Microsoft

BLUEHAT

SECURITY ABOVE ALL ELSE

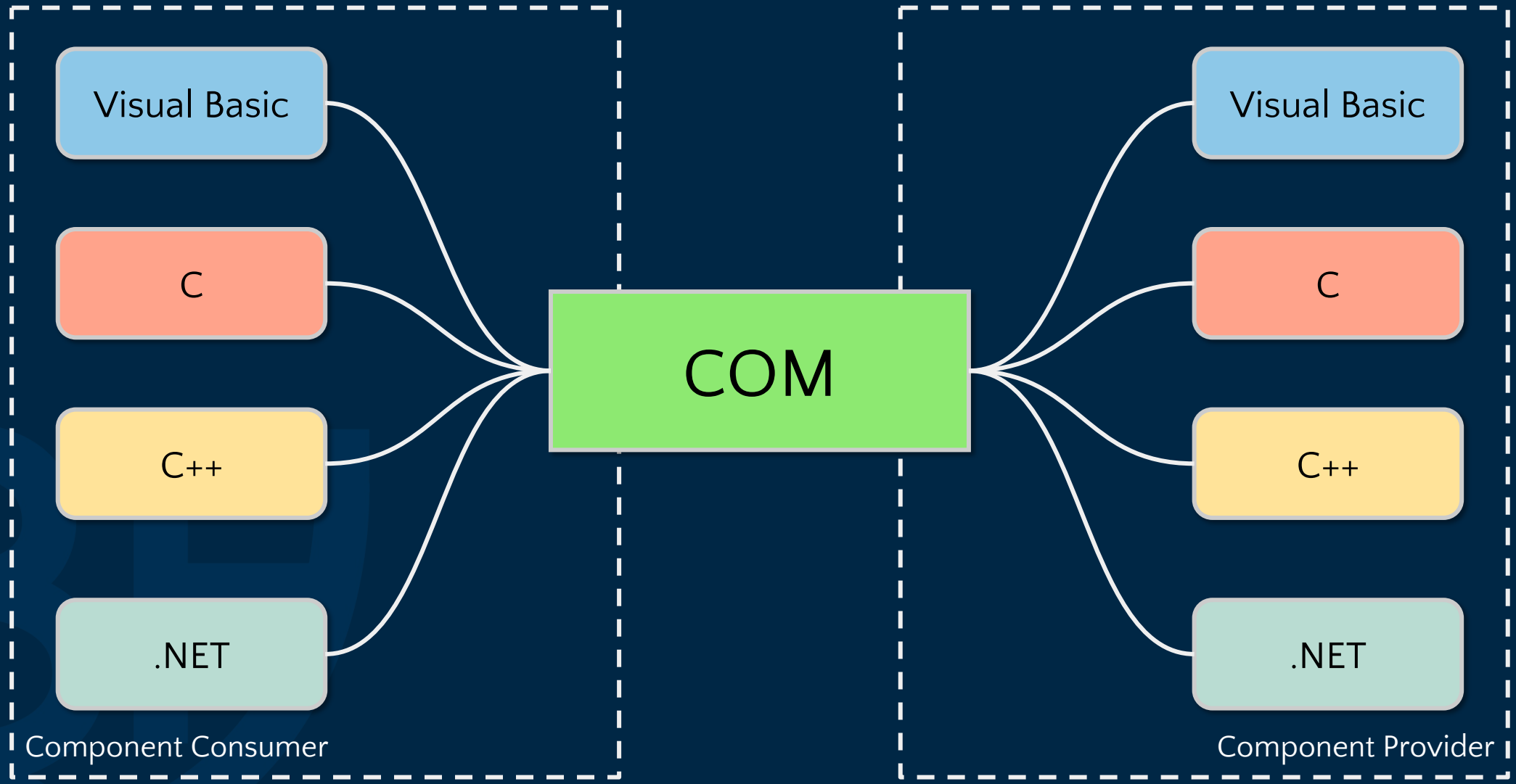
DCOM Research for Everyone!

James Forshaw - Google Project Zero

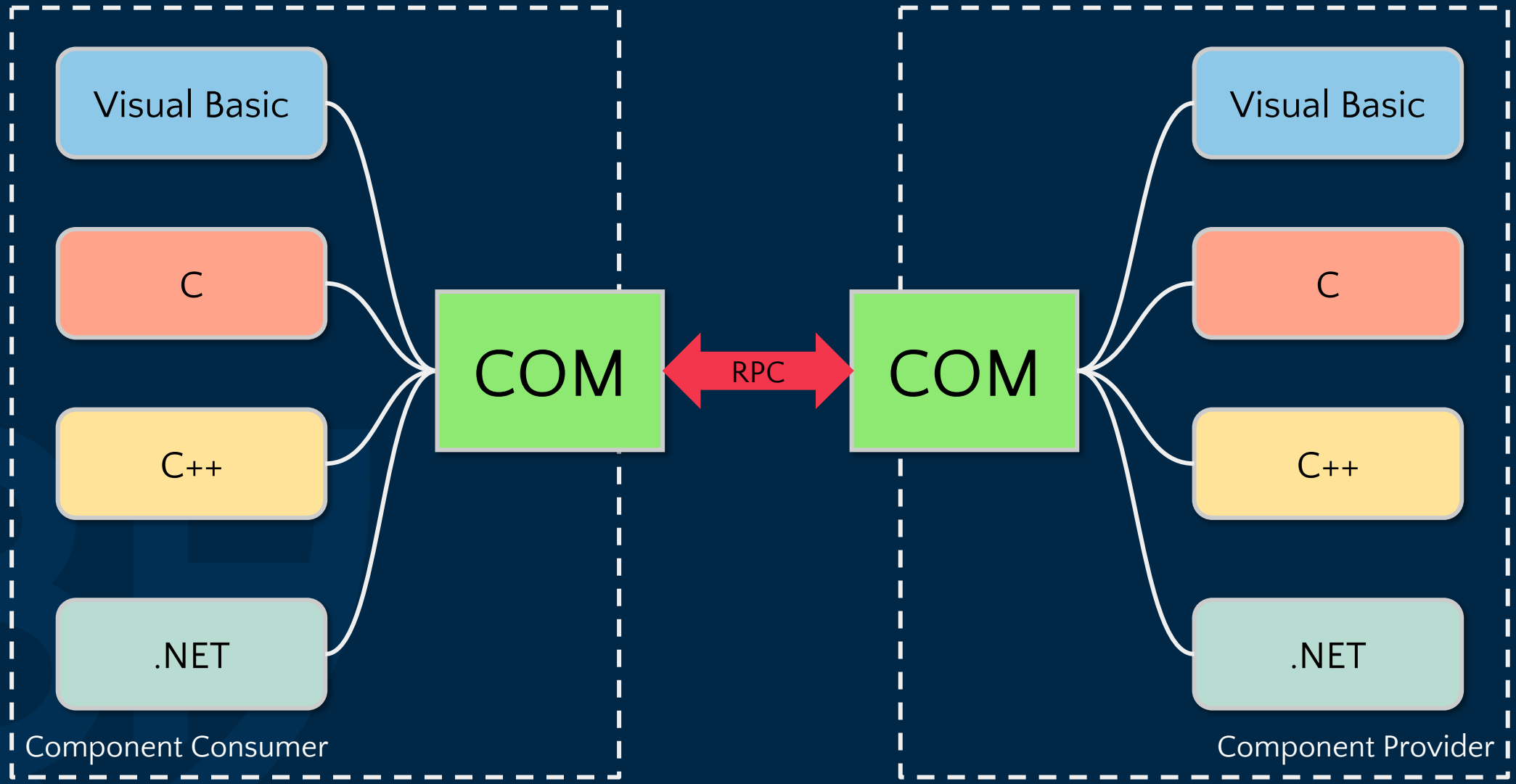
Agenda

1. Very brief introduction to COM
2. Why do we care anyway?
3. Using the Right Tool
4. DCOM Inter-process communication
5. Implementing DCOM clients
6. Conclusions

