Firewall Exploration Lab Report

Lab Environment Setup

Machine A 10.0.2.4
Machine B 10.0.2.7

Task 1: Using Firewall

使用ufw代替iptables [1]

```
[11/30/20]seed@VM:~$ sudo dpkg --get-selections | grep ufw
ufw                                             install
[11/30/20]seed@VM:~$ sudo ufw status
Status: inactive
[11/30/20]seed@VM:~$ sudo ufw enable
Firewall is active and enabled on system startup
[11/30/20]seed@VM:~$ sudo ufw default deny
Default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
[11/30/20]seed@VM:~$ sudo ufw status
Status: active
```

Prevent A from doing telnet to Machine B

```
$ sudo ufw deny out from 10.0.2.4 to 10.0.2.7 port 23
```

```
[11/30/20]seed@VM:~$ sudo ufw deny out from 10.0.2.7 to 10.0.2.
4 port 23
Rule added
[11/30/20]seed@VM:~$ sudo ufw deny out from 10.0.2.4 to 10.0.2.
7 port 23
Rule added
[11/30/20]seed@VM:~$ telnet 10.0.2.7
Trying 10.0.2.7...
```

Prevent B from doing telnet to Machine A

```
$ sudo ufw deny out from 10.0.2.7 to 10.0.2.4 port 23
```

```
[11/30/20]seed@VM:~$
[11/30/20]seed@VM:~$ telnet 10.0.2.4
Trying 10.0.2.4...
```

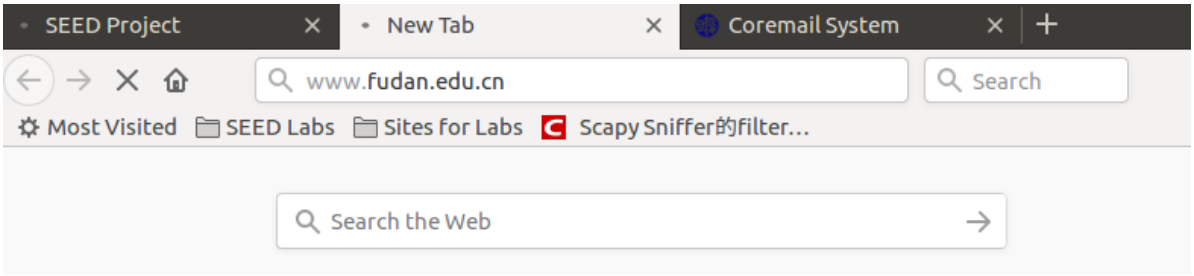
Prevent A from visiting an external web site

因为一般的商用网站都会有不止一个IP地址，所以这里用了学校官网作为外部网址。通过ping命令得到www.fudan.edu.cn的IP地址。

```
$ sudo ufw deny out from 10.0.2.4 to 202.120.224.81 port 80
```

```
[11/30/20]seed@VM:~$ ping www.fudan.edu.cn
PING www.fudan.edu.cn (202.120.224.81) 56(84) bytes of data.
64 bytes from 224.fudan.edu.cn (202.120.224.81): icmp_seq=1 ttl
=250 time=6.05 ms
64 bytes from 224.fudan.edu.cn (202.120.224.81): icmp_seq=2 ttl
=250 time=4.96 ms
^C
--- www.fudan.edu.cn ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 4.967/5.509/6.051/0.542 ms
[11/30/20]seed@VM:~$ sudo ufw deny out from 10.0.2.4 to 202.120
.224.81
Rule added
[11/30/20]seed@VM:~$
```

