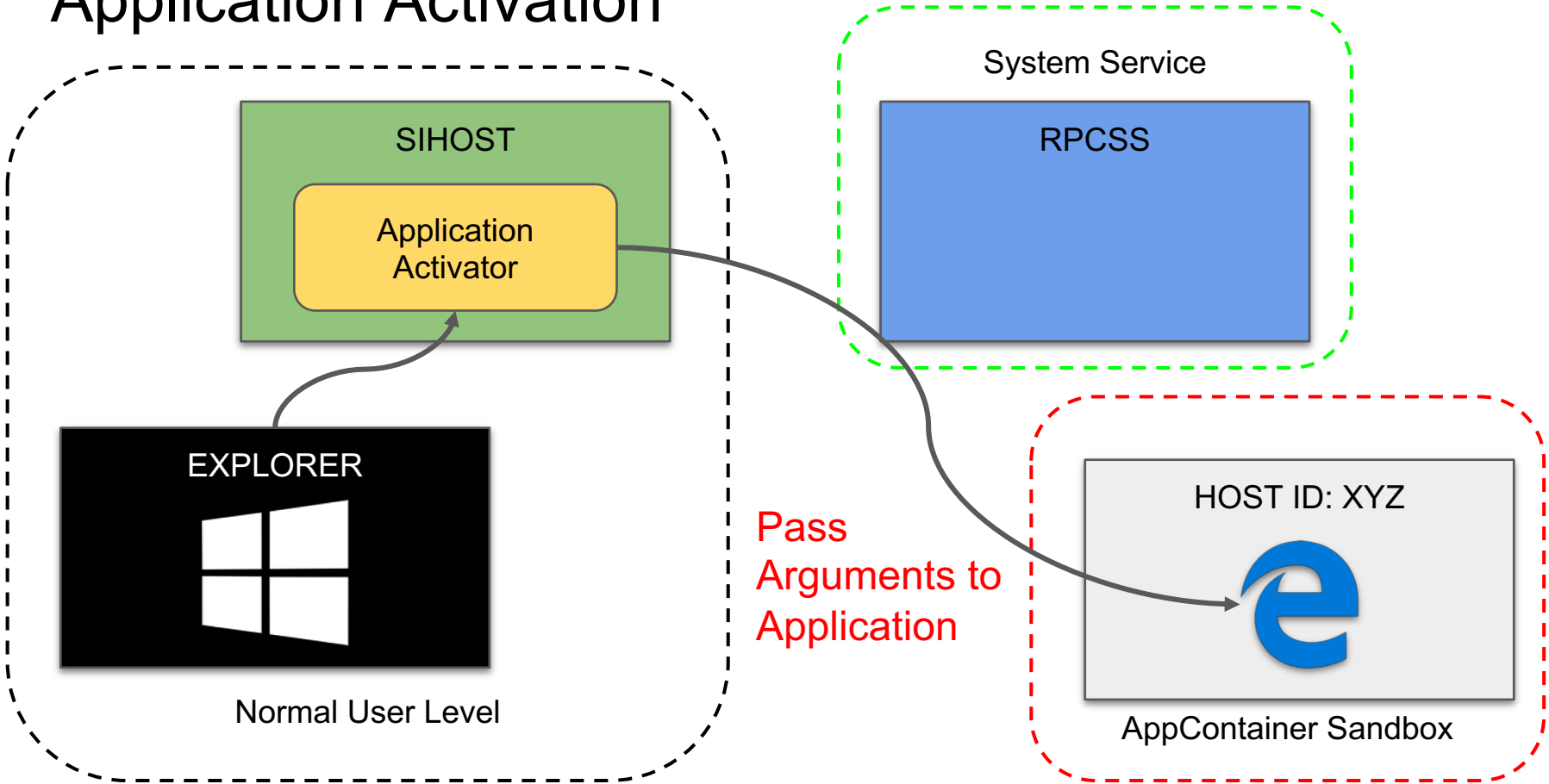
Application Activation



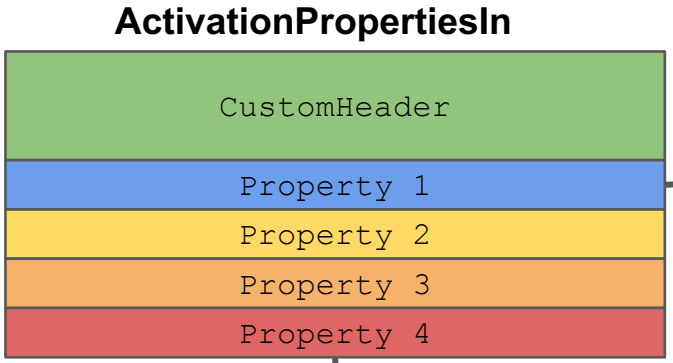
Application Activation



Application Activation



WinRT Activation Properties




```
struct ComWinRTActivationPropertiesData {
  HSTRING    activatableClassId;
  HSTRING    packageFullName;
  ULONGLONG  userContext;
  PBLOB      rtbProcessMitigationPolicyBlob;
};
```

