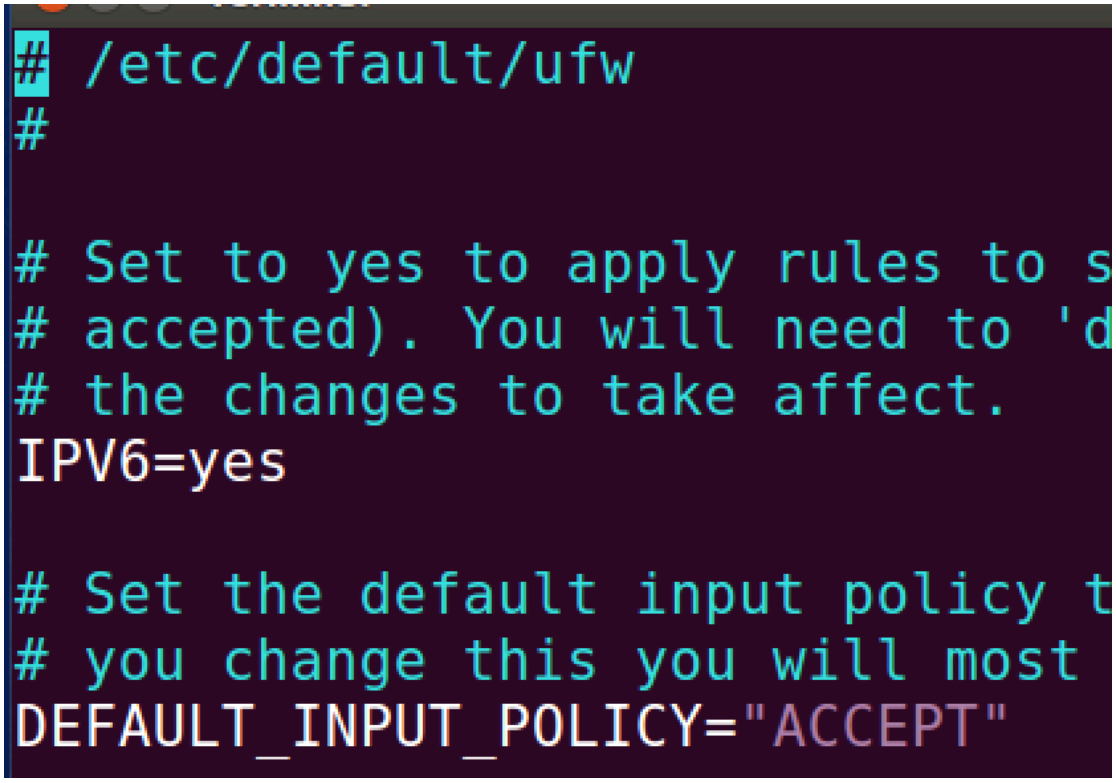
lab6-report

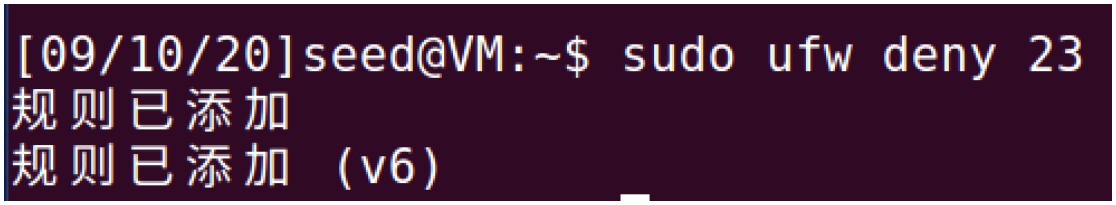
57117227 邵长捷

Linux Firewall Exploration Lab

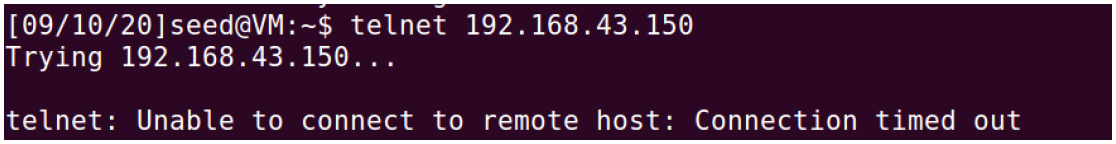
Task 1: Using Firewall



将/etc/default/ufw 文件的 DEFAULT_INPUT_POLICY 设置为 ACCEPT。

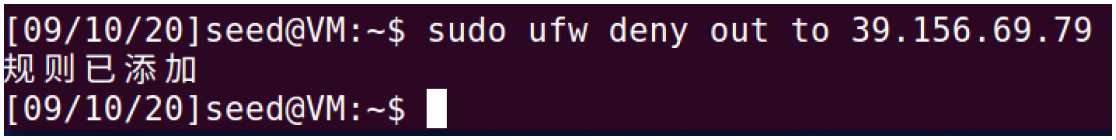


在主机 A 中添加一条规则，禁止访问本机的 23 号端口即 Telnet 服务。

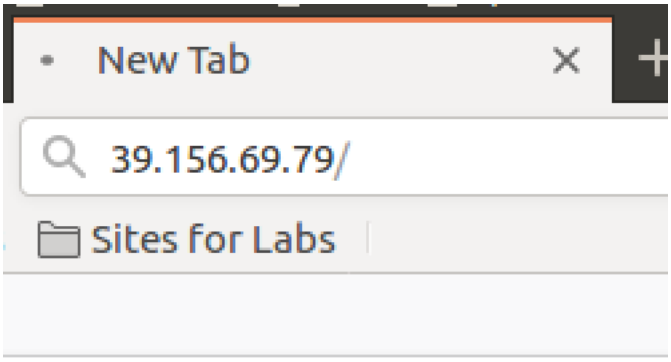


可见主机 B 无法访问主机 A 的 Telnet 服务。

同理在主机 B 中添加一条相同的规则，可以禁止主机 A 访问主机 B 的 Telnet 服务。



在主机 A 中 ufw 添加上述规则，即可阻止对 39.156.69.79（baidu.com）的访问。



可见，浏览器无法加载出 39.156.69.79 的网页，注意要提前清空浏览器相应的缓存才可以成功实验。

Task 2: Implementing a Simple Firewall

编写上述脚本，核心功能为 if 判断部分，钩子设为 NF_INET_PRE_ROUTING，以便过滤所有除了混杂模式以外的数据包。

下面举两个例子说明内核模块的功能。

```
  if (iph->protocol == IPPROTO_TCP && tcph->dest == htons(23) ) {
    printk(KERN_INFO "Dropping telnet packet to %d.%d.%d.%d\n",
        ((unsigned char *)&iph->daddr)[0],
        ((unsigned char *)&iph->daddr)[1],
        ((unsigned char *)&iph->daddr)[2],
        ((unsigned char *)&iph->daddr)[3]);
    return NF_DROP;
  }
else if(iph->saddr==*(__be32*)drop_ip)
{
  printk(KERN_INFO "Dropped packet from %d.%d.%d.%d\n",*drop_ip,*(drop_ip+1),*(d
rop_ip+2),*(drop_ip+3));
  return NF_DROP;
}
else {
    return NF_ACCEPT;
  }
```

上述核心代码实现拦截通往端口 23（Telnet 服务）的数据包。

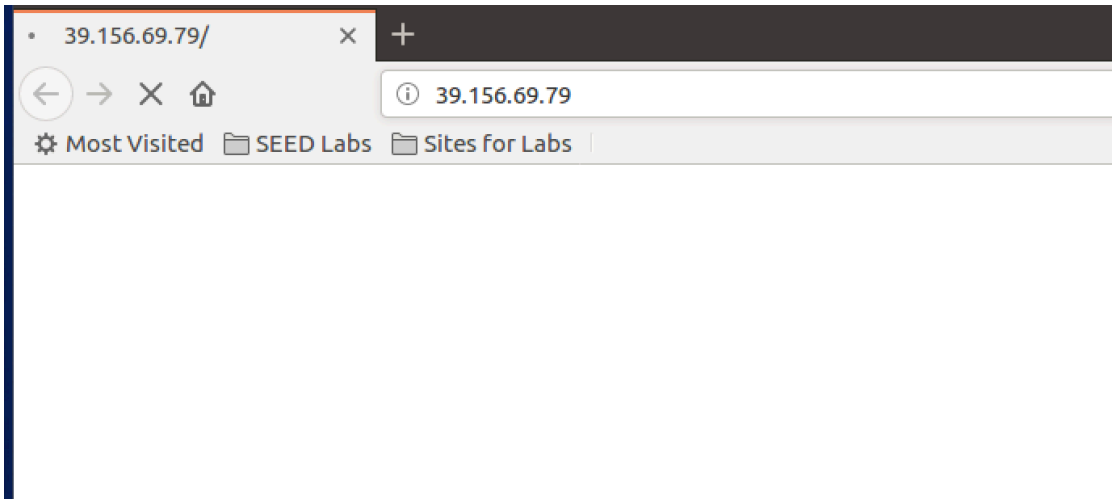
主机 B 尝试对主机 A 发起 Telnet 连接。

```
[  234.909030] Registering a Telnet filter.
[  241.170935] Dropping telnet packet to 192.168.43.150
[  242.178387] Dropping telnet packet to 192.168.43.150
[  244.194856] Dropping telnet packet to 192.168.43.150
[  248.417865] Dropping telnet packet to 192.168.43.150
[  256.610530] Dropping telnet packet to 192.168.43.150