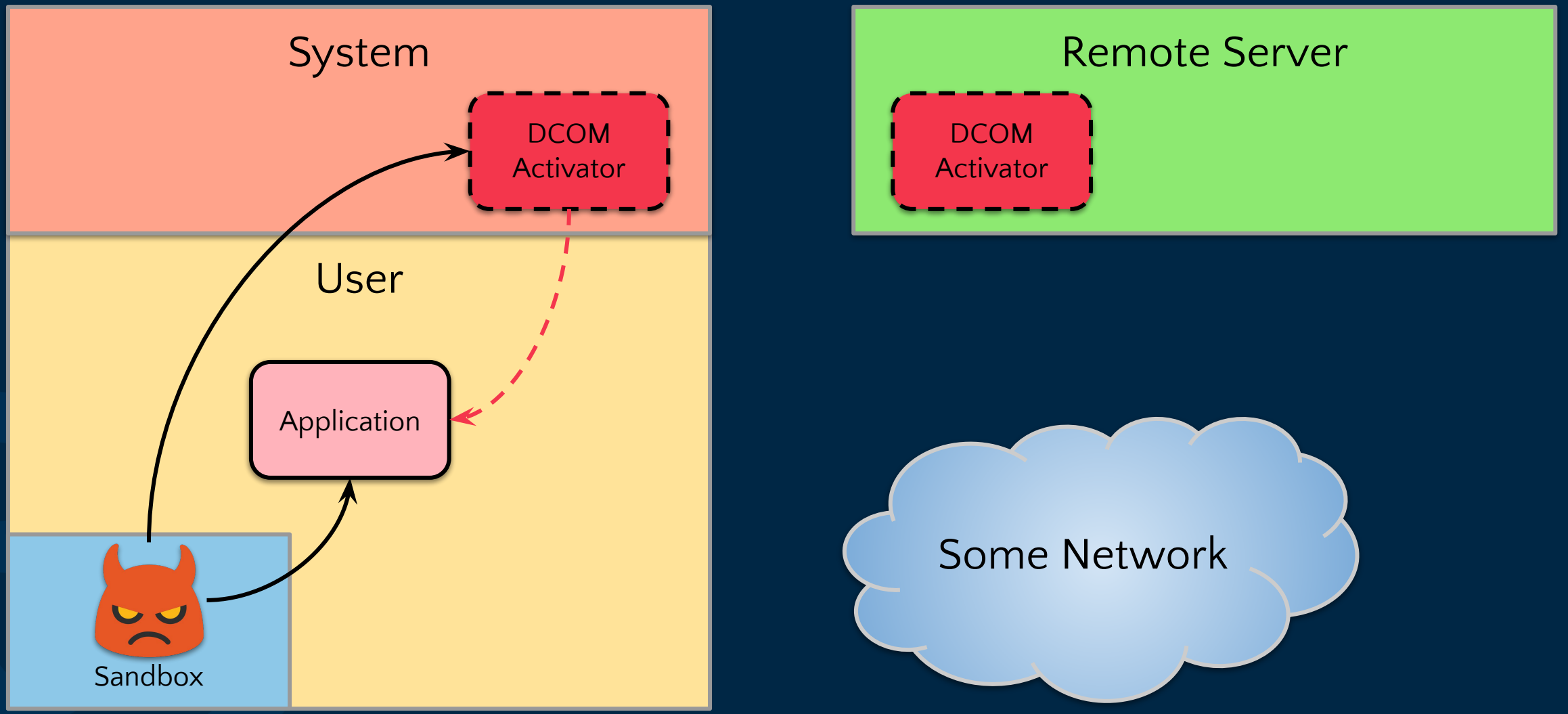
COM as Glue Logic



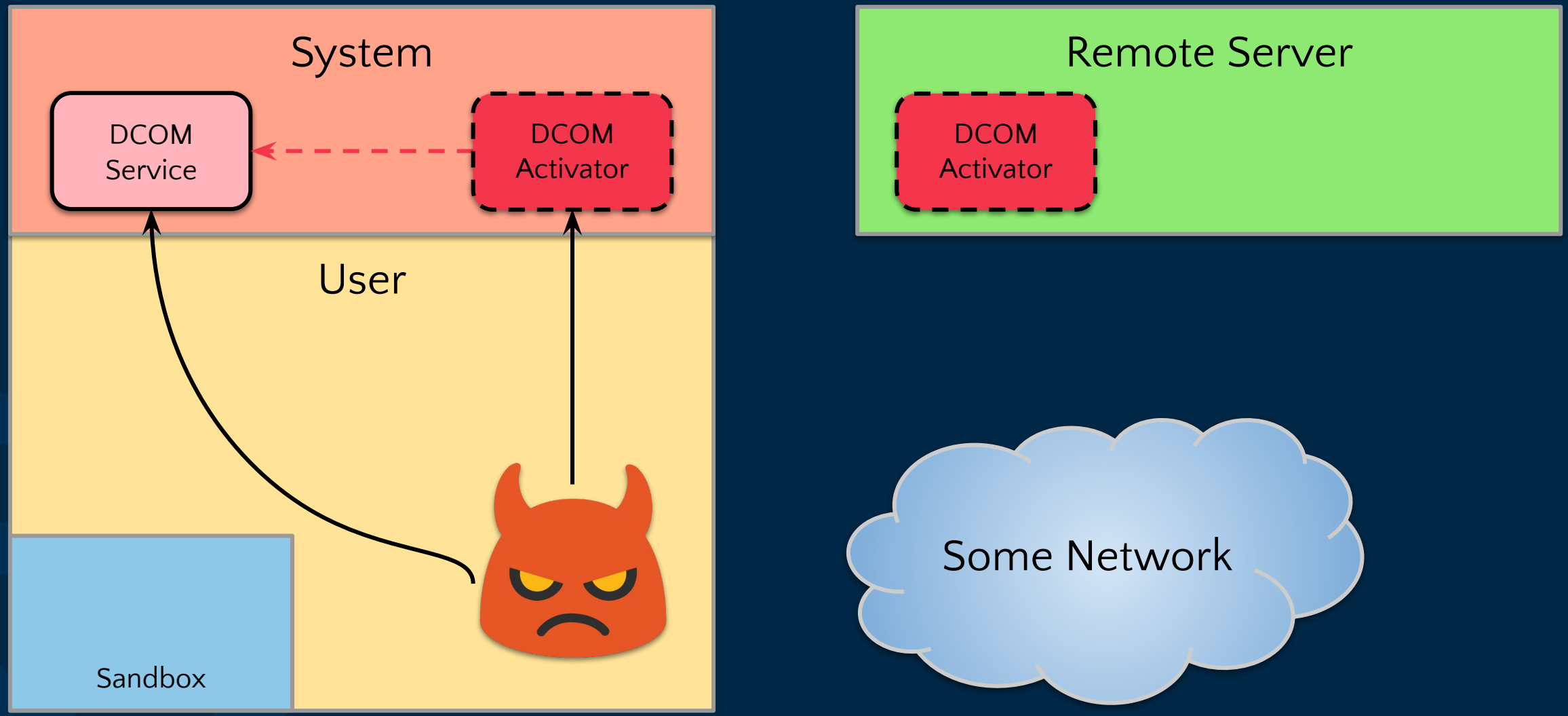
Distributed COM



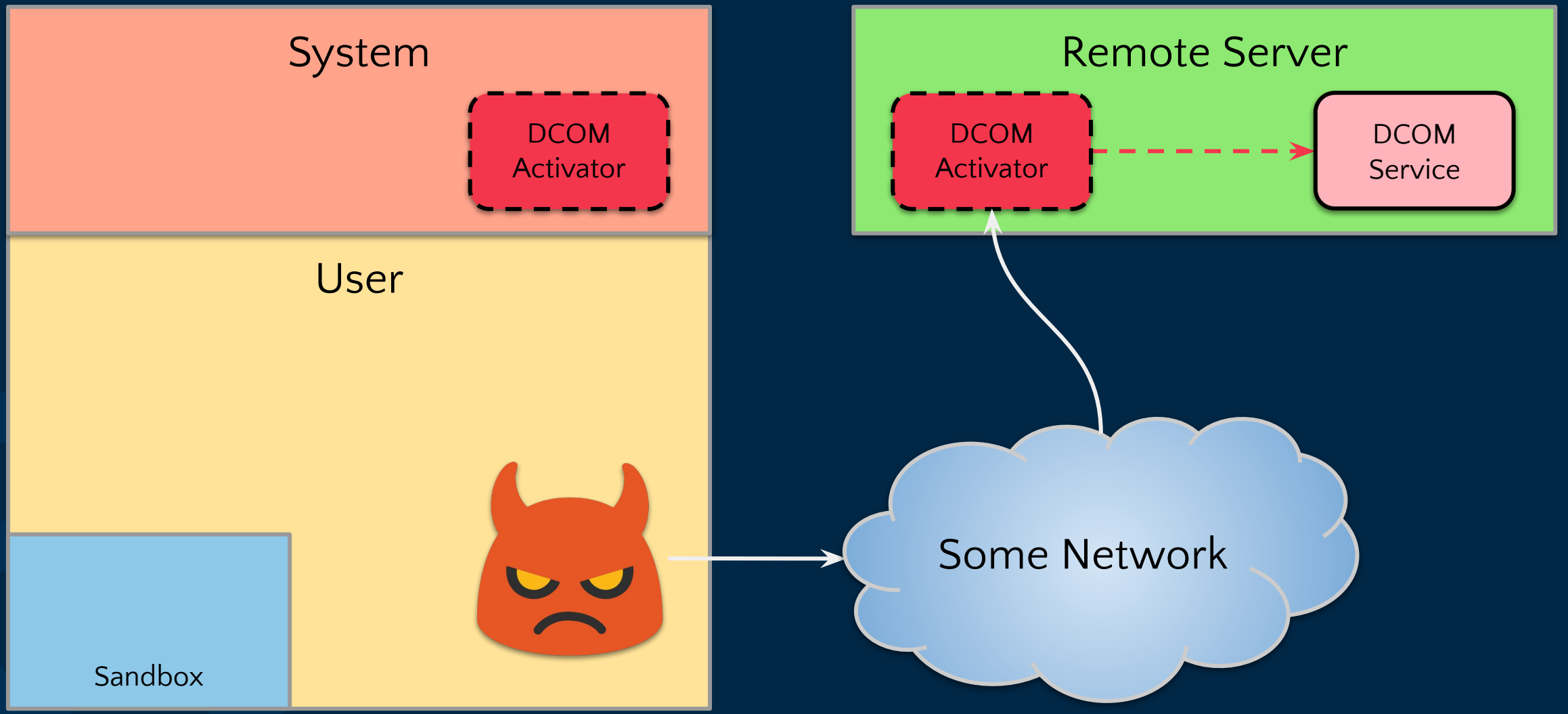
DCOM Attack Surface



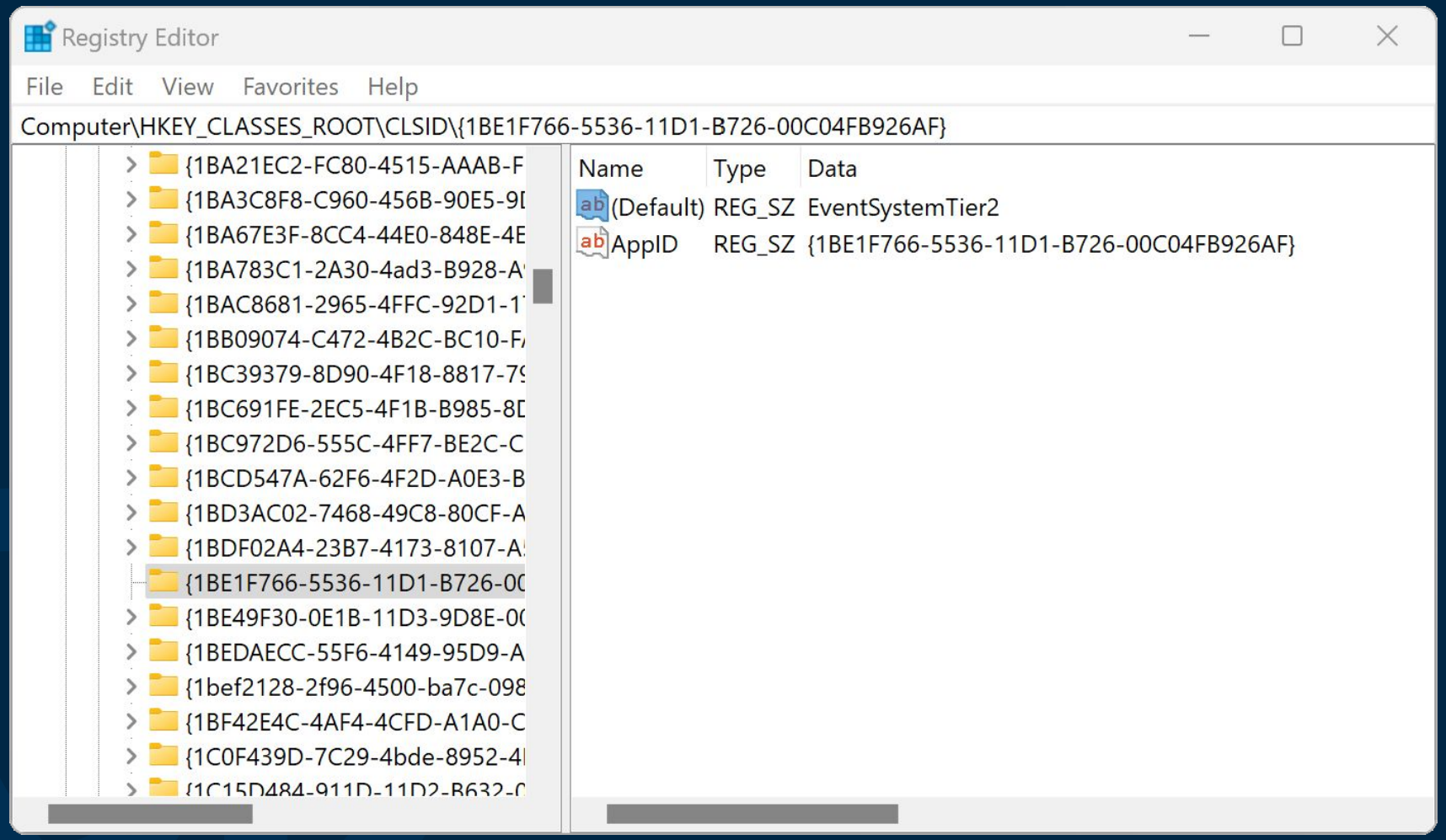
DCOM Attack Surface



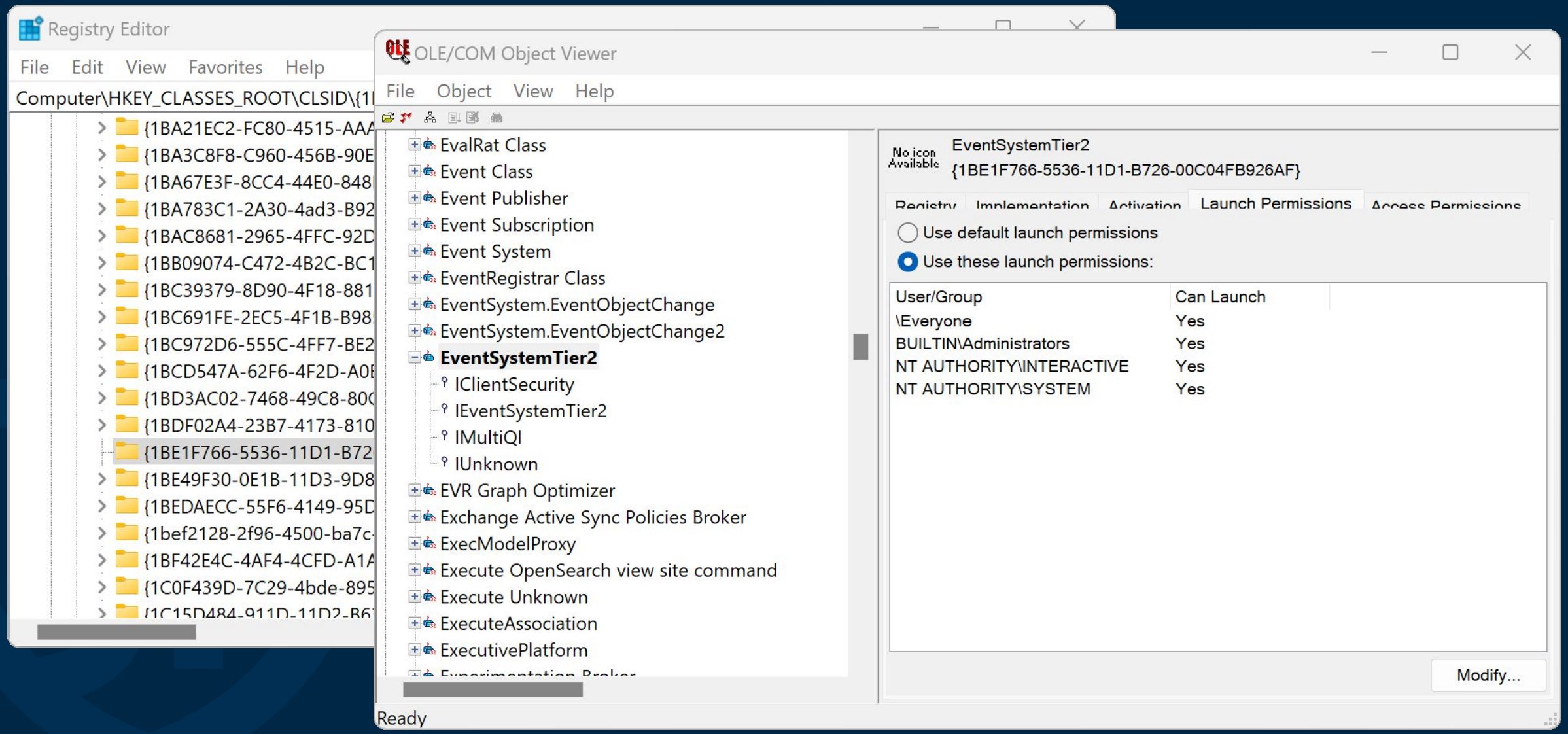
DCOM Attack Surface



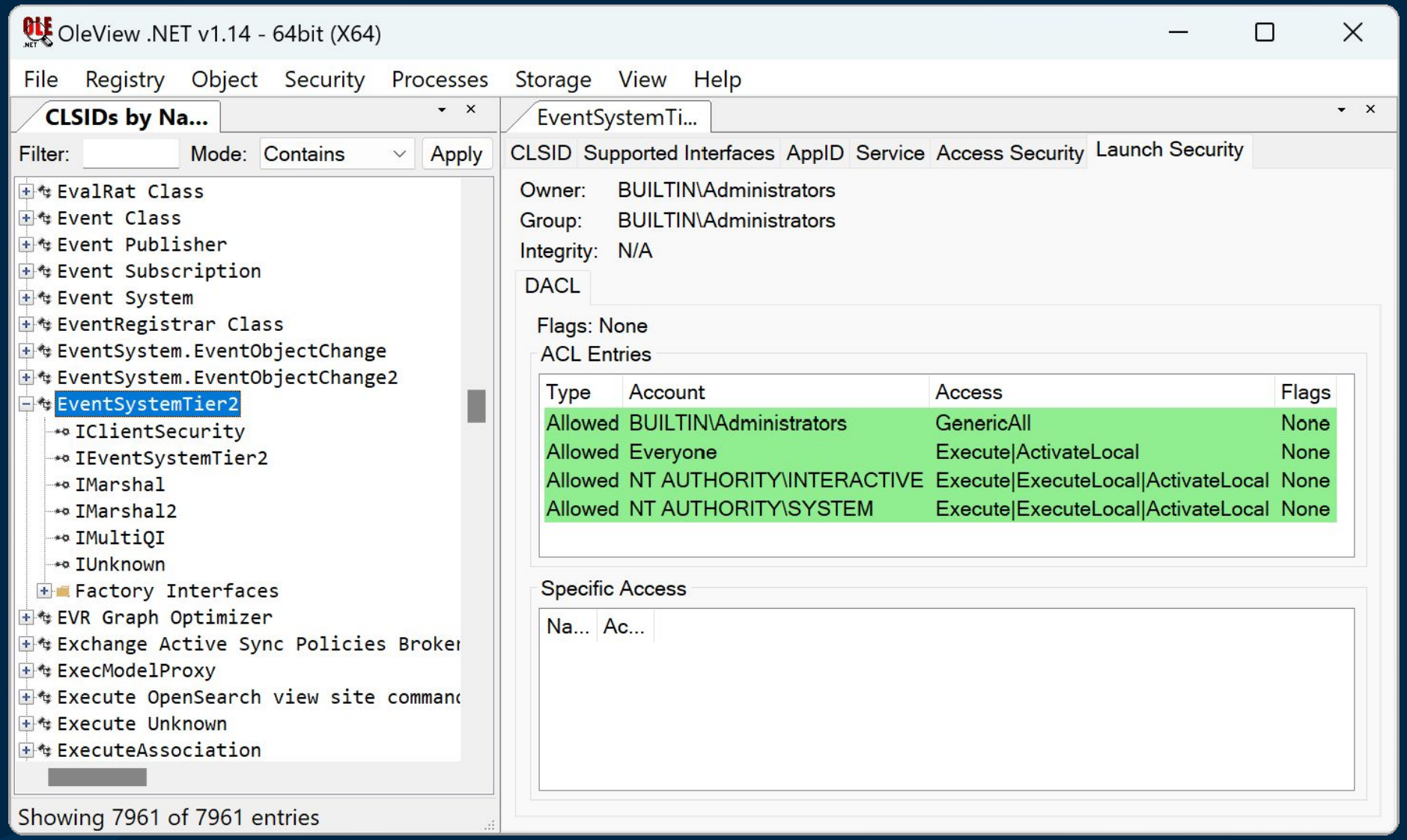
Old School COM Research



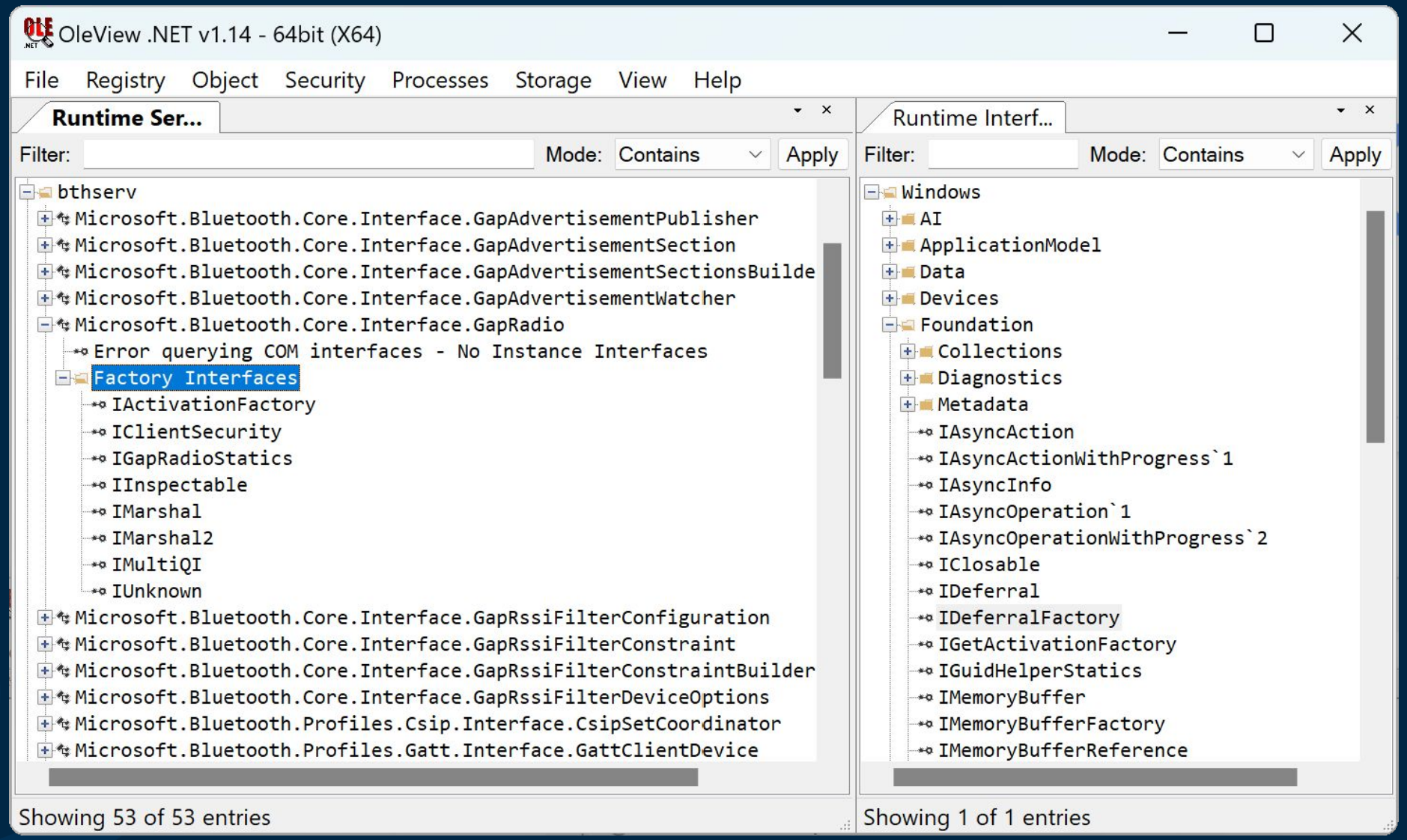
Old School COM Research



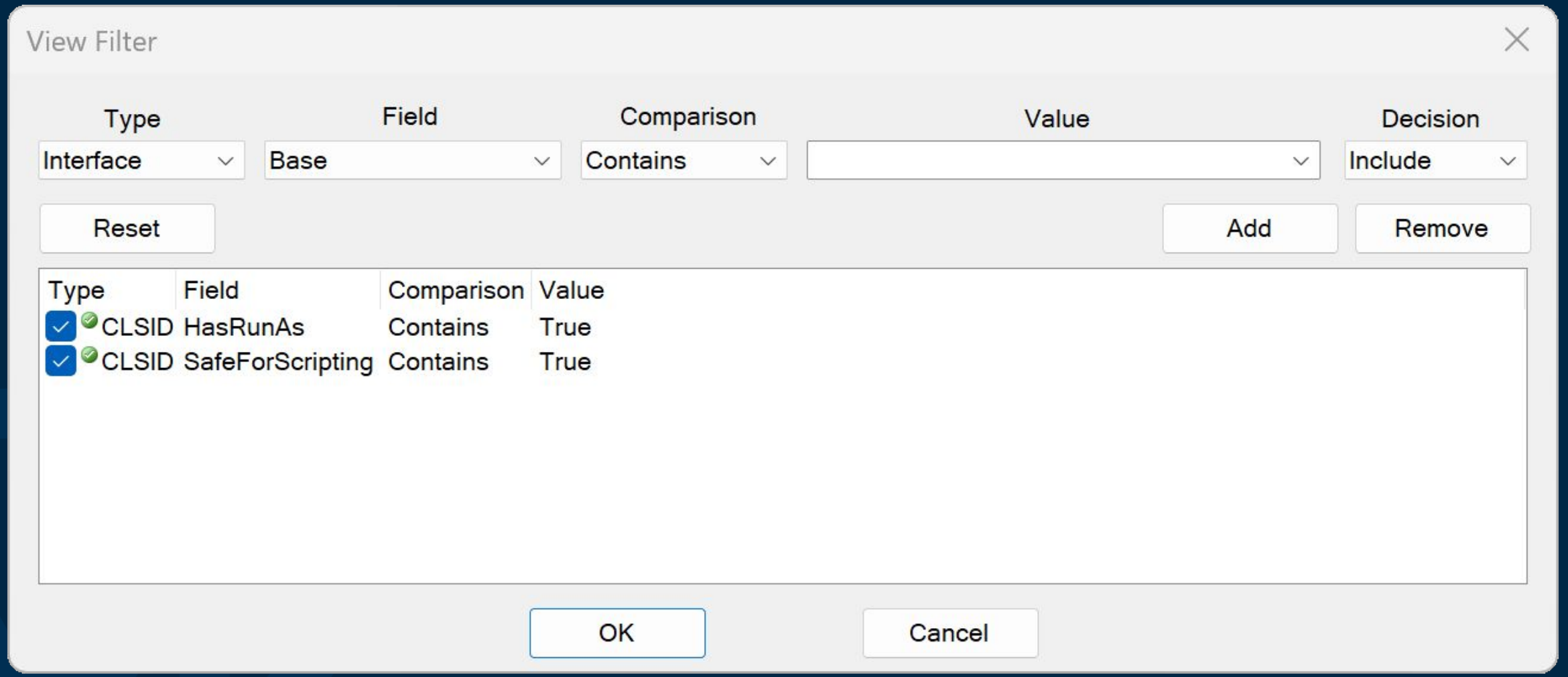
OleView.NET


OleView .NET v1.14 - 64bit (X64)
File Registry Object Security Processes Storage View Help
CLSIDs by Na...
Filter: Mode: Contains Apply
EvalRat Class
Event Class
Event Publisher
Event Subscription
Event System
EventRegistrar Class
EventSystem.EventObjectChange
EventSystem.EventObjectChange2
EventSystemTier2
IClientSecurity
IEventSystemTier2
IMarshal
IMarshal2
IMultiQI
IUnknown
Factory Interfaces
EVR Graph Optimizer