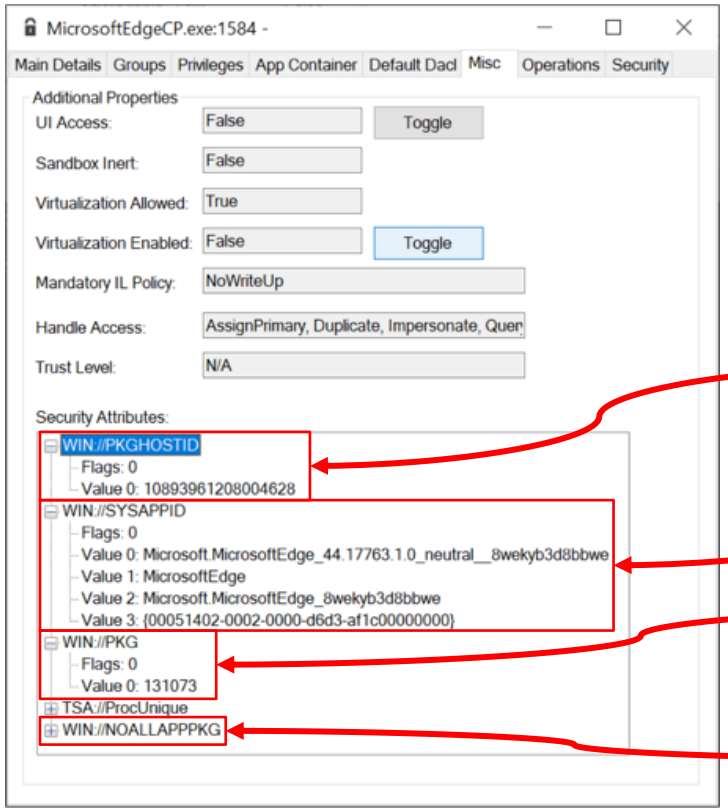
```
struct ExtensionActivationContextPropertiesData {
  ULONGLONG hostId;
  ULONGLONG userContext;
  GUID      componentProcessId;
  ULONGLONG racActivationTokenId;
  PBLOB     lpacAttributes;
  ULONGLONG consoleHandlesId;
  ULONGLONG aamActivationId;
};
```

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_xxxxxxxxx", "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