[  272.737969] Dropping telnet packet to 192.168.43.150
[  306.786105] Dropping telnet packet to 192.168.43.150
```

在主机 A 终端输入 dmesg 命令查看到对 Telnet 数据包的成功拦截。

```
  if(((ntohl(iph->saddr) & 0xff000000) >> 24 == 39)
 && ((ntohl(iph->saddr) & 0x00ff0000) >> 16 == 156)
 && ((ntohl(iph->saddr) & 0x0000ff00) >> 8 == 69)
 && ((ntohl(iph->saddr) & 0x000000ff) == 79))
{
  //printk(KERN_INFO "Dropped packet from %d.%d.%d.%d\n",*dro
p_ip,*(drop_ip+1),*(drop_ip+2),*(drop_ip+3));

  printk(KERN_INFO "Dropped packet from 39.156.69.79");
  return NF_DROP;
}
else {
    return NF_ACCEPT;
  }
}
```

编写上述代码，实现拦截来自 IP 地址为 39.156.69.79(baidu.com)的数据包，注意将 iph->saddr 进行大小端的转换。



编写 Makefile 文件并输入 make 命令进行编译，将编译好的可加载内核模块安装后，清空浏览器缓存后，尝试访问 39.156.69.79，无法正常访问。

```
[ 5843.127346] IP filter is being removed.
[ 5845.125869] Registering a IP filter.
[ 5864.506878] Dropped packet from 39.156.69.79
[ 5864.550924] Dropped packet from 39.156.69.79
[ 5865.183650] Dropped packet from 39.156.69.79
[ 5865.441193] Dropped packet from 39.156.69.79
[ 5865.486466] Dropped packet from 39.156.69.79
[ 5865.601937] Dropped packet from 39.156.69.79
[ 5866.203391] Dropped packet from 39.156.69.79
[ 5866.461229] Dropped packet from 39.156.69.79
[ 5867.526622] Dropped packet from 39.156.69.79
[ 5867.596410] Dropped packet from 39.156.69.79
[ 5868.223278] Dropped packet from 39.156.69.79
[ 5868.481118] Dropped packet from 39.156.69.79
[ 5871.721879] Dropped packet from 39.156.69.79
[ 5871.728640] Dropped packet from 39.156.69.79
```

dmesg 显示成功拦截来源为 39.156.69.79 的数据包。

同理，除上述拦截外，还实现了对 IP 地址 129.211.129.109 (4399.com)和对 IP 地址 119.3.238.64(bilibili.com)的访问拦截，对端口 22（ssh 服务）的拒绝访问，共成功实现了 5 个拦截规则。