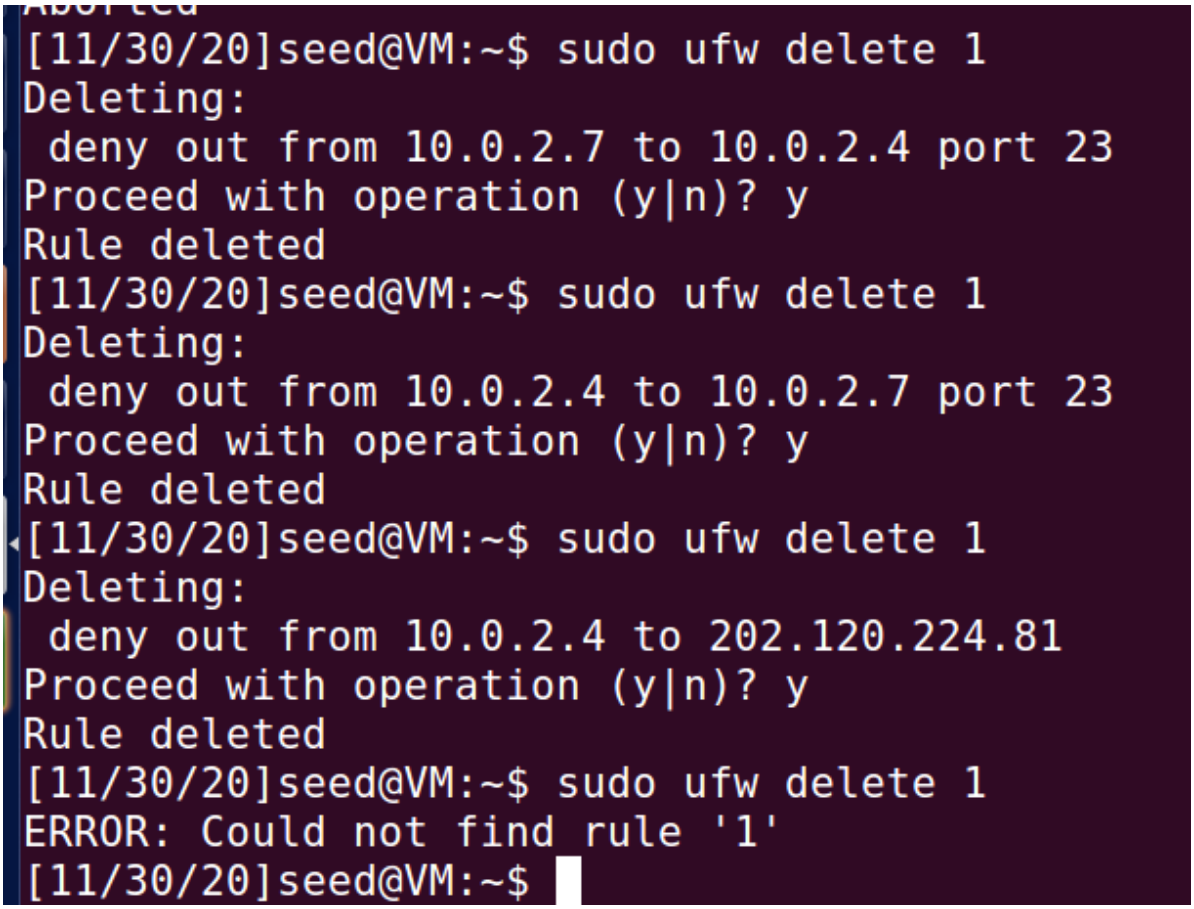
得到的结果如截图：www.fudan.edu.cn无法正常访问，但是其他网址如mail.fudan.edu.cn还是可以正常访问的。