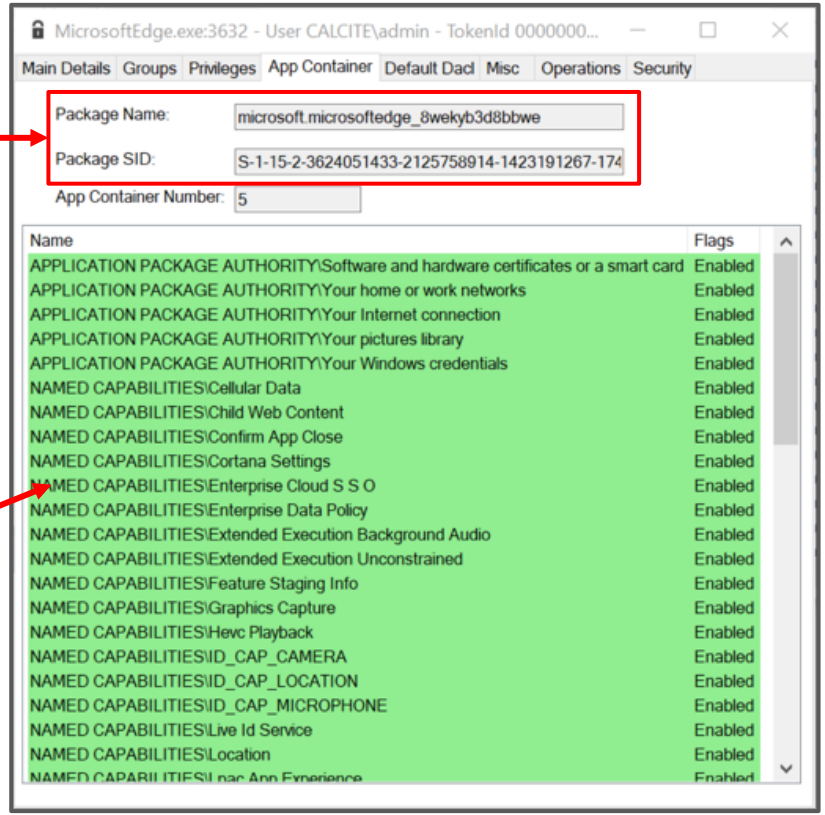
Building the System Application ID

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

AppContainer SID and Capabilities

Package Family Name and Package SID

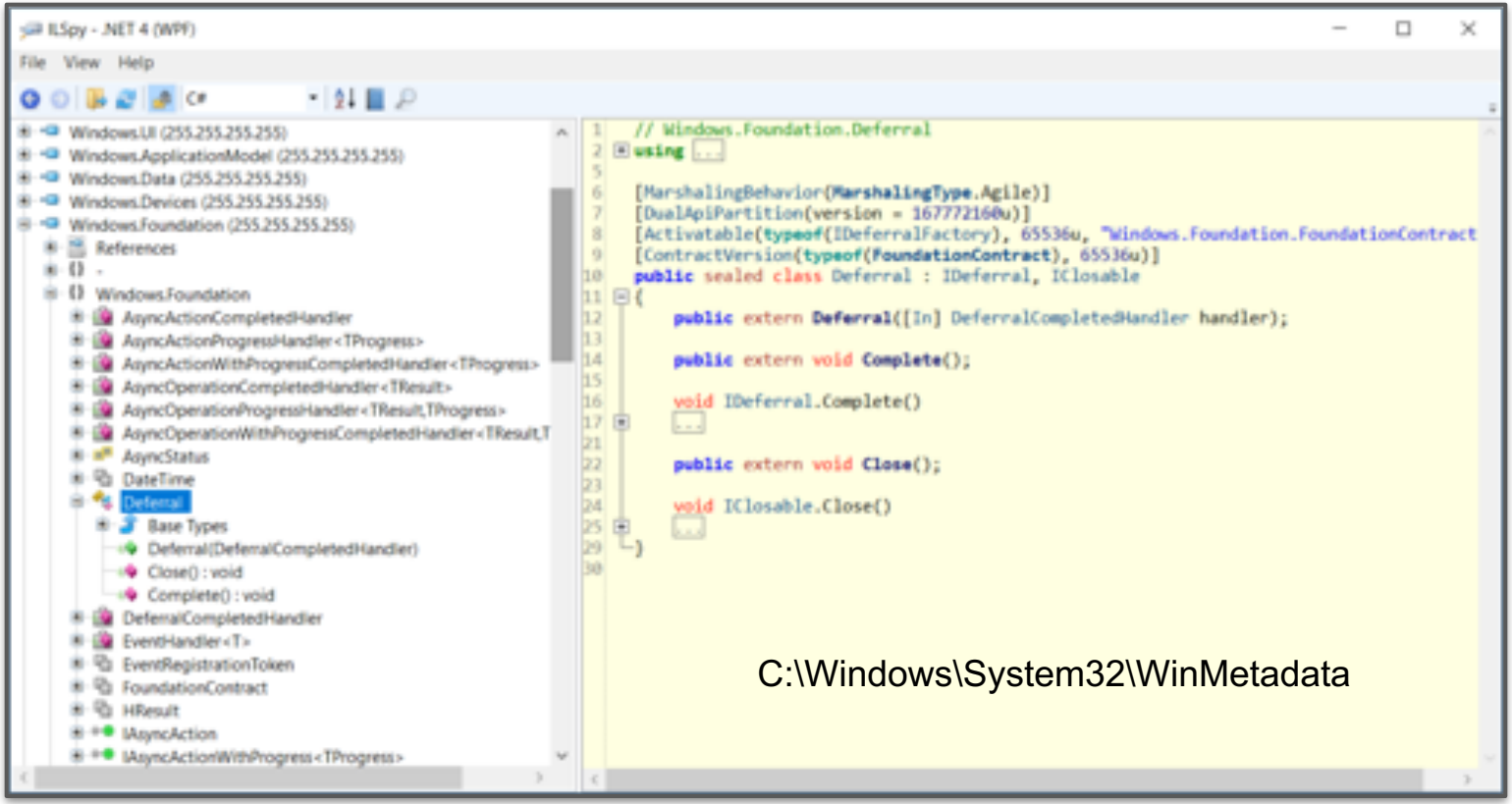
```
<Capabilities>
 <Capability Name="internetClient"/>
 <Capability Name="privateNetworkClientServer"/>
 <rescap:Capability Name="childWebContent"/>
 <rescap:Capability Name="confirmAppClose"/>
 <rescap:Capability Name="lpacCom"/>
 ...
 <DeviceCapability Name="location"/>
 <DeviceCapability Name="microphone"/>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



DEMO 2

Reverse Engineering Native Components

Windows Metadata



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

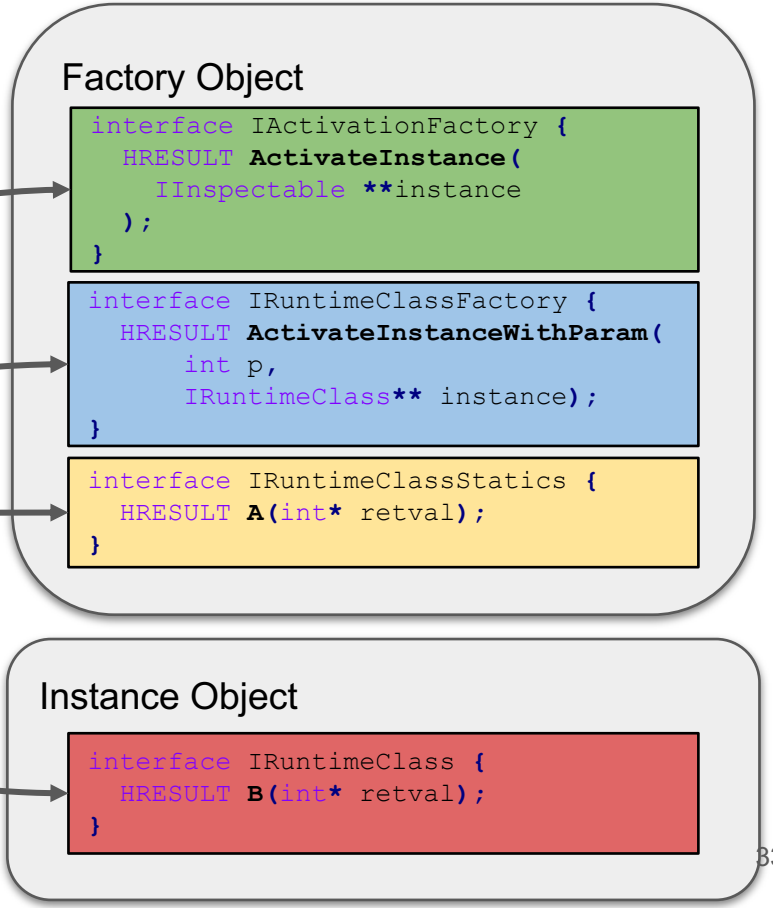
```
interface IActivationFactory {
  HRESULT ActivateInstance(
    IInspectable **instance
  );
}
```

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();
}
```