Task 3: Evading Egress Filtering

主机 A：192.168.43.150

主机 B：192.168.43.88

```
[09/12/20]seed@VM:~/.../exp5$ sudo ufw deny out to any port 23
规则已添加
规则已添加 (v6)
[09/12/20]seed@VM:~/.../exp5$ telnet 192.168.43.88
Trying 192.168.43.88...
telnet: Unable to connect to remote host: Connection timed out
```

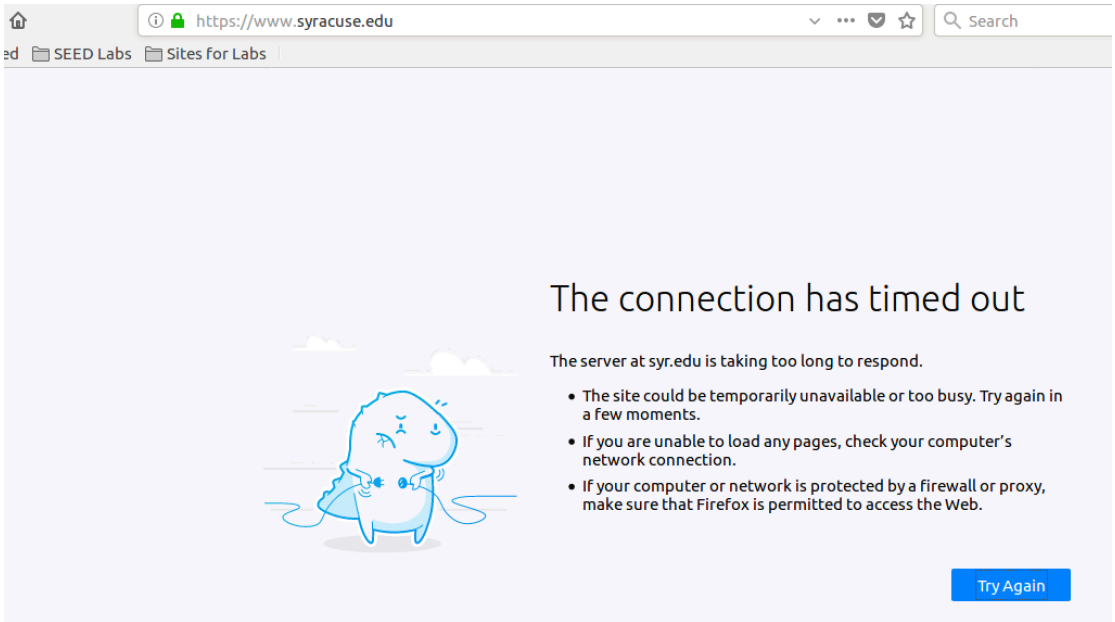
在主机 A 的 ufw 中添加上述规则，阻止主机 A 访问任何 Telnet 服务器。

```
[09/12/20]seed@VM:~$ dig www.syr.edu

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.syr.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29708
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.syr.edu.                   IN      A

;; ANSWER SECTION:
www.syr.edu.            60      IN      CNAME   syr.edu.
syr.edu.                60      IN      A       128.230.18.200
```

利用 dig 查看 www.syr.edu 的 ip 地址。

```
[09/12/20]seed@VM:~/.../exp5$ sudo ufw deny out to 128.230.18.200
规则已添加
```

在主机 A 的 ufw 中添加上述规则，拦截对 www.syr.edu 的访问。



尝试通过浏览器访问 www.syr.edu，不能正常访问，拦截成功。

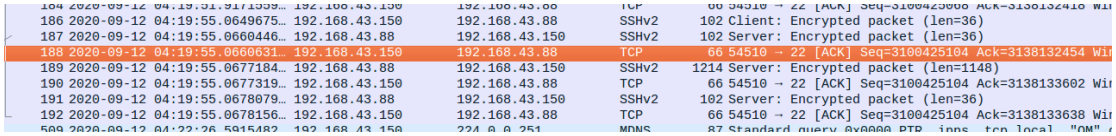
```
[09/12/20]seed@VM:~$ ssh -L 8000:192.168.43.88:23 seed@192.16
8.43.88
seed@192.168.43.88's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i68
6)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

0 packages can be updated.
0 updates are security updates.

Last login: Sat Sep 12 03:58:09 2020 from 192.168.43.150
$ exit
```