Exchange Active Sync Policies Broker
ExecModelProxy
Execute OpenSearch view site comman
Execute Unknown
ExecuteAssociation
Showing 7961 of 7961 entries
EventSystemTi...
CLSID Supported Interfaces AppID Service Access Security Launch Security
Owner: BUILTIN\Administrators
Group: BUILTIN\Administrators
Integrity: N/A
DACL
Flags: None
ACL Entries
Type Account Access Flags
Allowed BUILTIN\Administrators GenericAll None
Allowed Everyone Execute|ActivateLocal None
Allowed NT AUTHORITY\INTERACTIVE Execute|ExecuteLocal|ActivateLocal None
Allowed NT AUTHORITY\SYSTEM Execute|ExecuteLocal|ActivateLocal None
Specific Access
Na... Ac...

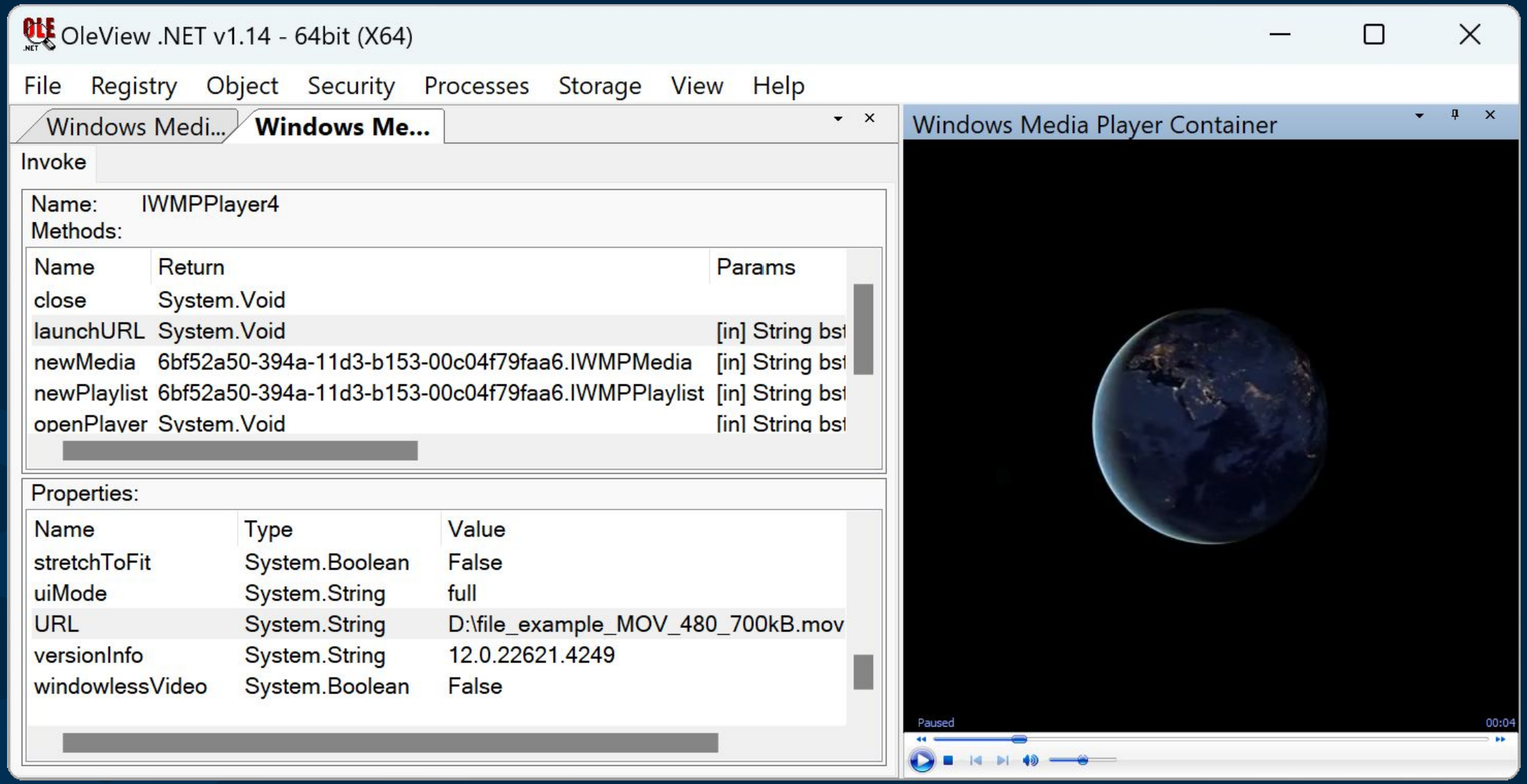
More Comprehensive Analysis



Filtering


View Filter
Type
Field
Comparison
Value
Decision
Interface
Base
Contains
Include
Reset
Add
Remove
Type
Field
Comparison
Value
CLSID
HasRunAs
Contains
True
CLSID
SafeForScripting
Contains
True
OK
Cancel