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

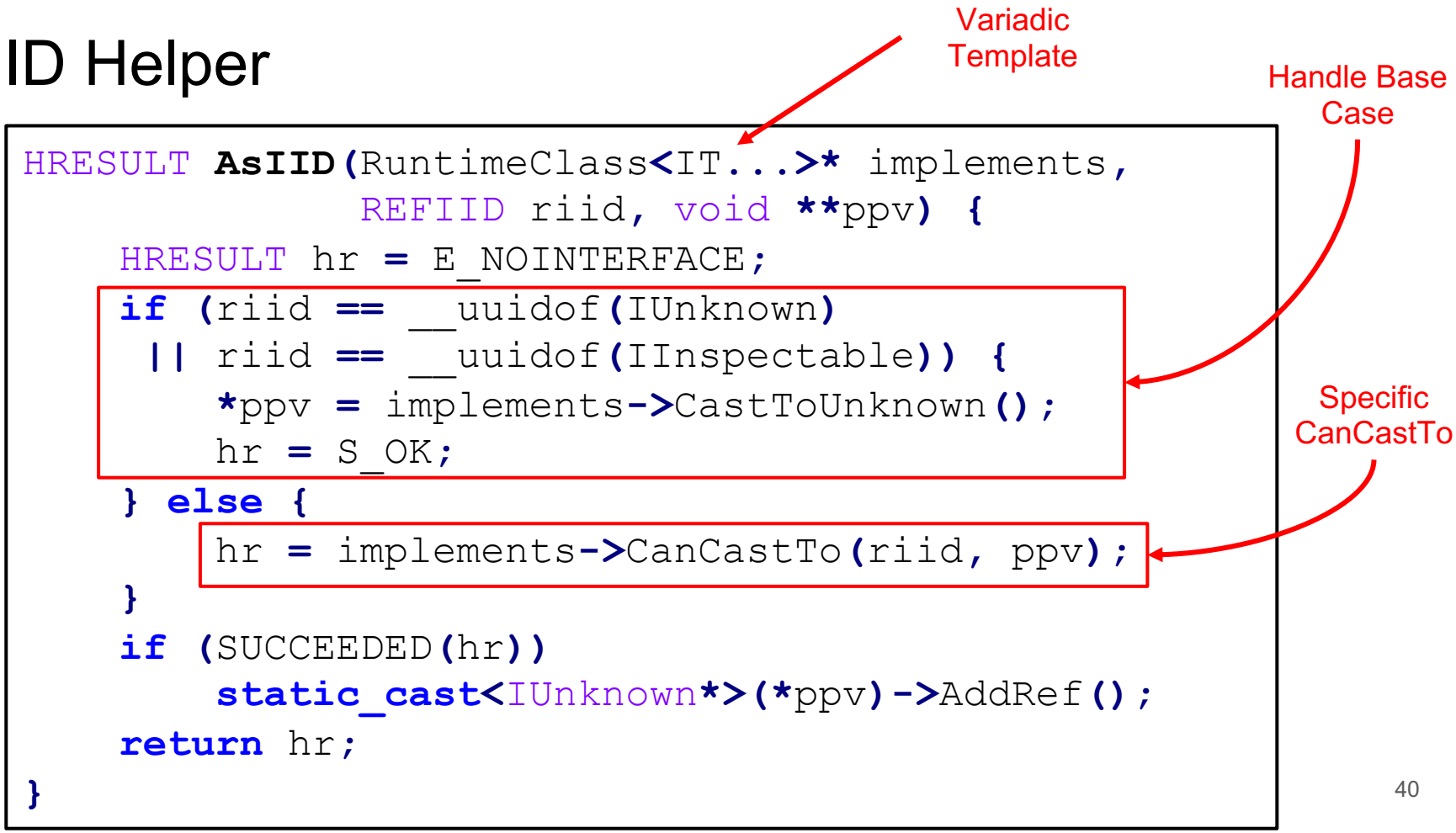
```
HRESULT QueryInterface(REFIID riid, void** ppv) {
    bool handled = false;
    HRESULT hr = CustomQueryInterface(riid, ppv, &handled);
    if (FAILED(hr) || handled)
        return hr;

    return Super::AsIID(this, riid, ppv);
}
```