Task 3.a: Telnet to Machine B through the firewall

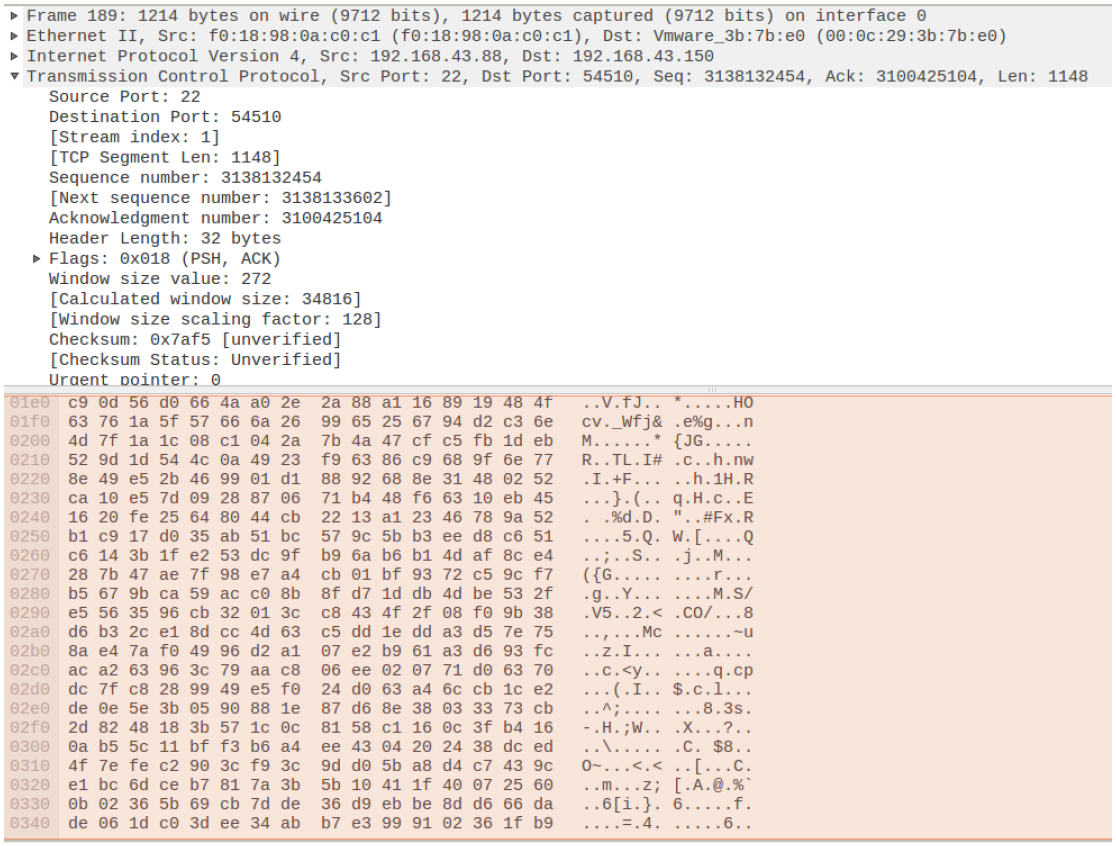
首先在主机 A 中输入上述 ssh 命令。

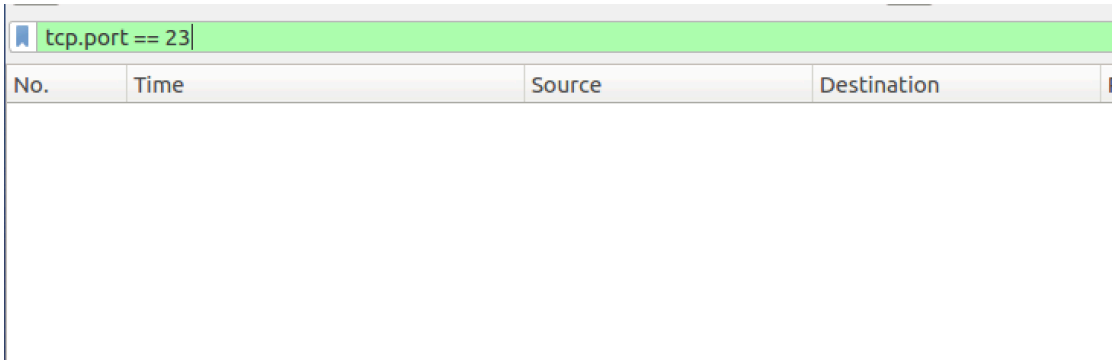
```
[09/12/20]seed@VM:~$ telnet localhost 8000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Sat Sep 12 04:13:41 EDT 2020 from 192.168.43.150
on pts/18
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i68
6)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

0 packages can be updated.
0 updates are security updates.

$ ls
android         Documents         lib       source
bin             Downloads         Music     Templates
Customization   examples.desktop  Pictures  Videos
Desktop         get-pip.py        Public
$ cd Desktop
$ ls
exp1 setuid  exp2 stack overflow  exp3  exp4
$ ifconfig
ens33     Link encap:Ethernet  HWaddr 00:0c:29:3c:ad:20
          inet addr:192.168.43.88  Bcast:192.168.43.255  Mask
```