Test Container

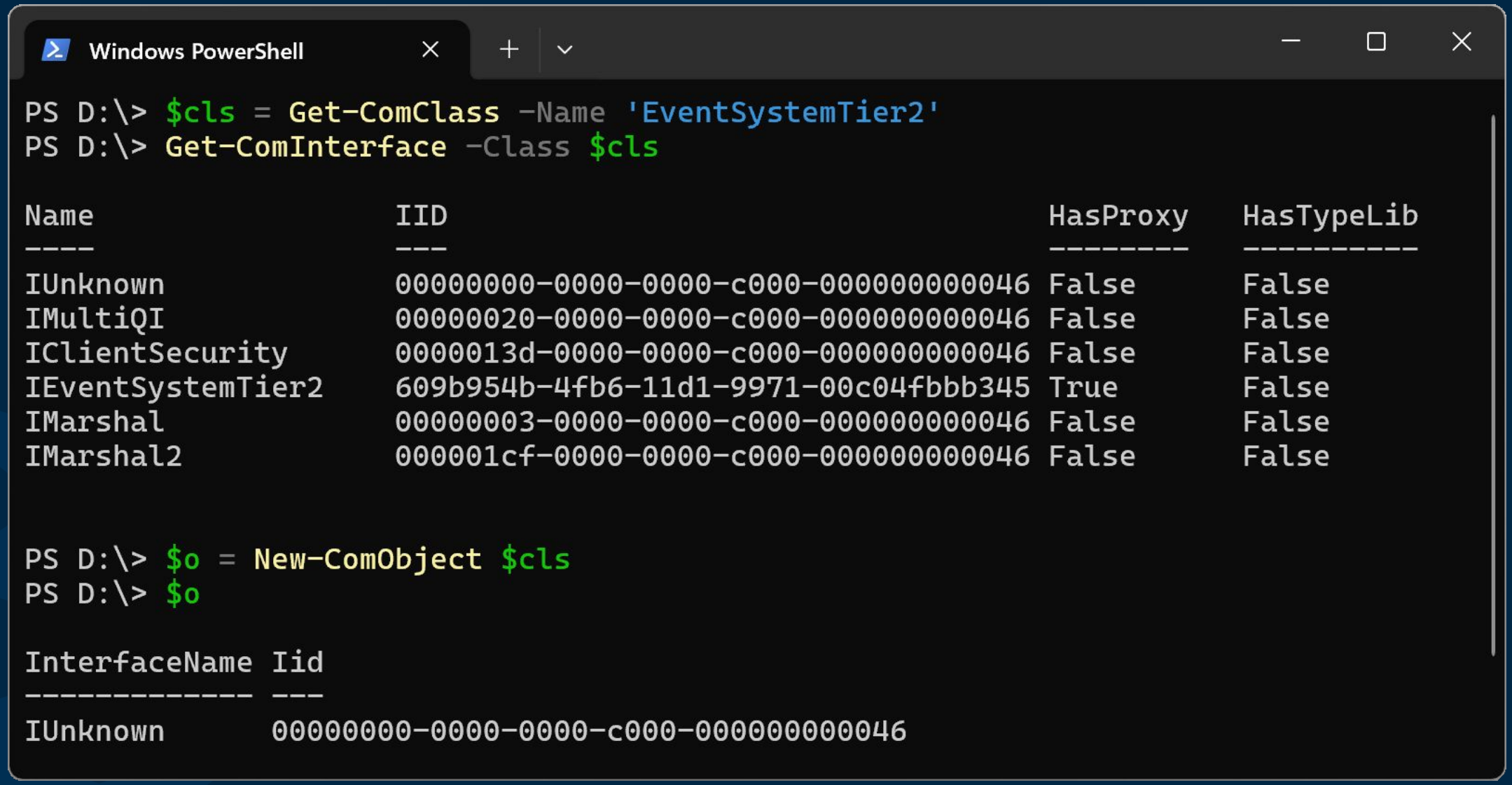


DEMO

DCOM Security Research

Enumerate Classes

PowerShell Scripting



Parse registry information into a database.

```
PS> Get-ComDatabase
```

Get all COM classes from the current database.

```
PS> Get-ComClass
```

Get all COM classes which have a registered OOP server.

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

Get all COM classes with the name "EventSystemTier2" in their registration

```
PS> $cls = Get-ComClass -Name "EventSystemTier2"
```

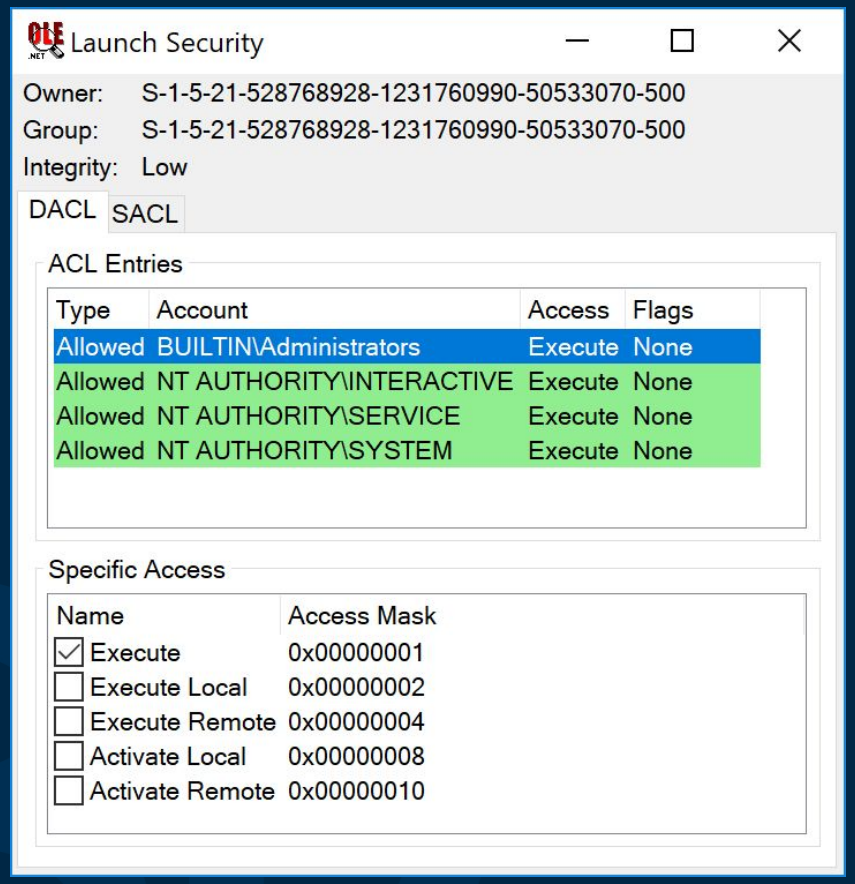
DEMO

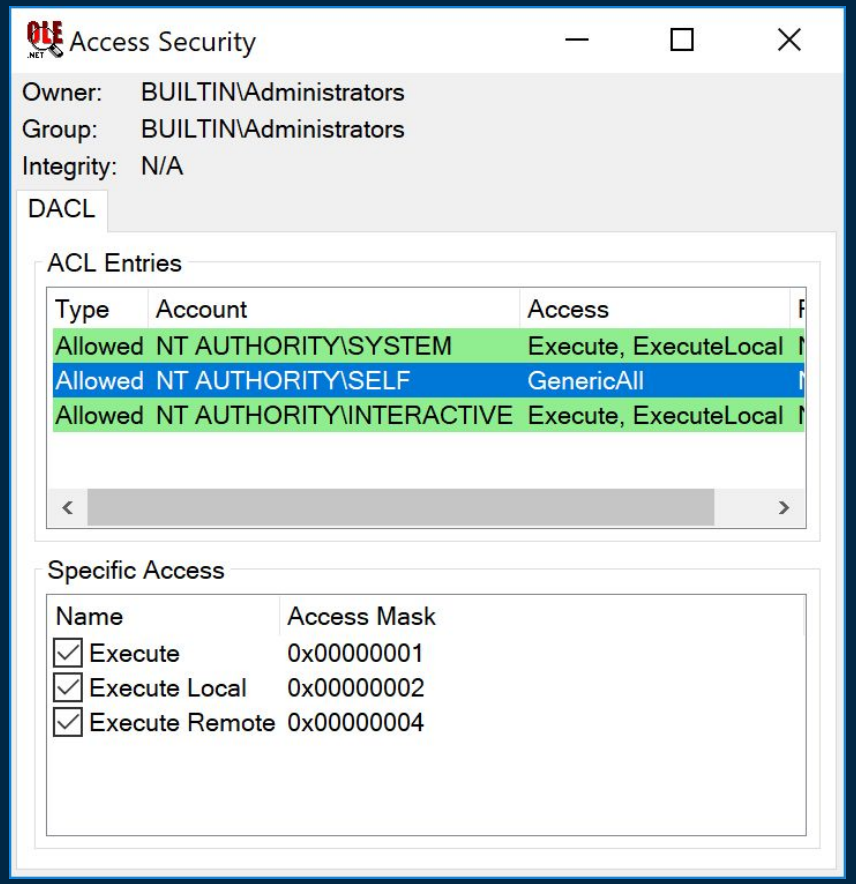
DCOM Security Research

Enumerate Classes

Verify Accessibility

DCOM Security





Launch = Create a new instance of the server.
Activate = Create new object on existing server.
Enforced in RPCSS

Access = Call methods on existing objects.
Enforced in Server Process
SELF = Process Token User SID

Print the COM launch security descriptor for a class to the console.

```
PS> Format-ComSecurityDescriptor $cls
```

Print the COM access security descriptor for a class to the console.

```
PS> Format-ComSecurityDescriptor $cls -ShowAccess
```

Get service hosted classes accessible by the current user.

```
PS> Get-ComClass -Service | Select-ComAccess
```

Get interactive user classes accessible by the token from PID 1234

```
PS> Get-ComClass -InteractiveUser |
    Select-ComAccess -ProcessId 1234
```