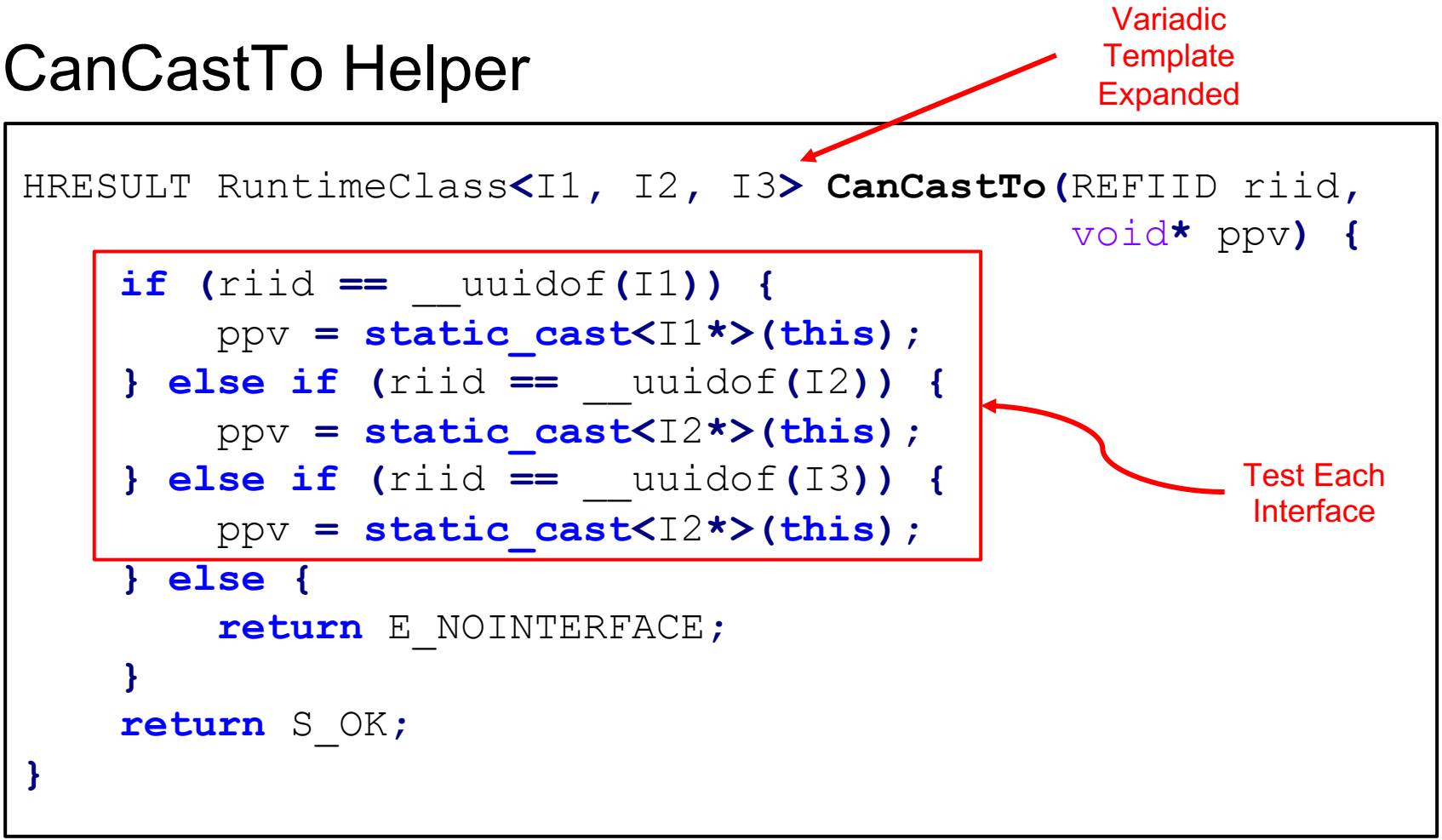
Overridable Custom QI

Call AsIID helper method

AsIID Helper



CanCastTo Helper



String Handles (HSTRING)

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

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

Opaque string handle structure.

```
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;
  unsigned int padding1;
  unsigned int padding2;
  const wchar_t *stringRef;
};
```

Used for stack scoped "reference" strings.

```
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
```