在主机 A 中另开一个终端，对本机的 8000 端口尝试进行 Telnet 连接，可以看到连接到了 192.168.43.88，即主机 B。



在 Wireshark 中抓包分析。可以根据 len 判断 189 号 ssh 加密包返回了 ifconfig 的信息。



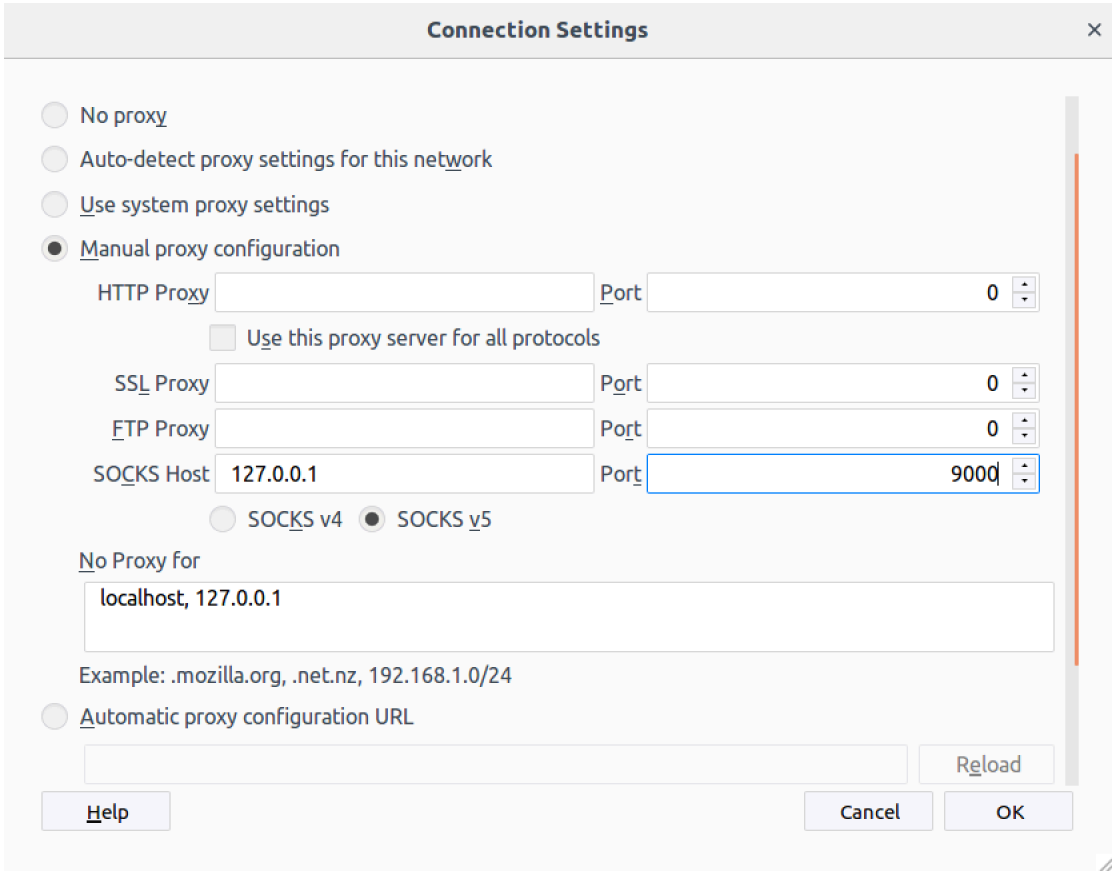


进一步观察数据包内容，可以看到数据部分进行了加密处理，不存在任何可以被 ufw 检测到的有关端口 23 的信息，因而可以实现从主机 A 的本地 IP 端口 8000 映射到目标主机 IP23 端口，即使用了 Telnet 服务。

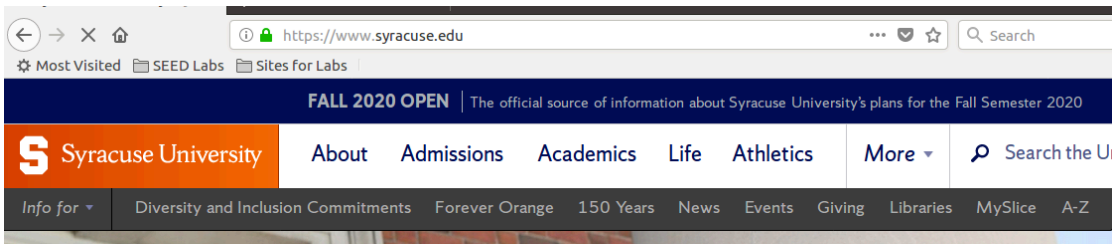
Task 3.b: Connect to syr.edu using SSH Tunnel

```
[09/12/20]seed@VM:~$ ssh -D 9000 -C seed@192.168.43.88
seed@192.168.43.88's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i68
6)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

0 packages can be updated.
0 updates are security updates.

Last login: Sat Sep 12 04:19:40 2020 from bogon
$
```