DEMO

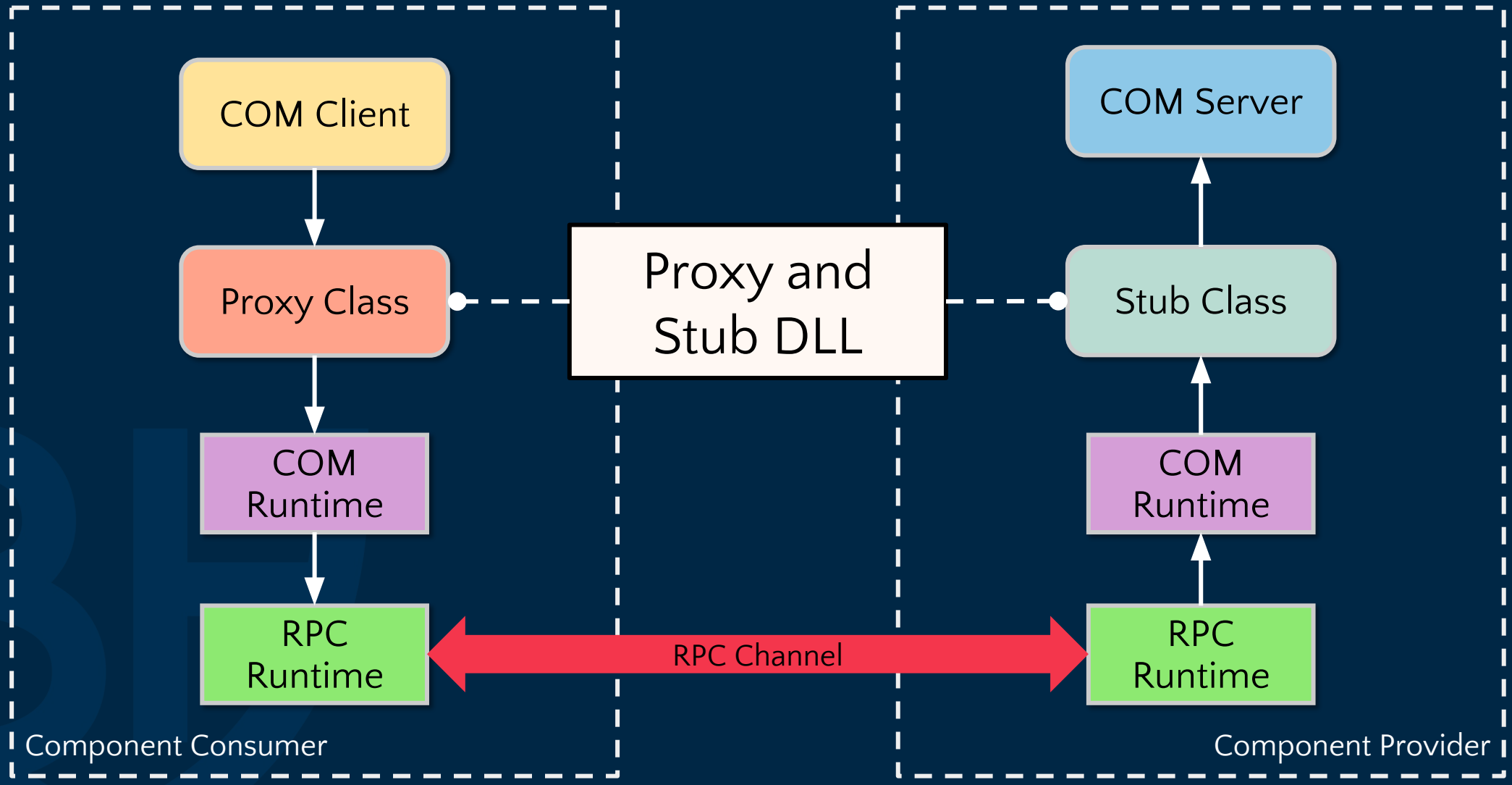
DCOM Security Research

Enumerate Classes

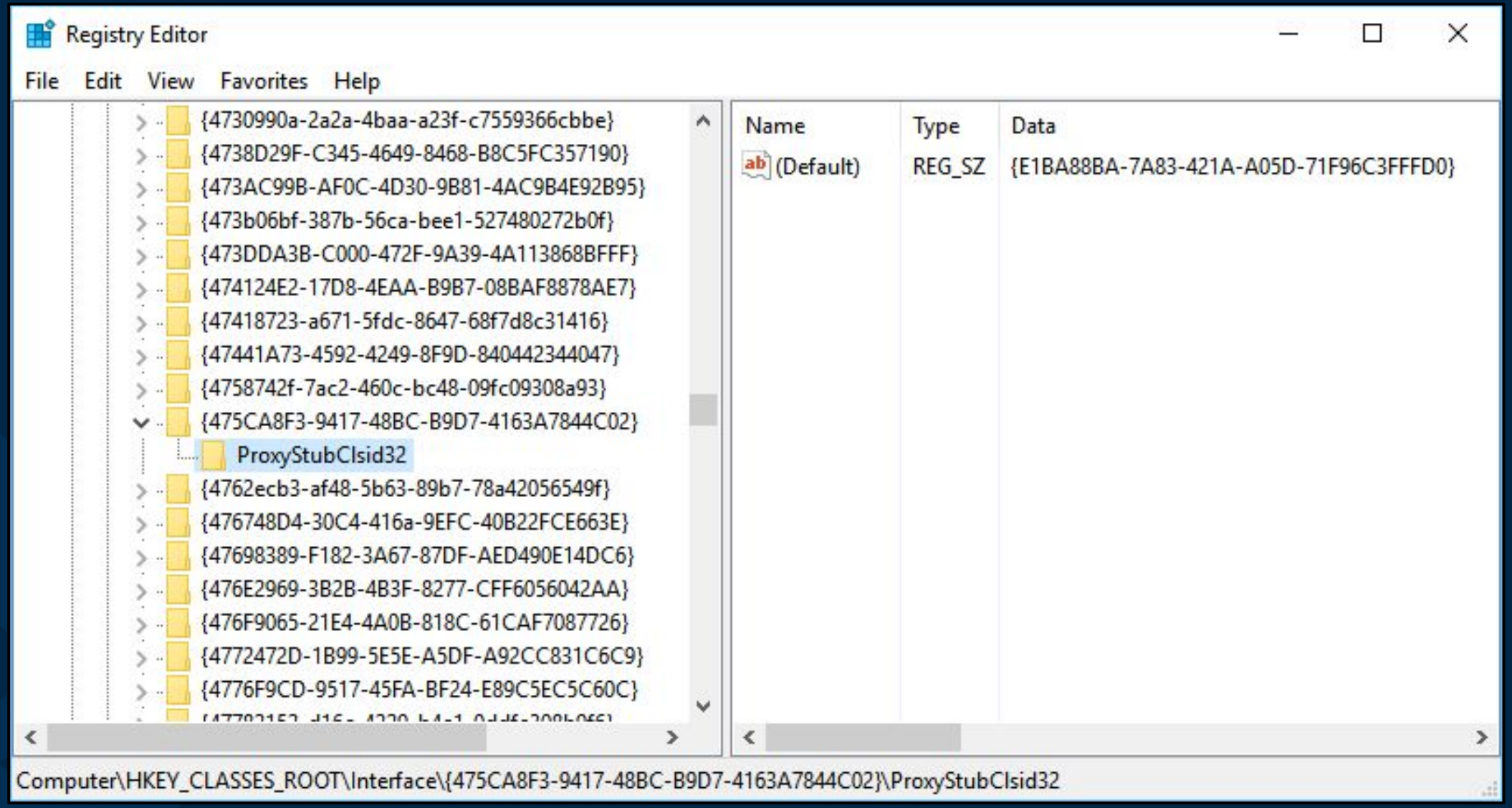
Verify Accessibility

Call Methods

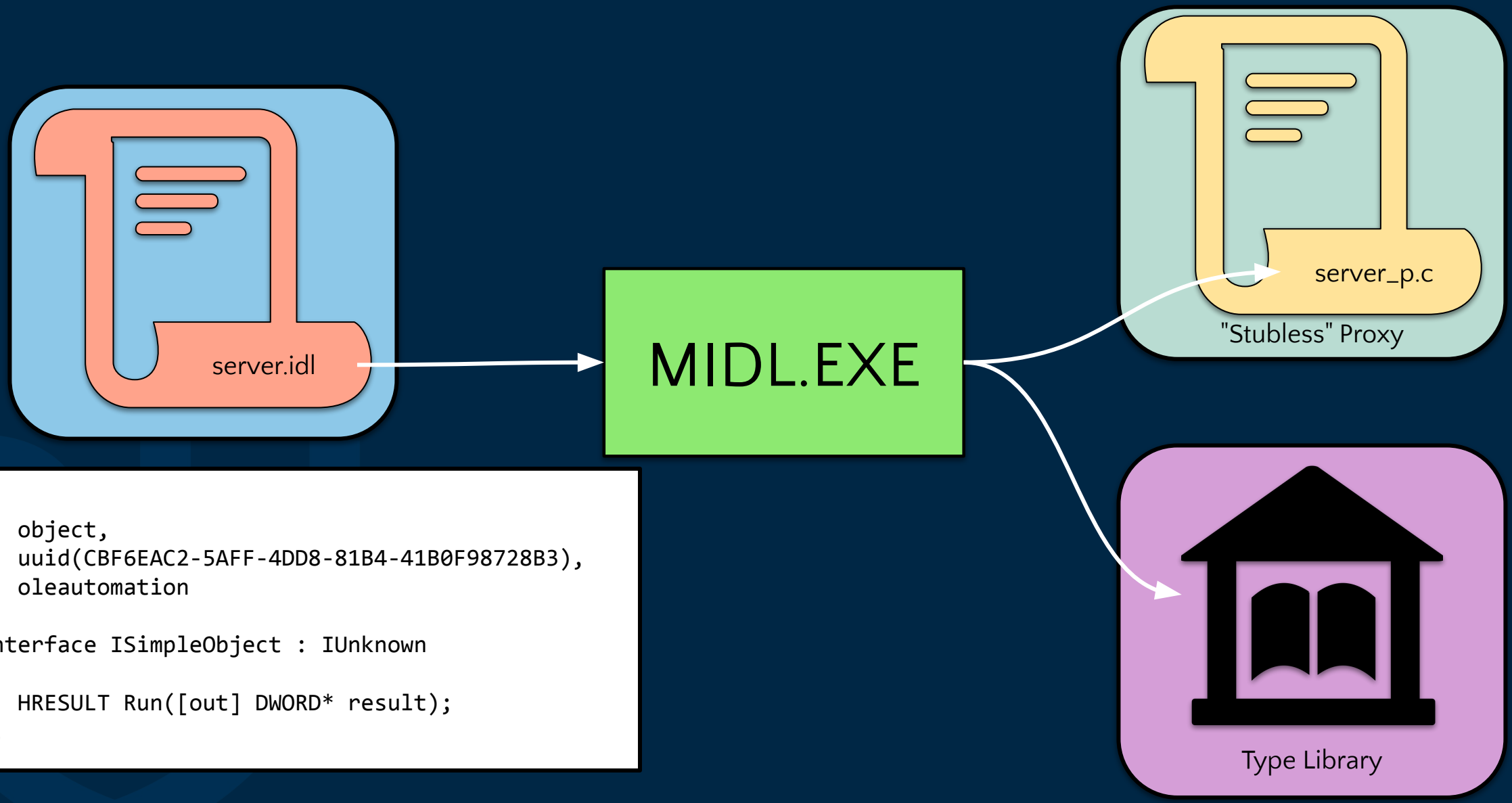
DCOM Proxies



Proxy Interface Registration

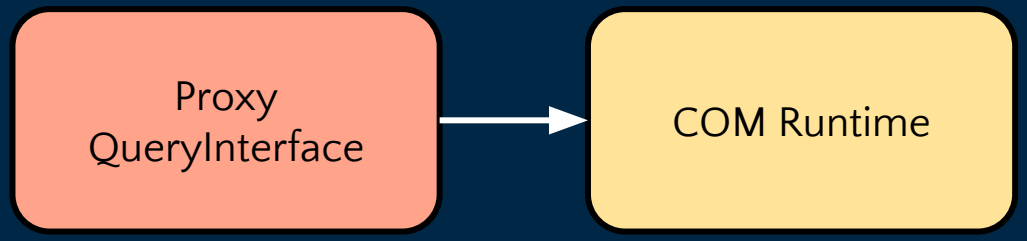


Building a Proxy Class

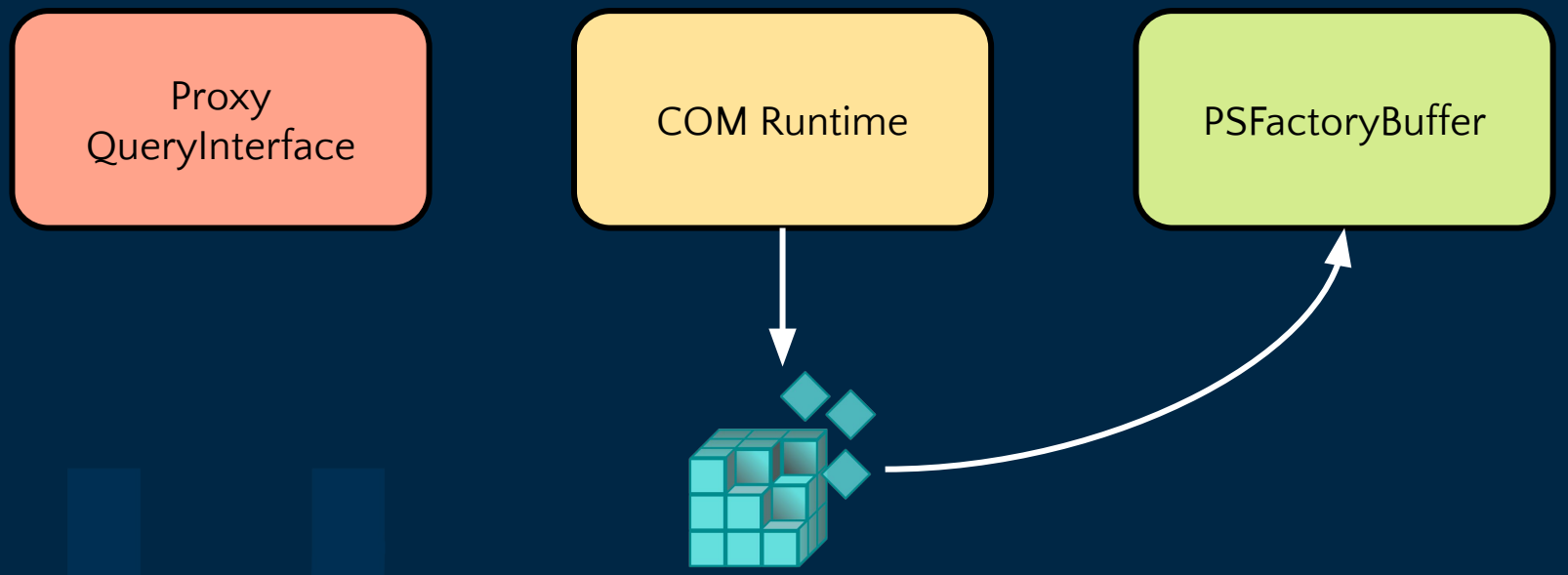



```
[
   object,
   uuid(CBF6EAC2-5AFF-4DD8-81B4-41B0F98728B3),
   oleautomation
]
interface ISimpleObject : IUnknown
{
   HRESULT Run([out] DWORD* result);
};
```