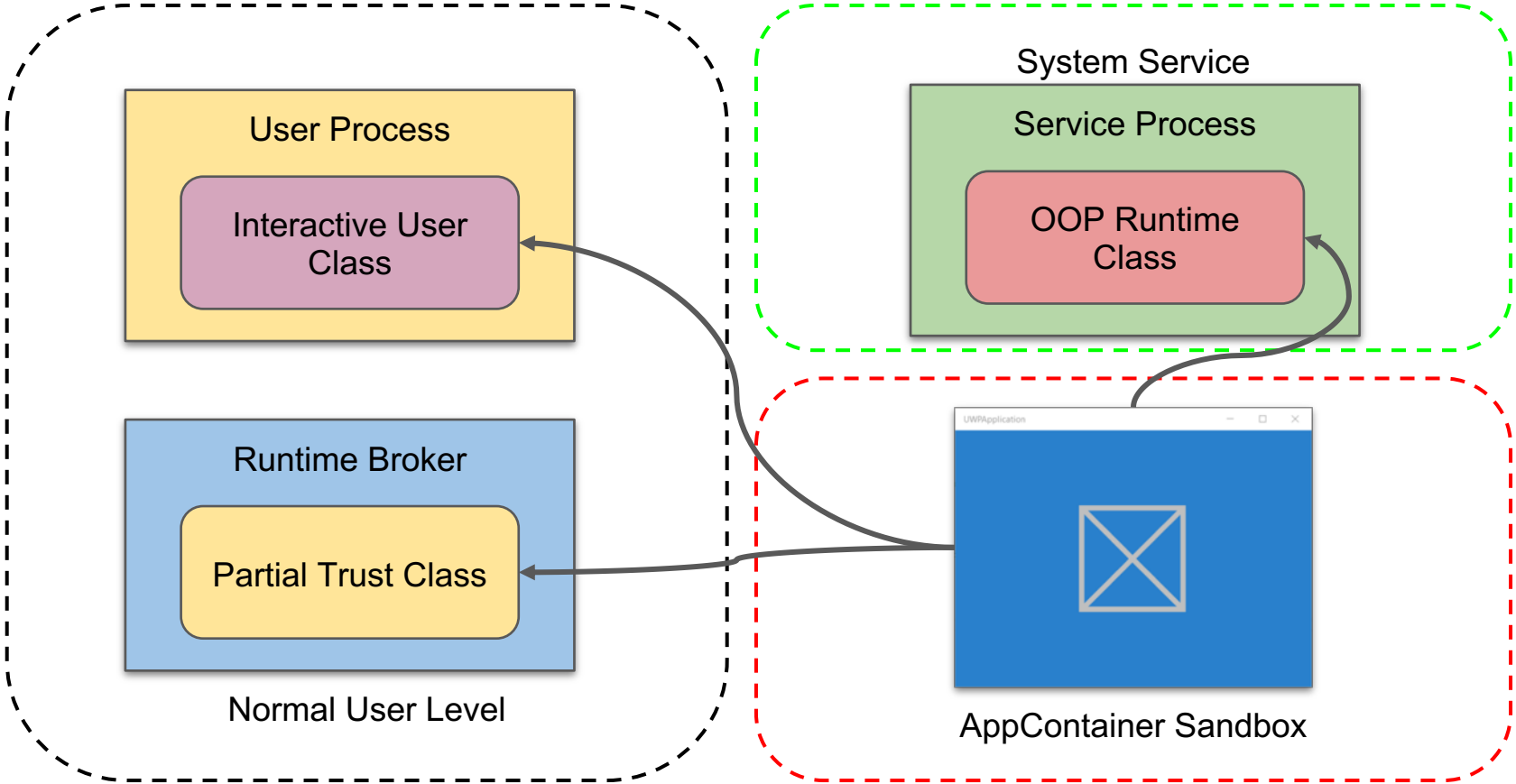
Enable debugging for a package

```
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

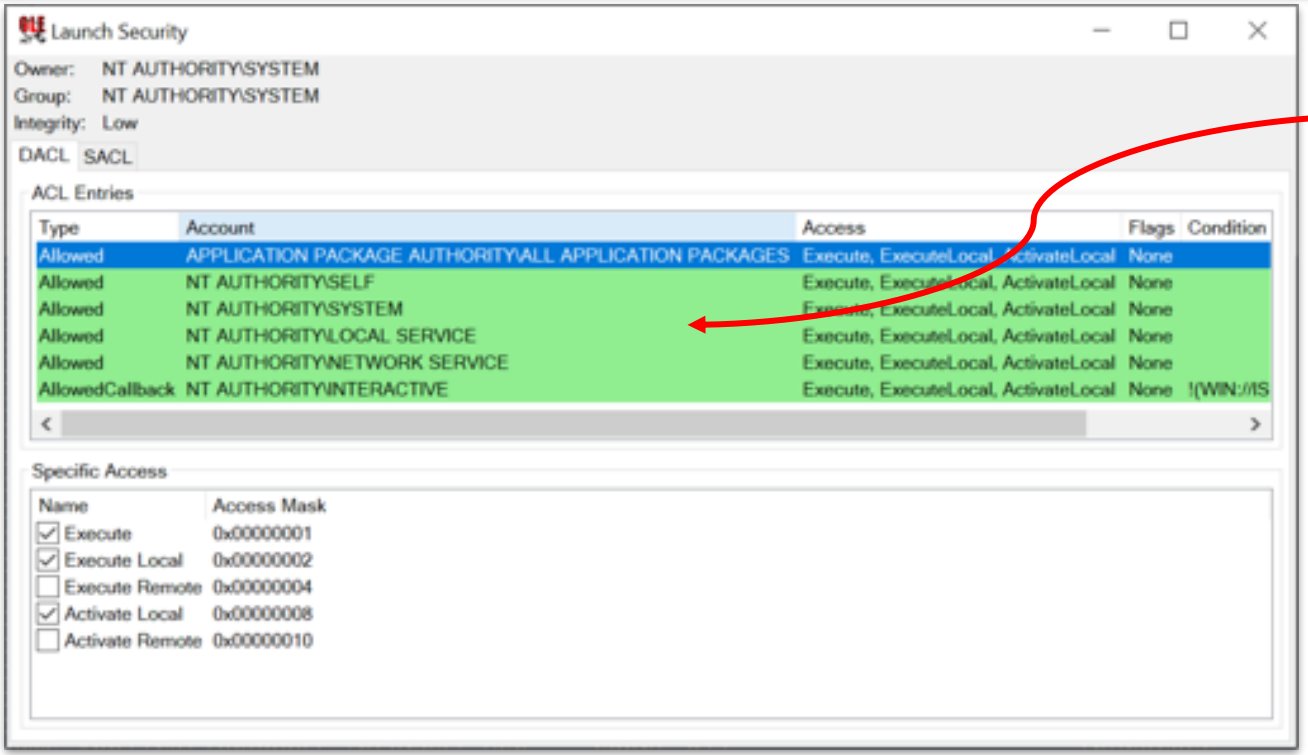
```
PS> Get-ComRuntimeServer -ServerType SvchostService `
    | Select -ExpandProperty Classes
```

Get EXE hosted classes

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

Partial Trust Class Default Permissions

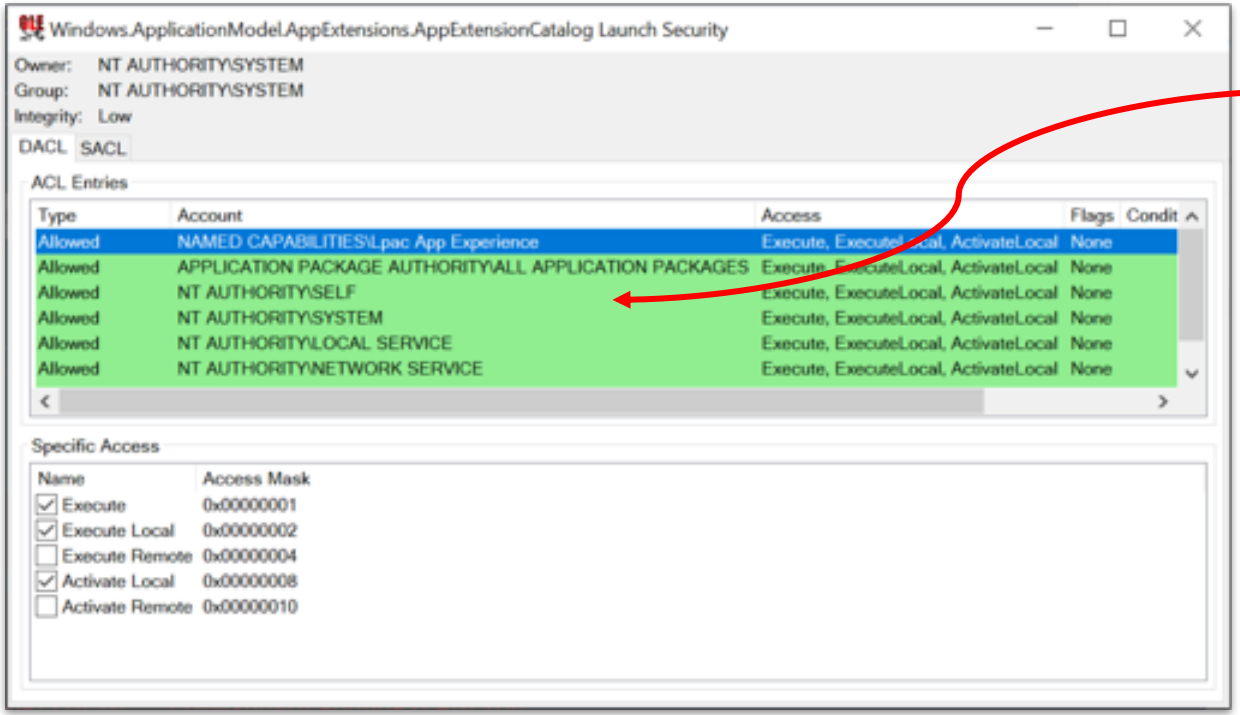
```