采用动态端口转发的方式，映射本地主机 A 的 9000 端口到主机 B。

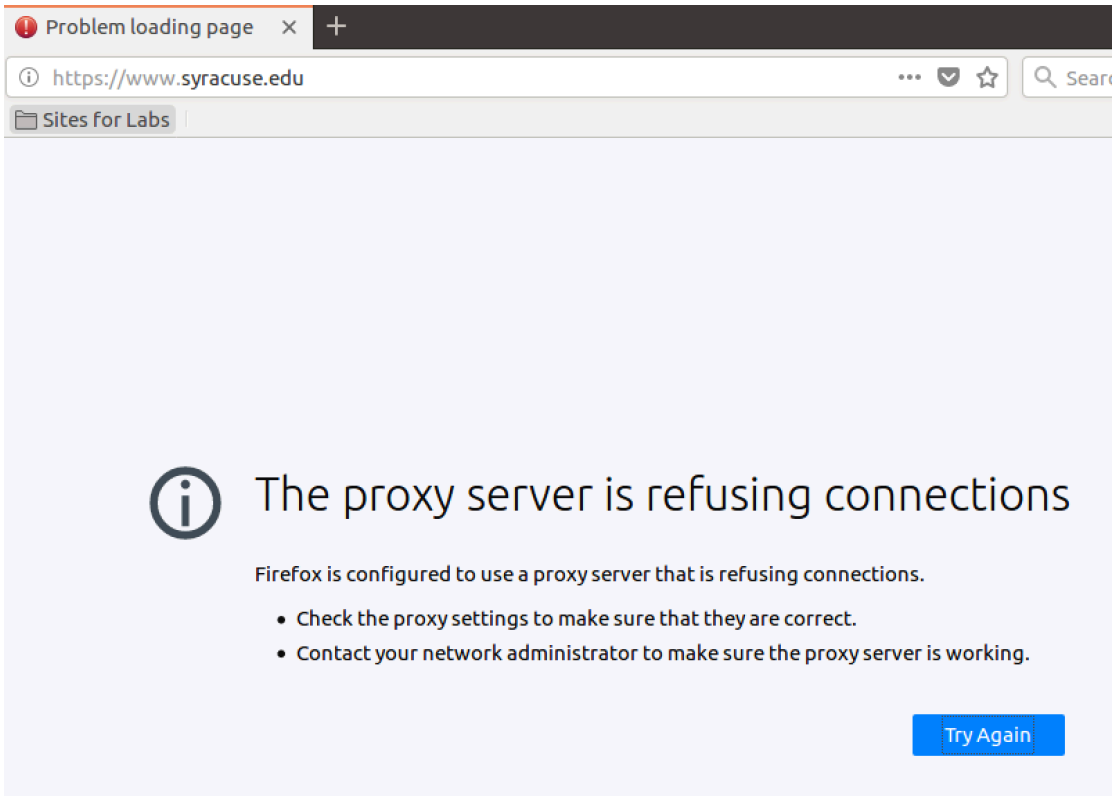


在 Firefox 浏览器中设置代理到本地端口 9000。

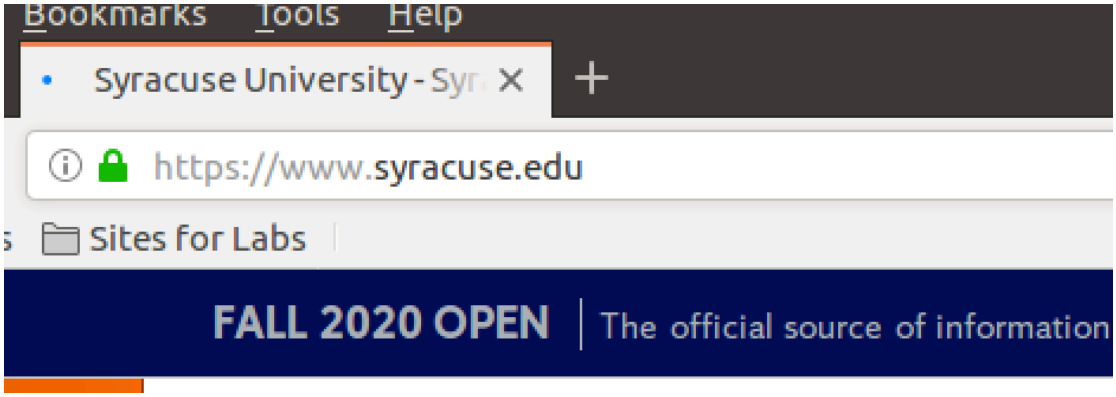


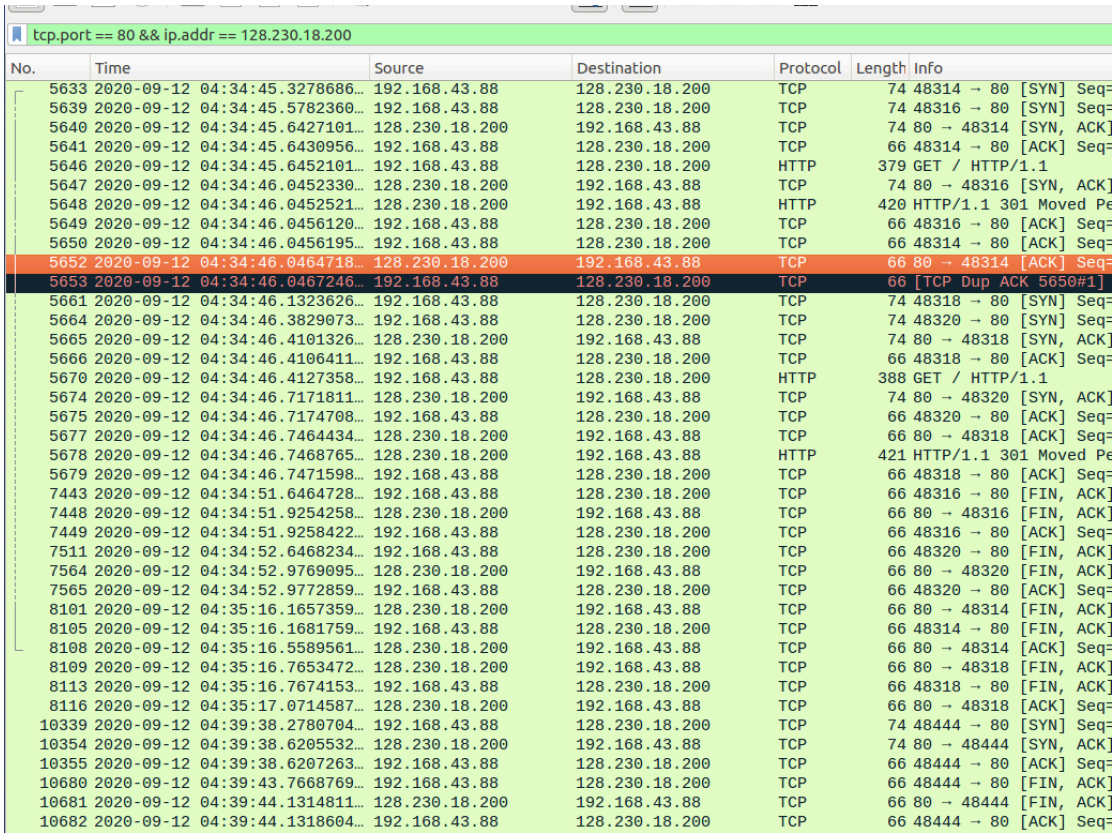
成功访问先前被 ufw 拦截的网站 www.syr.edu。

断开与主机 B 的 ssh 连接，并清空浏览器缓存，再次访问目标网站则不能访问。



重新连接 ssh 后，再次尝试访问目标网站，则恢复正常访问。





tcp.port == 80 && ip.addr == 128.230.18.200

No.	Time	Source	Destination	Protocol	Length	Info
5633	2020-09-12 04:34:45.3278686…	192.168.43.88	128.230.18.200	TCP	74	48314 → 80 [SYN] Seq=
5639	2020-09-12 04:34:45.5782360…	192.168.43.88	128.230.18.200	TCP	74	48316 → 80 [SYN] Seq=
5640	2020-09-12 04:34:45.6427101…	128.230.18.200	192.168.43.88	TCP	74	80 → 48314 [SYN, ACK]
5641	2020-09-12 04:34:45.6430956…	192.168.43.88	128.230.18.200	TCP	66	48314 → 80 [ACK] Seq=
5646	2020-09-12 04:34:45.6452101…	192.168.43.88	128.230.18.200	HTTP	379	GET / HTTP/1.1
5647	2020-09-12 04:34:46.0452330…	128.230.18.200	192.168.43.88	TCP	74	80 → 48316 [SYN, ACK]
5648	2020-09-12 04:34:46.0452521…	128.230.18.200	192.168.43.88	HTTP	420	HTTP/1.1 301 Moved Pe
5649	2020-09-12 04:34:46.0456120…	192.168.43.88	128.230.18.200	TCP	66	48316 → 80 [ACK] Seq=
5650	2020-09-12 04:34:46.0456195…	192.168.43.88	128.230.18.200	TCP	66	48314 → 80 [ACK] Seq=
5652	2020-09-12 04:34:46.0464718…	128.230.18.200	192.168.43.88	TCP	66	80 → 48314 [ACK] Seq=
5653	2020-09-12 04:34:46.0467246…	192.168.43.88	128.230.18.200	TCP	66	[TCP Dup ACK 5650#1]
5661	2020-09-12 04:34:46.1323626…	192.168.43.88	128.230.18.200	TCP	74	48318 → 80 [SYN] Seq=
5664	2020-09-12 04:34:46.3829073…	192.168.43.88	128.230.18.200	TCP	74	48320 → 80 [SYN] Seq=
5665	2020-09-12 04:34:46.4101326…	128.230.18.200	192.168.43.88	TCP	74	80 → 48318 [SYN, ACK]
5666	2020-09-12 04:34:46.4106411…	192.168.43.88	128.230.18.200	TCP	66	48318 → 80 [ACK] Seq=
5670	2020-09-12 04:34:46.4127358…	192.168.43.88	128.230.18.200	HTTP	388	GET / HTTP/1.1
5674	2020-09-12 04:34:46.7171811…	128.230.18.200	192.168.43.88	TCP	74	80 → 48320 [SYN, ACK]
5675	2020-09-12 04:34:46.7174708…	192.168.43.88	128.230.18.200	TCP	66	48320 → 80 [ACK] Seq=
5677	2020-09-12 04:34:46.7464434…	128.230.18.200	192.168.43.88	TCP	66	80 → 48318 [ACK] Seq=
5678	2020-09-12 04:34:46.7468765…	128.230.18.200	192.168.43.88	HTTP	421	HTTP/1.1 301 Moved Pe
5679	2020-09-12 04:34:46.7471598…	192.168.43.88	128.230.18.200	TCP	66	48318 → 80 [ACK] Seq=
7443	2020-09-12 04:34:51.6464728…	192.168.43.88	128.230.18.200	TCP	66	48316 → 80 [FIN, ACK]
7448	2020-09-12 04:34:51.9254258…	128.230.18.200	192.168.43.88	TCP	66	80 → 48316 [FIN, ACK]