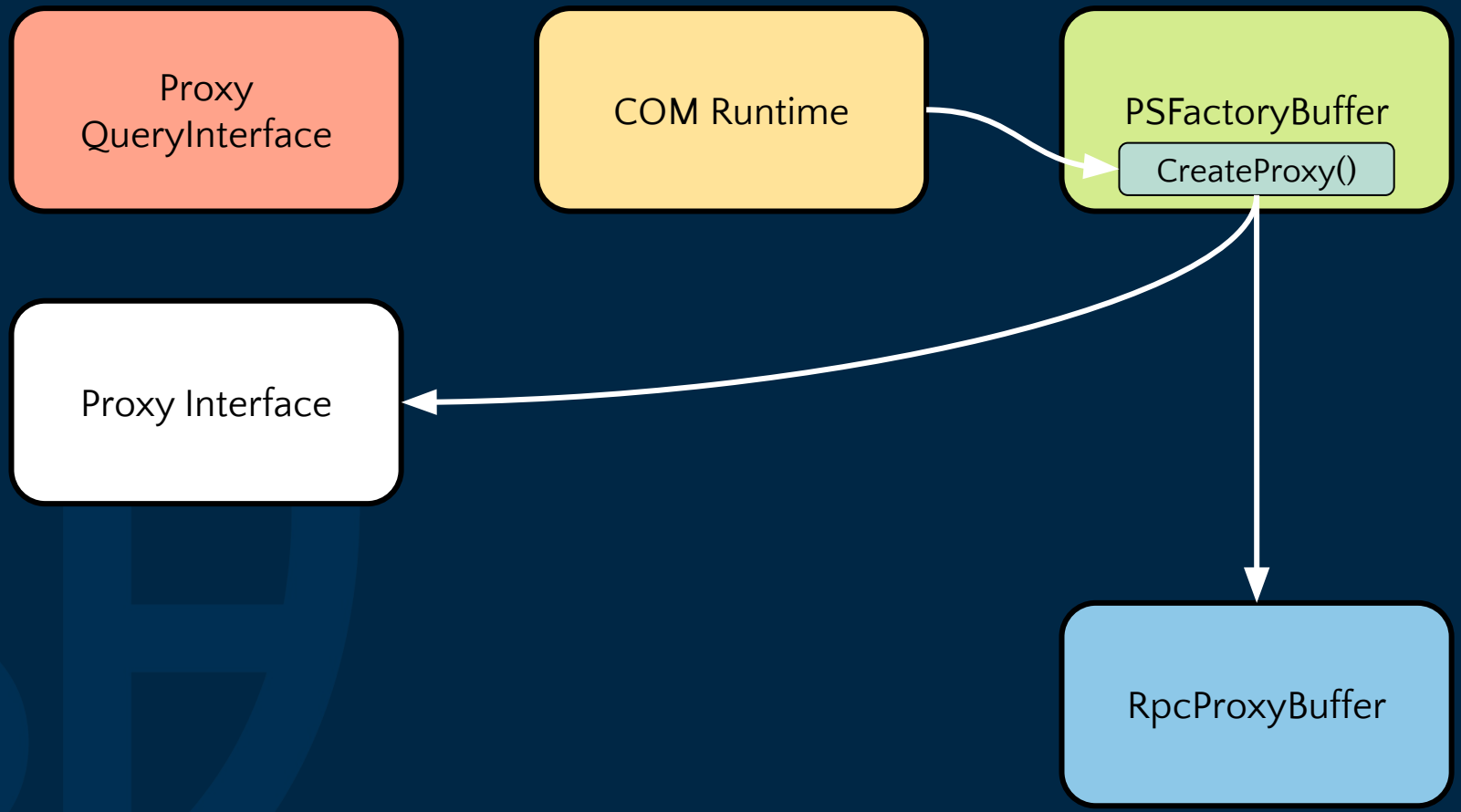
Creating a Proxy Instance



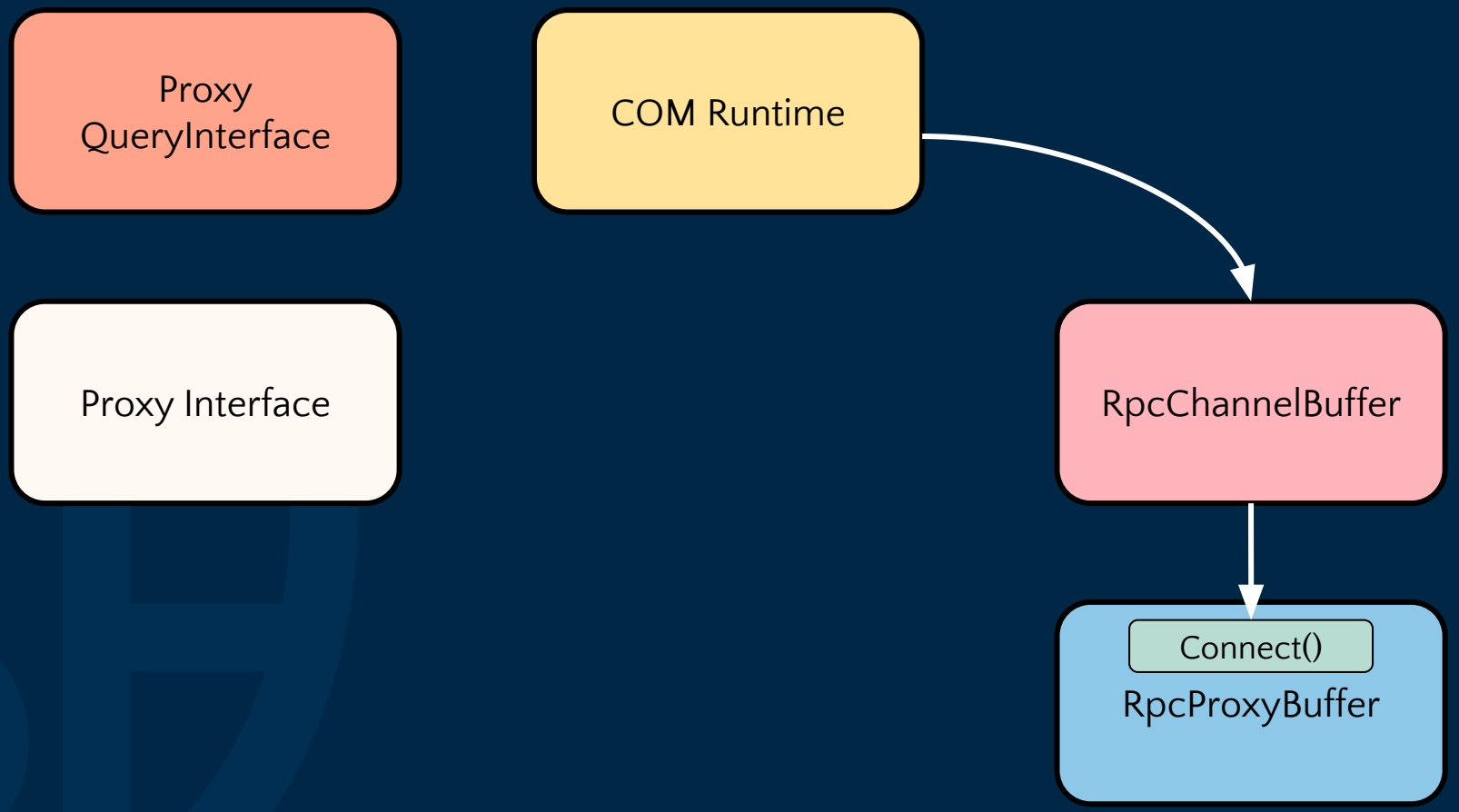
Creating a Proxy Instance



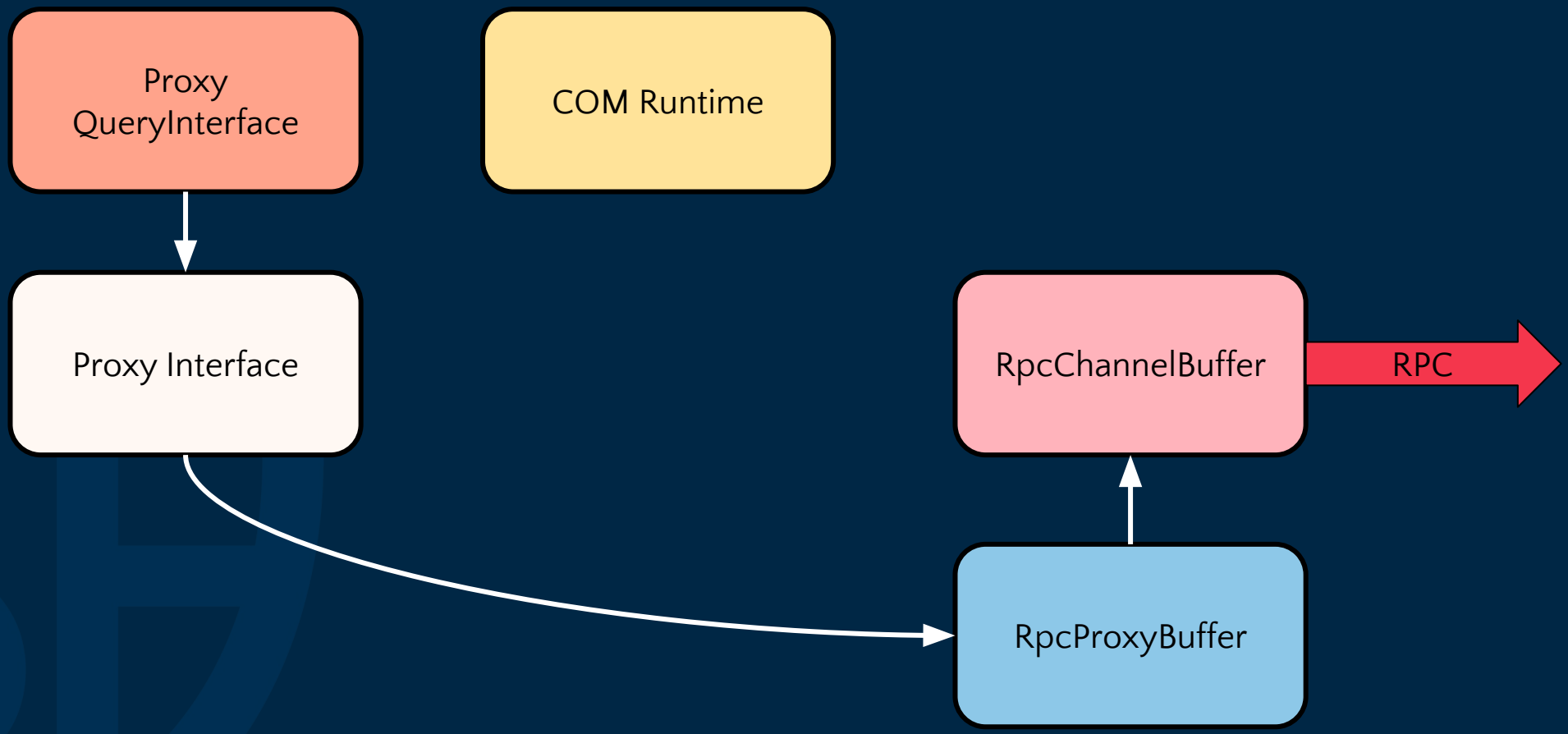
Creating a Proxy Instance



Creating a Proxy Instance



Creating a Proxy Instance



Enumerating Supported Interfaces

```
PS> Get-ComInterface -Class $cls
Name         IID                    HasProxy HasTypeLib
----         ---                    -------- ----------
IUnknown     00000000-0000-... False    False
IDispatch    00020400-0000-... True     False
...
```

Calling a Known Interface

```
PS> $src = @"using System;
using System.Runtime.InteropServices;
[ComImport, Guid("12345678-1234-1234-1234-121212121212"),
   InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
public interface IDemoInterface {
  int Run();
}"@
PS> Add-Type -TypeDefinition $src
PS> [IDemoInterface]$obj
Cannot convert the value of type "System.__ComObject" to
type "IDemoInterface".
```

A Second Try

```
PS> $src = @"
public static class DemoInterfaceCast {
  public static IDemoInterface Cast(object obj) {
    return (IDemoInterface)obj;
  }
}"@
PS> Add-Type -TypeDefinition $src
PS> $i = [DemoInterfaceCast]::Cast($obj)
PS> $i.Run()
Method invocation failed because [System.__ComObject]
does not contain a method named 'Run'.
```

Custom Forwarding Wrapper

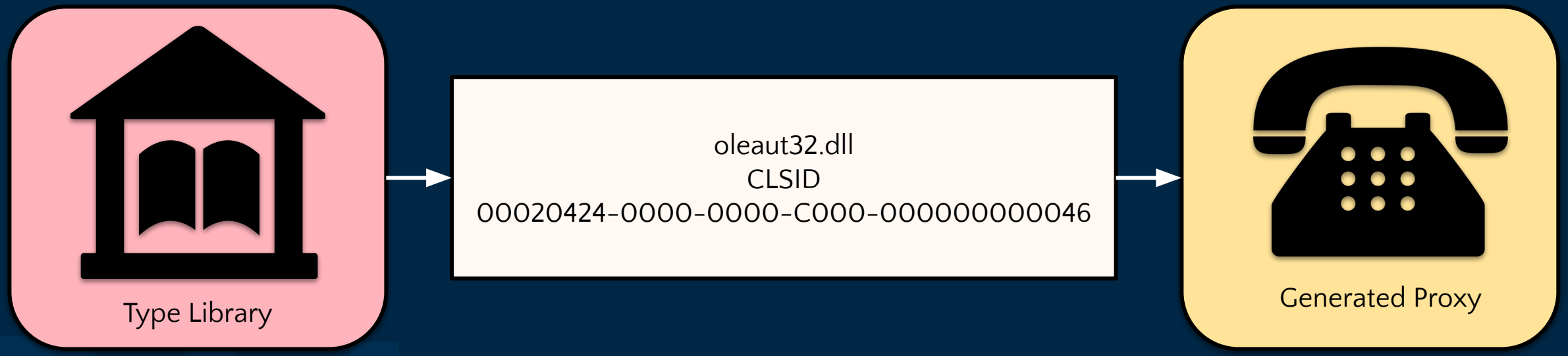
```
public class CustomWrapper {
  private readonly IDemoInterface _o;
  public CustomWrapper(IDemoInterface o) {
    _o = o;
  }
  public int Run() {
    return _o.Run();
  }
}
```

Custom Forwarding Wrapper

```
public class CustomWrapper {
  private readonly IDemoInterface _o;
  }
}
```

```
PS> $t = [IDemoInterface]
PS> $i = Get-ComObjectInterface $obj -Type $t
PS> $i.Run()
12345678
```

Type Library Proxy




```
PS> $i = Get-ComInterface $cls | ? IsAutomationProxy
```

```
PS> $i | ConvertTo-ComSourceCode -Parse
```

Creating a Callable Type

Marshal.GetTypeForITypeInfo(IntPtr) Method

Namespace: System.Runtime.InteropServices

Assembly: mscorlib.dll

Converts an unmanaged ITypeInfo object into a managed Type object.