PS> Show-ComSecurityDescriptor -RuntimeDefault
```



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

Class Specific Permissions

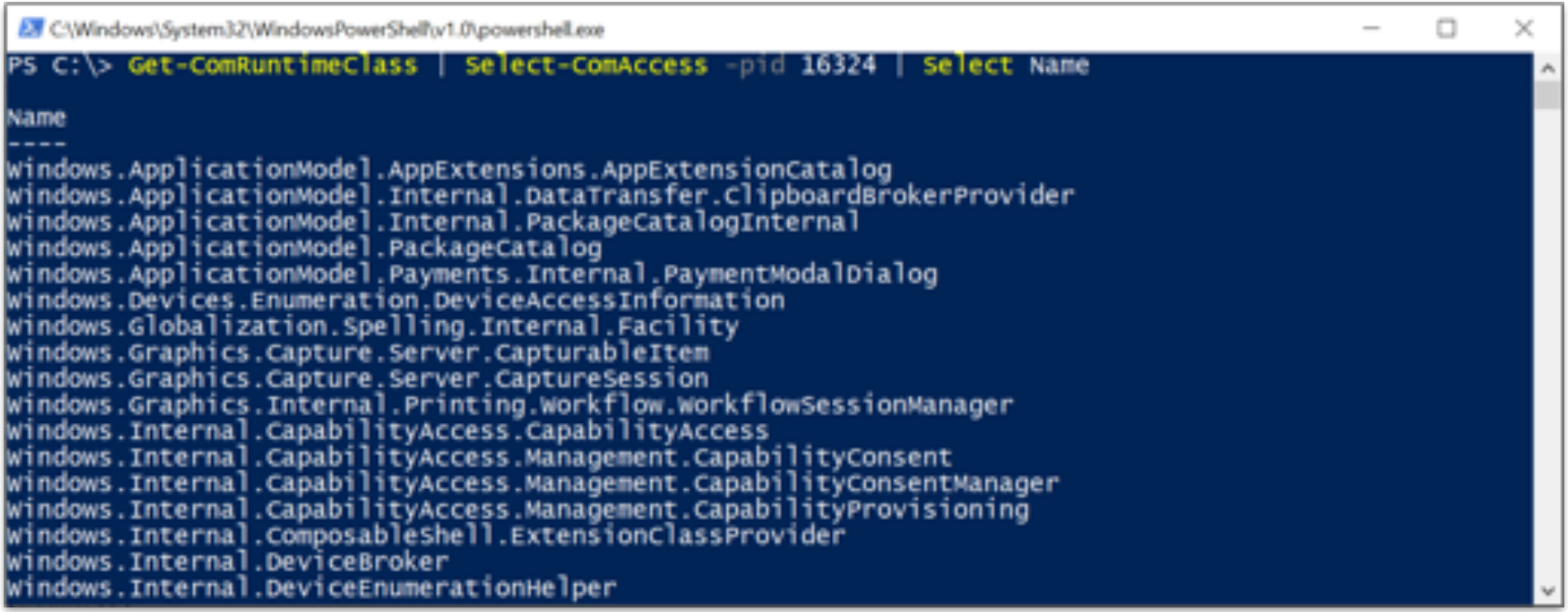
```
PS> Show-ComSecurityDescriptor $cls
```



Adds the ***lpacAppExperience*** capability

Finding Accessible Classes

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



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.

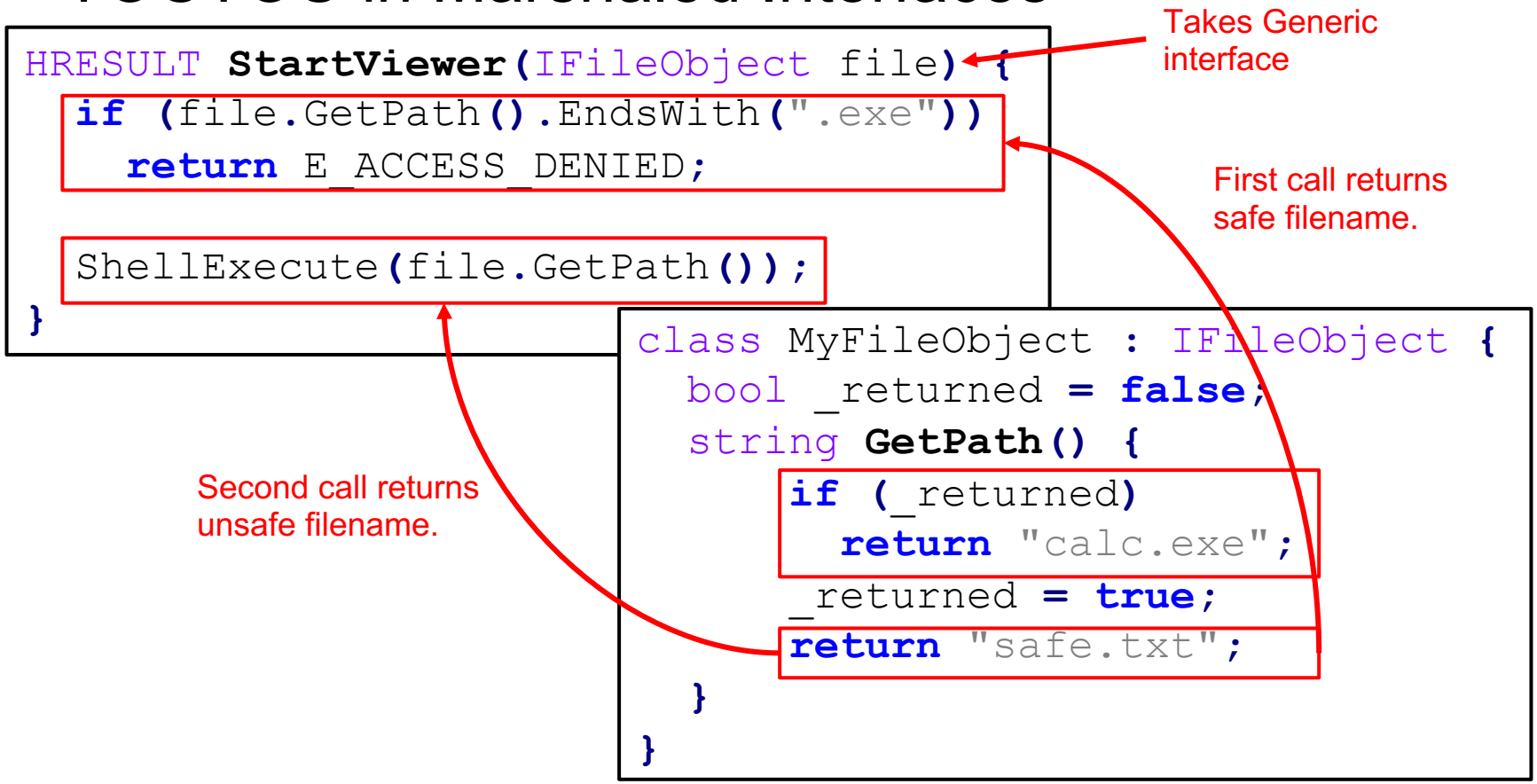