7449	2020-09-12 04:34:51.9258422…	192.168.43.88	128.230.18.200	TCP	66	48316 → 80 [ACK] Seq=
7511	2020-09-12 04:34:52.6468234…	192.168.43.88	128.230.18.200	TCP	66	48320 → 80 [FIN, ACK]
7564	2020-09-12 04:34:52.9769095…	128.230.18.200	192.168.43.88	TCP	66	80 → 48320 [FIN, ACK]
7565	2020-09-12 04:34:52.9772859…	192.168.43.88	128.230.18.200	TCP	66	48320 → 80 [ACK] Seq=
8101	2020-09-12 04:35:16.1657359…	128.230.18.200	192.168.43.88	TCP	66	80 → 48314 [FIN, ACK]
8105	2020-09-12 04:35:16.1681759…	192.168.43.88	128.230.18.200	TCP	66	48314 → 80 [FIN, ACK]
8108	2020-09-12 04:35:16.5589561…	128.230.18.200	192.168.43.88	TCP	66	80 → 48314 [ACK] Seq=
8109	2020-09-12 04:35:16.7653472…	128.230.18.200	192.168.43.88	TCP	66	80 → 48318 [FIN, ACK]
8113	2020-09-12 04:35:16.7674153…	192.168.43.88	128.230.18.200	TCP	66	48318 → 80 [FIN, ACK]
8116	2020-09-12 04:35:17.0714587…	128.230.18.200	192.168.43.88	TCP	66	80 → 48318 [ACK] Seq=
10339	2020-09-12 04:39:38.2780704…	192.168.43.88	128.230.18.200	TCP	74	48444 → 80 [SYN] Seq=
10354	2020-09-12 04:39:38.6205532…	128.230.18.200	192.168.43.88	TCP	74	80 → 48444 [SYN, ACK]
10355	2020-09-12 04:39:38.6207263…	192.168.43.88	128.230.18.200	TCP	66	48444 → 80 [ACK] Seq=
10680	2020-09-12 04:39:43.7668769…	192.168.43.88	128.230.18.200	TCP	66	48444 → 80 [FIN, ACK]
10681	2020-09-12 04:39:44.1314811…	128.230.18.200	192.168.43.88	TCP	66	80 → 48444 [FIN, ACK]
10682	2020-09-12 04:39:44.1318604…	192.168.43.88	128.230.18.200	TCP	66	48444 → 80 [ACK] Seq=

观察 Wireshark 与目标网站 IP 地址 128.230.18.200 相关的数据包，全部都只和主机 B 的 IP 地址 192.168.43.88 有关，证明了主机 A 成功访问该网站是利用了端口转发技术，通过将本地 9000 端口与主机 B 进行绑定，并设置浏览器代理为本地 9000 端口实现绕过 ufw 防火墙的拦截。

Task 4: Evading Ingress Filtering

```
[09/12/20]seed@VM:~$ sudo ufw deny 22
规则已添加
规则已添加 (v6)
```

```
[09/12/20]seed@VM:~$ sudo ufw deny 80
规则已添加
规则已添加 (v6)
[09/12/20]seed@VM:~$
```

在主机 A 中添加上述规则，将自己设置为“内网”Web 服务器，且拦截 ssh 服务。

```
[09/12/20]seed@VM:~$ nmap 192.168.43.150

Starting Nmap 7.01 ( https://nmap.org ) at 2020-09-12 06:03 EDT
Nmap scan report for VM (192.168.43.150)
Host is up (0.0026s latency).
Not shown: 991 closed ports
PORT     STATE    SERVICE
21/tcp   open     ftp
22/tcp   filtered ssh
23/tcp   open     telnet
53/tcp   open     domain
80/tcp   filtered http
513/tcp  open     login
514/tcp  open     shell
3128/tcp open     squid-http
8080/tcp open     http-proxy

Nmap done: 1 IP address (1 host up) scanned in 1.34 seconds
```

在主机 B 中使用 Nmap 可以看到主机 A 的 22 端口和 80 端口被防火墙过滤。

```
[09/12/20]seed@VM:~$ ssh -f -N -R 10000:localhost:22 seed@192.168.43.88
seed@192.168.43.88's password:
```