C# Copy

```
[System.Security.SecurityCritical]
public static Type GetTypeForITypeInfo (IntPtr piTypeInfo);
```

Creating a Callable Type

Marshal.GetTypeForITypeInfo(IntPtr) Method

Namespace: System.Runtime.InteropServices

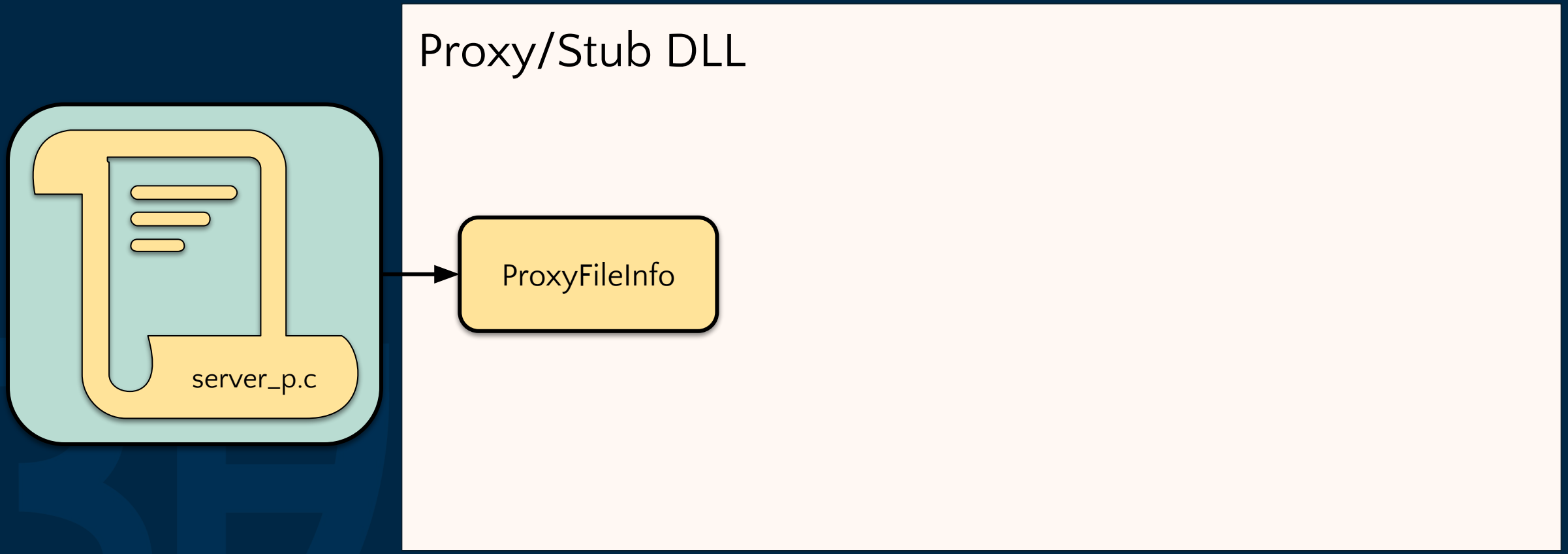
```
PS> $o = Get-ComObjectInterface $obj -Interface $i
```

C# Copy

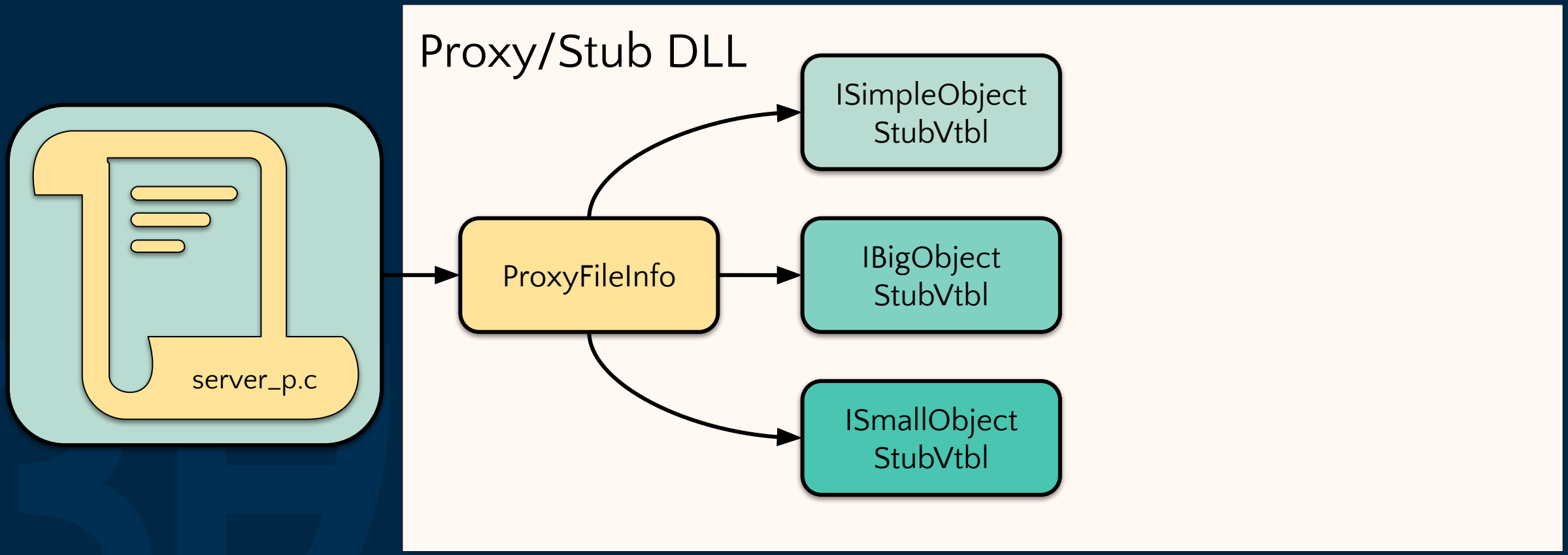
```
[System.Security.SecurityCritical]
public static Type GetTypeForITypeInfo (IntPtr piTypeInfo);
```

DEMO

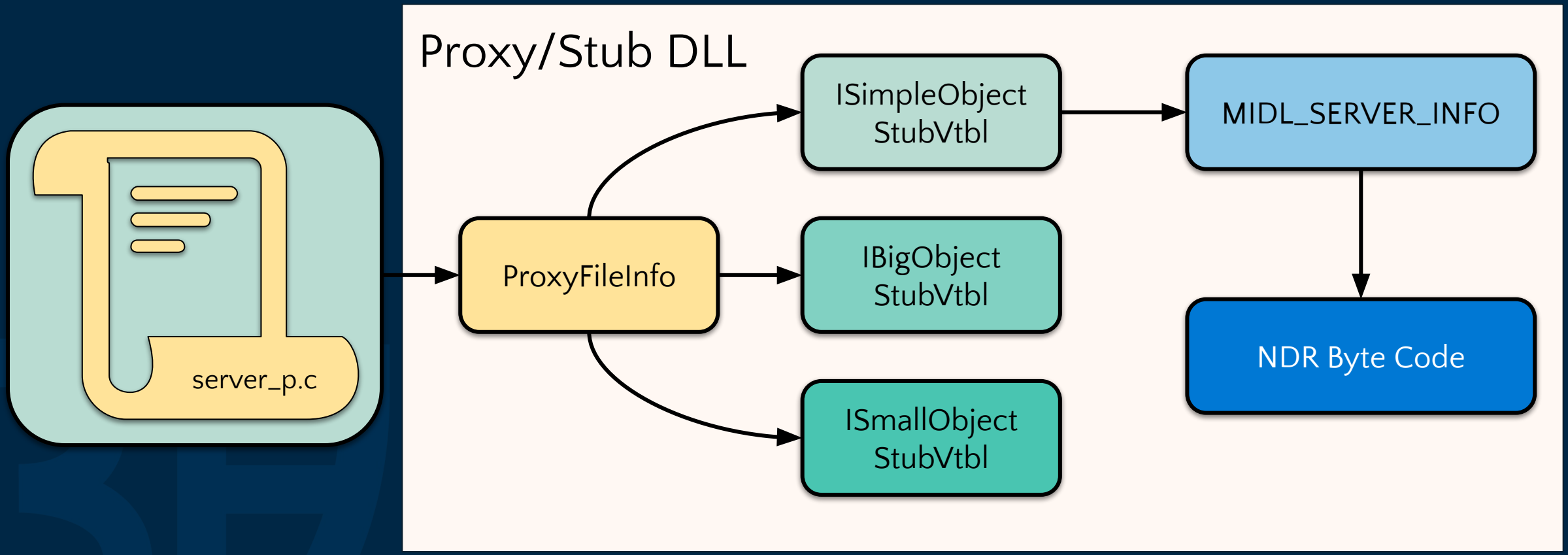
Stubless Proxy



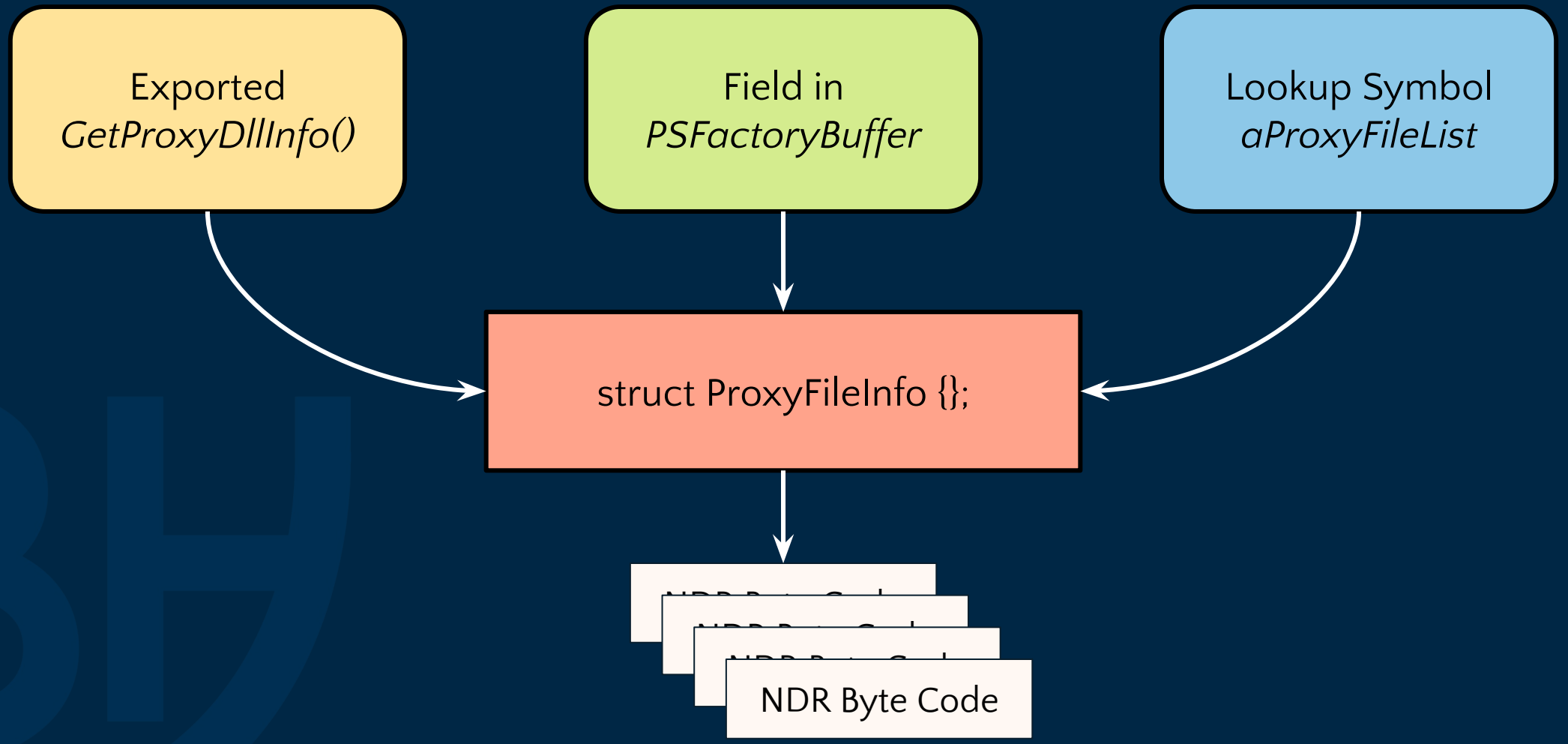
Stubless Proxy



Stubless Proxy



Extract Proxy Information



Converting NDR to a Callable Interface

Generate .NET COM Interface

Pros:

- Use built-in COM/.NET marshalling

Cons:

- Difficult to represent complex structures
- Passing invalid data could crash process

Generate .NET RPC Client

Pros:

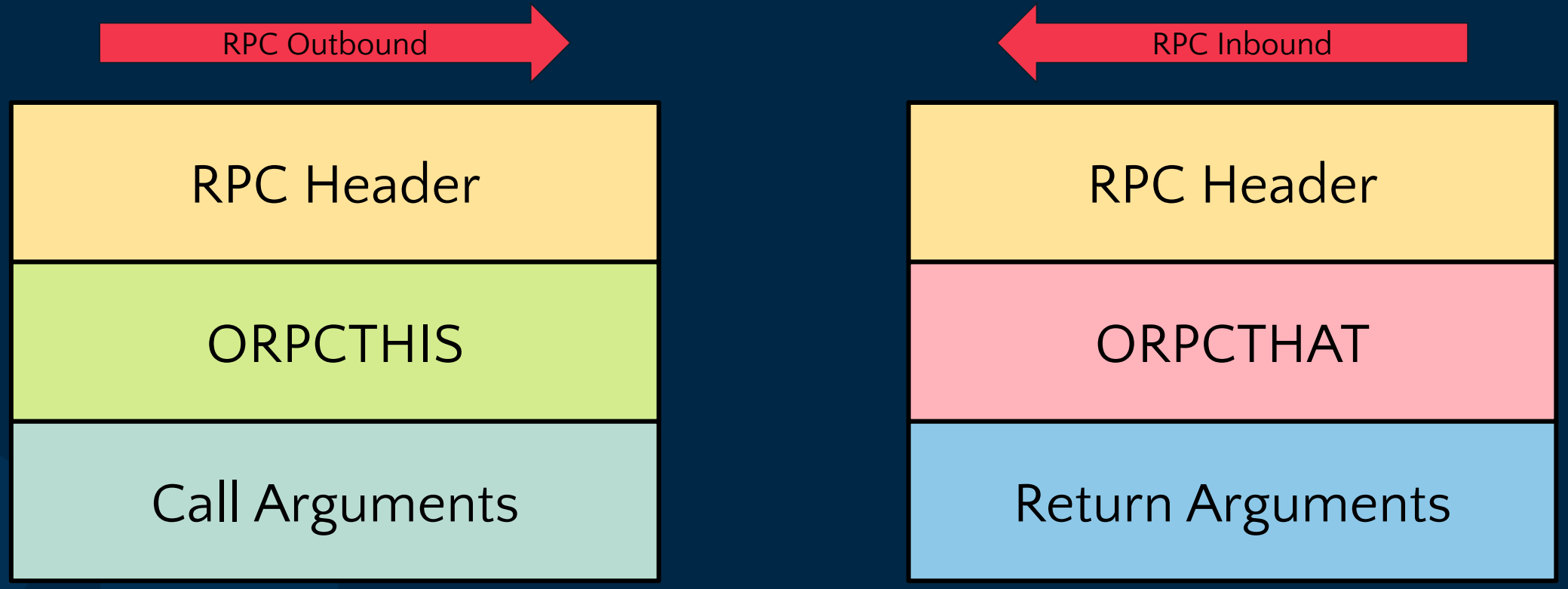
- Accurate representation of underlying proxy call
- Can't crash process†

Cons:

- Need to implement the RPC transport

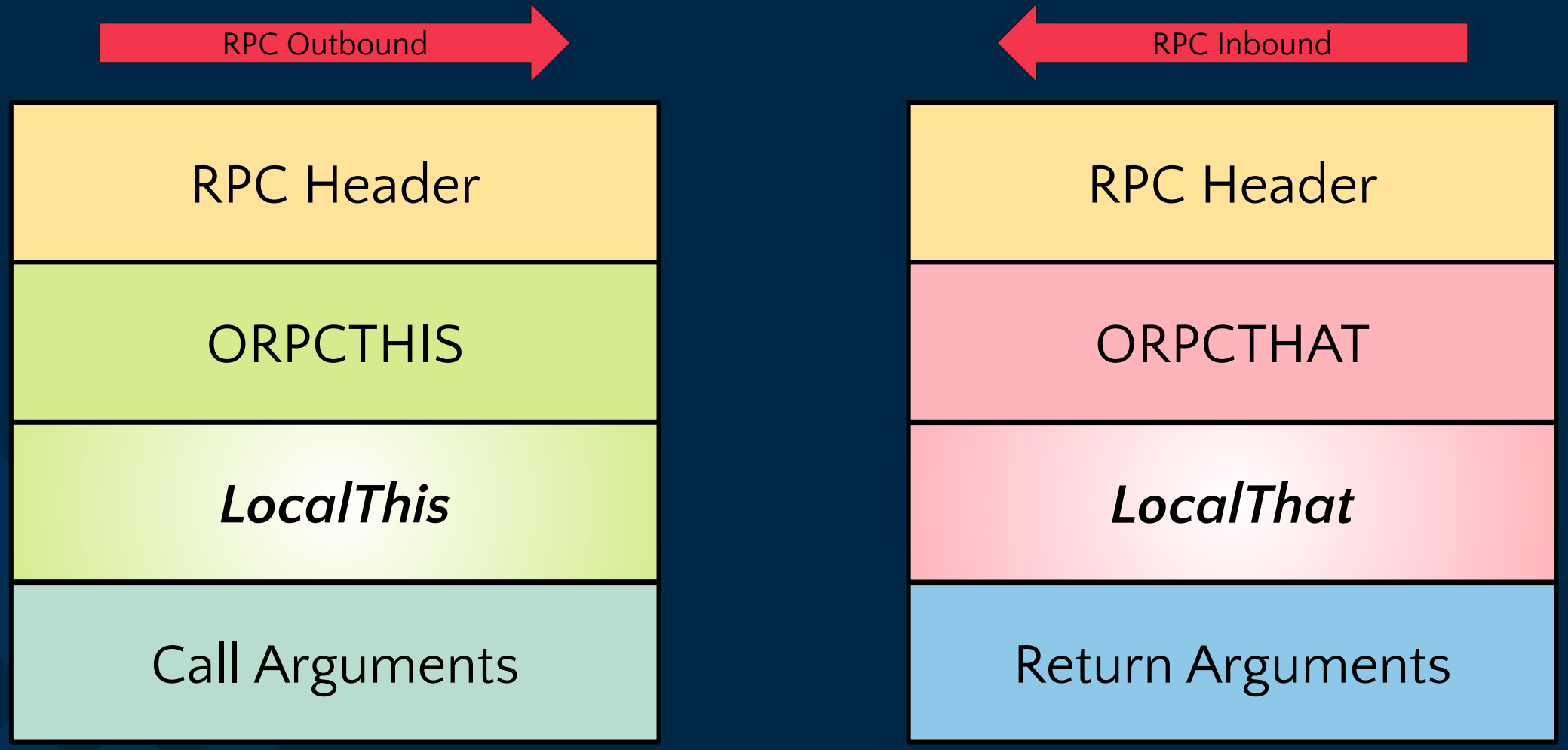
† *Never say never*

DCOM RPC Protocol



https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-dcom/86b9cf84-df2e-4f0b-ac22-1b957627e1ca

"ALPC" DCOM RPC Protocol



https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-dcom/86b9cf84-df2e-4f0b-ac22-1b957627e1ca

Reusing IRpcChannelBuffer

```
void NdrProxyInitialize(
  [in]       void                 *This,
  [in]       PRPC_MESSAGE         pRpcMsg,
  [in, out]  PMIDL_STUB_MESSAGE   pStubMsg,
  [in]       PMIDL_STUB_DESC      pStubDescriptor,
  [in]       unsigned int         ProcNum
);
```

Pass pointer to proxy

Reusing IRpcChannelBuffer

```
void NdrProxyInitialize(
  [in]       void                 *This,
  [in]       PRPC_MESSAGE         pRpcMsg,
  [in, out]  PMIDL_STUB_MESSAGE   pStubMsg,
  [in]       PMIDL_STUB_DESC      pStubDescriptor,
  [in]       unsigned int         ProcNum
);
```

```
struct MIDL_STUB_MESSAGE {
  // ...
  IRpcChannelBuffer* pRpcChannelBuffer;
  // ...
};
```

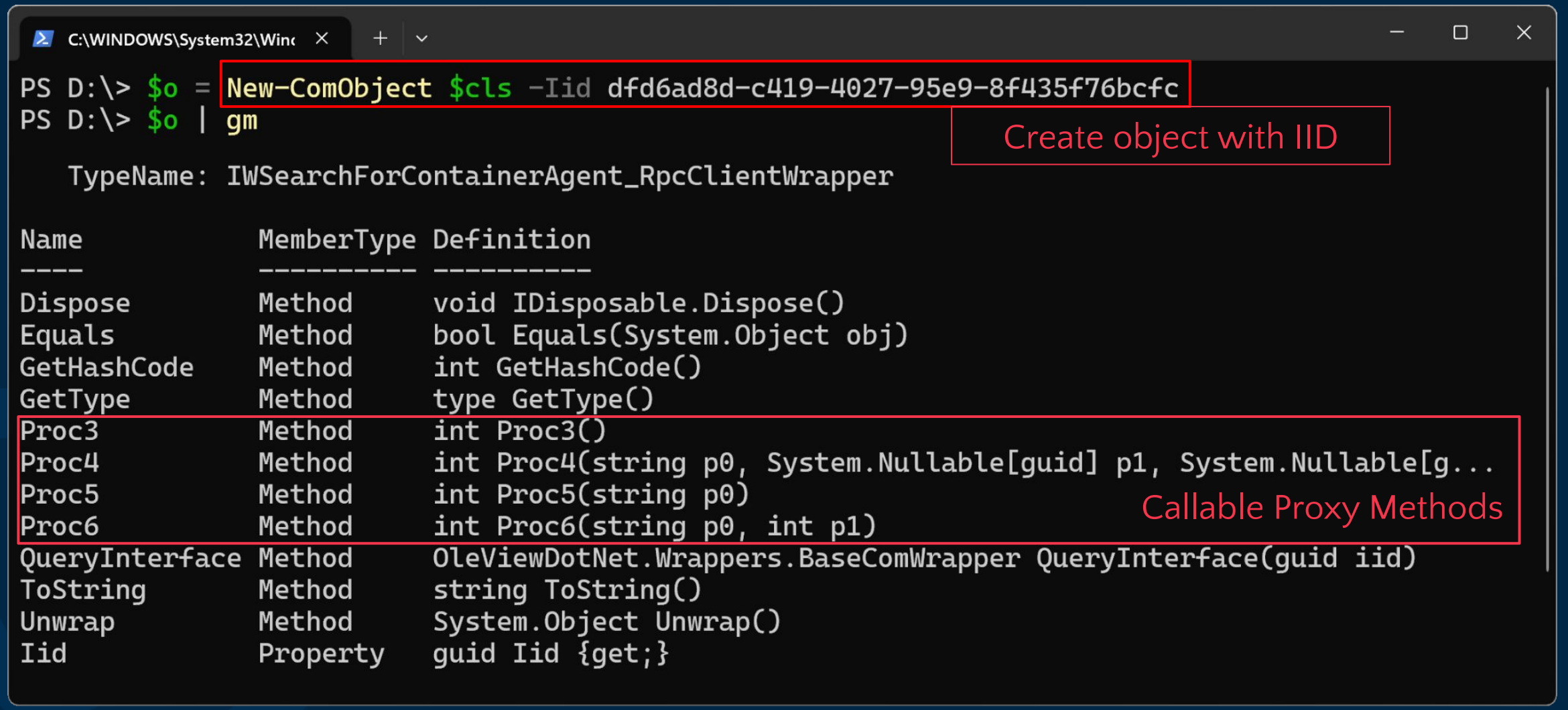
Reusing IRpcChannelBuffer

```
void NdrProxyInitialize(
  [in]       void                 *This,
  [in]       PRPC_MESSAGE         pRpcMsg,
  [in, out] PMIDL_STUB_MESSAGE pStubMsg,
  [in]       PMIDL_STUB_DESC      pStubDescrip
  [in]       unsigned int         ProcNum
);
```

```
struct MIDL_STUB_MESSAGE {
  // ...
  IRpcChannelBuffer* pRpcChannelBuffer;
  // ...
};
```

```
public byte[] SendReceive(byte[] ndr_data, int proc_num) {
  RPC_MESSAGE msg = new()
  {
      BufferLength = ndr_data.Length,
      ProcNum = proc_num
  };
  m_buffer.GetBuffer(ref msg, in m_iid);
  Marshal.Copy(ndr_data, 0, msg.Buffer, ndr_data.Length);
  m_buffer.SendReceive(ref msg, out int status);
  byte[] ret = new byte[msg.BufferLength];
  Marshal.Copy(msg.Buffer, ret, 0, msg.BufferLength);
  m_buffer.FreeBuffer(ref msg);
  return ret;
}
```

Putting it All Together



DEMO

Conclusions

Get the Tooling

GITHub Source
https://oleview.net

PS> Install-Module
"OleViewDotNet"

Microsoft

BLUEHAT

SECURITY ABOVE ALL ELSE