Checking for internetClient capability

HSTRING is a Counted String

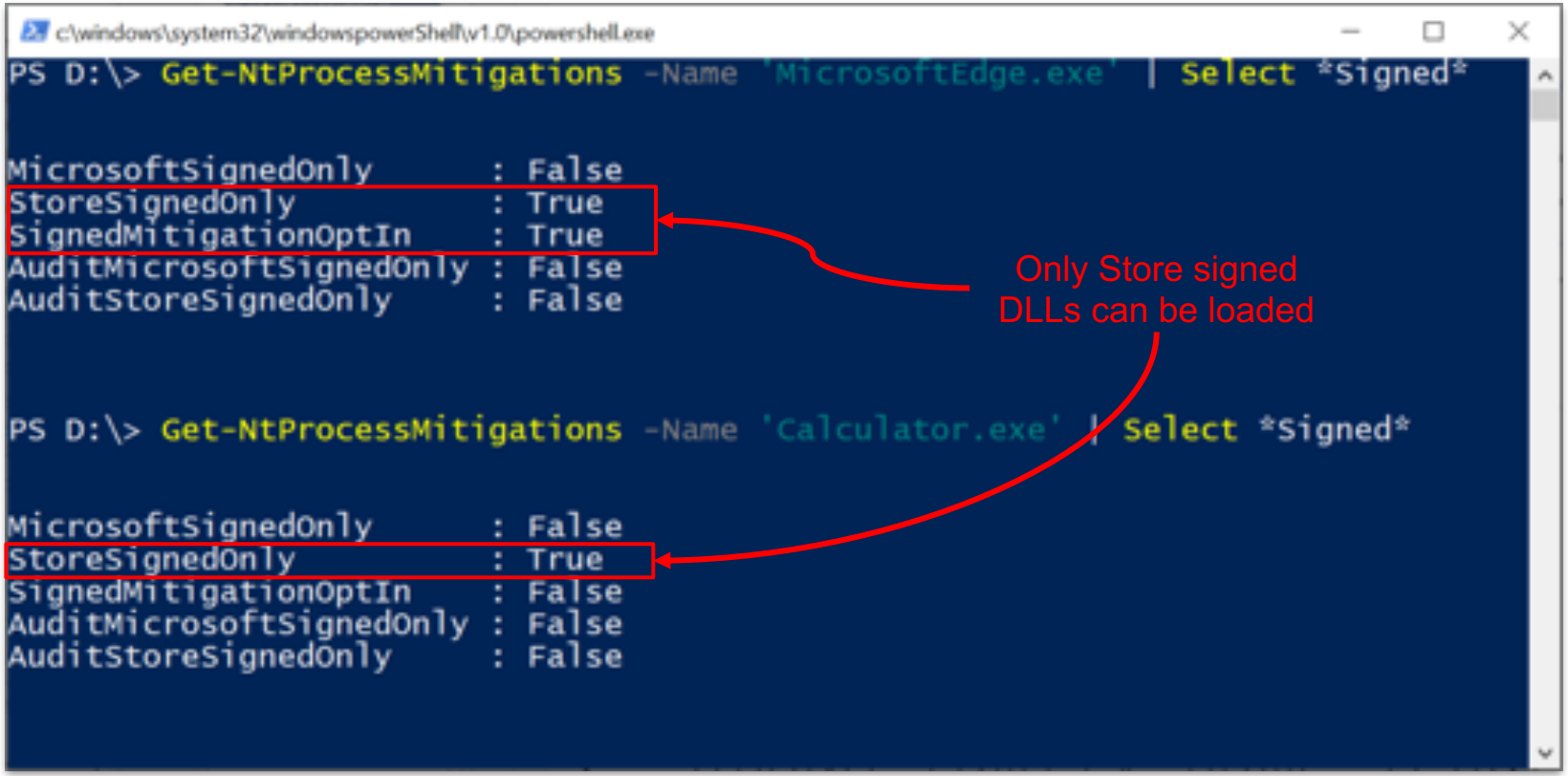
```
UINT32 length;
PCWSTR str = WindowsGetStringRawBuffer(hString,
                                       &length);
// Might not be equal.
assert(wcslen(str) == length);
```

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

TOCTOU in Marshaled Interfaces



Inject a DLL Into Running Process



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

谢谢