在主机 A 中运行上述命令，将主机 B 的端口 10000 转发到本地的端口 22

```
[09/12/20]seed@VM:~$ ssh seed@localhost -p 10000
The authenticity of host '[localhost]:10000 ([127.0.0.1]:10000)' can't be establ
ished.
ECDSA key fingerprint is SHA256:p1zAio6c1bI+8HDp5xa+eKRi561aFDaPE1/xq1eYzCI.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[localhost]:10000' (ECDSA) to the list of known host
s.
seed@localhost's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

0 packages can be updated.
0 updates are security updates.

Last login: Thu Sep 10 08:07:07 2020 from bogon
$ ifconfig
ens33     Link encap:以太网  硬件地址 00:0c:29:3b:7b:e0
          inet 地址:192.168.43.150  广播:192.168.43.255  掩码:255.255.255.0
```

主机 B 运行上述命令，通过反向 ssh 隧道成功建立与主机 A 的 ssh 连接。

Firewall Evasion Lab: Bypassing Firewalls using VPN

Task 1: VM Setup

主机 A：192.168.43.150　VPN client

主机 B：192.168.43.88　VPN server

Task 2: Set up Firewall

```
[09/13/20]seed@VM:~$ sudo ufw deny out on ens33 to 128.230.18.0/24
规则已添加
[09/13/20]seed@VM:~$ ping 128.230.18.200
PING 128.230.18.200 (128.230.18.200) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^Z
[1]+  已停止               ping 128.230.18.200
[09/13/20]seed@VM:~$
```

在主机 A 上启用 ufw 防火墙并添加上图规则，尝试对 128.230.18.200(syr.edu)进行 ping，无法 ping 通，防火墙设置生效。

Task 3: Bypassing Firewall using VPN

Step 1: Run VPN Server

```
[09/13/20]seed@VM:~/.../exp6$ sudo ./vpnserver
```

在主机 B 中编译 vpn_server.c 并运行程序。ifconfig -a 可以查看到未配置的 tun0 接口。

```
[09/13/20]seed@VM:~/.../exp5$ sudo sysctl net.ipv4.ip_forward
=1
net.ipv4.ip_forward = 1
```

```
[09/13/20]seed@VM:~$ sudo ifconfig tun0 192.168.53.88/24 up
```

运行上述命令将 tun0 的 ip 地址设为 192.168.53.88，并打开主机 B 的端口转发功能。

Step 2: Run VPN Client

```
[09/13/20]seed@VM:~/.../exp6$ sudo ./vpnclient
```

```
[09/13/20]seed@VM:~$ sudo ifconfig tun0 192.168.53.150/24 up
```

在客户端（主机 A）运行上述命令。

Step 3: Set Up Routing on Client and Server VMs

```
[09/13/20]seed@VM:~$ sudo route add -net 128.230.18.0/24 tun0
```

在 Client 中添加路由，将发往子网 128.230.18.0 的数据包设置为由 tun0 接口发出。

Step 4: Set Up NAT on Server VM

```
[09/13/20]seed@VM:~$ sudo iptables -F
[09/13/20]seed@VM:~$ sudo iptables -t nat -F
[09/13/20]seed@VM:~$ sudo iptables -t nat -A POSTROUTING -j MASQUERADE -o ens33
```

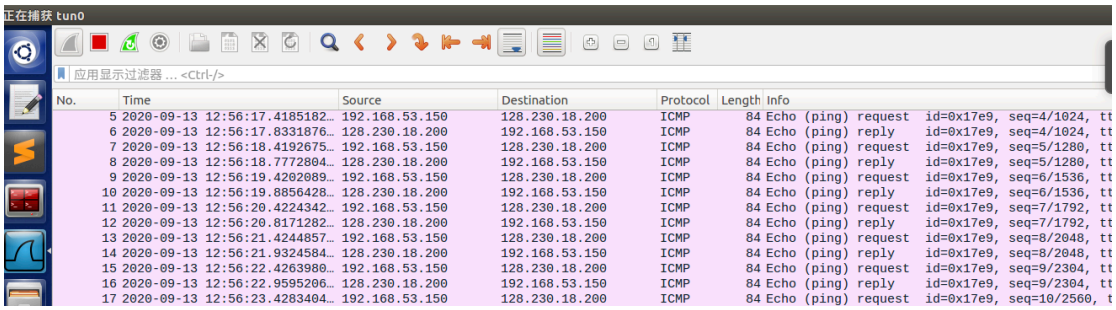
在 Server 中运行上述命令，在主机 B 增加一个 NAT，出口为网卡 ens33。

```
[09/13/20]seed@VM:~$ ping 128.230.18.200
PING 128.230.18.200 (128.230.18.200) 56(84) bytes of data.
64 bytes from 128.230.18.200: icmp_seq=3 ttl=42 time=603 ms
64 bytes from 128.230.18.200: icmp_seq=4 ttl=42 time=415 ms
64 bytes from 128.230.18.200: icmp_seq=5 ttl=42 time=359 ms
64 bytes from 128.230.18.200: icmp_seq=6 ttl=42 time=466 ms
64 bytes from 128.230.18.200: icmp_seq=7 ttl=42 time=396 ms
64 bytes from 128.230.18.200: icmp_seq=8 ttl=42 time=510 ms
64 bytes from 128.230.18.200: icmp_seq=9 ttl=42 time=535 ms
64 bytes from 128.230.18.200: icmp_seq=10 ttl=42 time=459 ms
64 bytes from 128.230.18.200: icmp_seq=11 ttl=42 time=471 ms
64 bytes from 128.230.18.200: icmp_seq=12 ttl=42 time=391 ms
64 bytes from 128.230.18.200: icmp_seq=13 ttl=42 time=412 ms
```

现在，在 Client 中尝试对目标网站 IP 地址进行 ping，发现可以 ping 通。

```
[09/13/20]seed@VM:~$ sudo ufw status
状态：  激活

至                          动作          来自
-                          --          --
128.230.18.0/24            DENY OUT    Anywhere on ens33
```

再次确认 Client 中 ufw 防火墙的开启和对访问目标网站数据的拦截。



在 Server 中的 Wireshark 对 tun0 接口抓包可以看到，ICMP echo request 报文的源 IP 地址是 Client 虚拟接口对应的 IP 地址 192.168.53.150，证明了 VPN 功能的存在和实验的成功。