Task 2: Implementing a Simple Firewall

首先重复以下命令来删除Task 1中用ufw添加的过滤规则

```
$ sudo ufw delete 1
```



Code

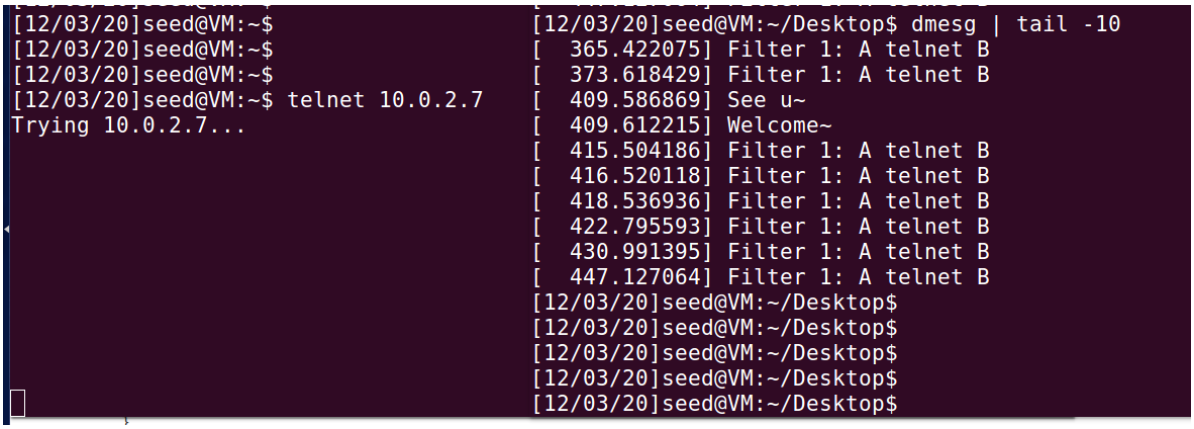
```
#include <linux/ip.h>
#include <linux/tcp.h>
#include <linux/icmp.h>
#include <linux/skbuff.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/netfilter.h>
#include <linux/netfilter_ipv4.h>

struct nf_hook_ops nfho_in;
struct nf_hook_ops nfho_out;

struct iphdr *ipHeader;
struct tcphdr *tcpHeader;
struct icmphdr *icmpHeader;

bool isAddressEqual(struct iphdr *ip, int srcORdst, int a, int b, int c, int d){
    bool res = true;
    if (srcORdst == 0) { // 判断src IP
        res &= ((ip->saddr & 0xff000000) >> 24 == d); // big endian
        res &= ((ip->saddr & 0x00ff0000) >> 16 == c);
        res &= ((ip->saddr & 0x0000ff00) >> 8 == b);
        res &= ((ip->saddr & 0x000000ff) == a);
    }
    else if (srcORdst == 1) { // 判断dst IP
        res &= ((ip->daddr & 0xff000000) >> 24 == d);
        res &= ((ip->daddr & 0x00ff0000) >> 16 == c);
        res &= ((ip->daddr & 0x0000ff00) >> 8 == b);
```

```
        res &= ((ip->daddr & 0x000000ff) == a);
    }
    return res;
}

unsigned int hook_func_in(void *priv, struct sk_buff *skb, const struct
nf_book_state *state){
    ipHeader = (struct iphdr *)skb_network_header(skb);
    if (ipHeader->protocol == 6) { // TCP
        tcpHeader = (struct tcphdr *)((__u32 *)ipHeader + ipHeader->ihl);
        // ip + ip首部长度
        unsigned int dst = (unsigned int)ntohs(tcpHeader->dest);

        // Filter 2: B telnet A
        if (dst == 23) { // telnet
            // check ip addr
            if (isAddressEqual(ipHeader,0,10,0,2,7) == false){
                printk(KERN_INFO "Filter 2: incorrect target src ip\n");
                return NF_ACCEPT;
            }
            if (isAddressEqual(ipHeader,1,10,0,2,4) == false){
                printk(KERN_INFO "Filter 2: incorrect target dst ip\n");
                return NF_ACCEPT;
            }
            printk(KERN_INFO "Filter 2: B telnet A\n");
            return NF_DROP;
        }
    }

    if (ipHeader->protocol == 1) { // ICMP
        icmpHeader = (struct icmphdr *)((__u32 *)ipHeader + ipHeader->ihl);
        // Filter 5: B ping A
        if (icmpHeader->type == 8){ // ping
            if (isAddressEqual(ipHeader,0,10,0,2,7) == false){
                printk(KERN_INFO "Filter 5: incorrect target src ip\n");
                return NF_ACCEPT;
            }
            if (isAddressEqual(ipHeader,1,10,0,2,4) == false){
                printk(KERN_INFO "Filter 5: incorrect target dst ip\n");
                return NF_ACCEPT;
            }
            printk(KERN_INFO "Filter 5: B ping A\n");
            return NF_DROP;
        }
    }
    return NF_ACCEPT;
}
unsigned int hook_func_out(void *priv, struct sk_buff *skb, const struct
nf_book_state *state){
    ipHeader = (struct iphdr *)skb_network_header(skb);
    if (ipHeader->protocol == 6) { // TCP
        tcpHeader = (struct tcphdr *)((__u32 *)ipHeader + ipHeader->ihl);
        unsigned int dst = (unsigned int)ntohs(tcpHeader->dest);

        // Filter 1: A telnet B
        if (dst == 23) { // telnet
            // check ip addr
            if (isAddressEqual(ipHeader,0,10,0,2,4) == false){
```

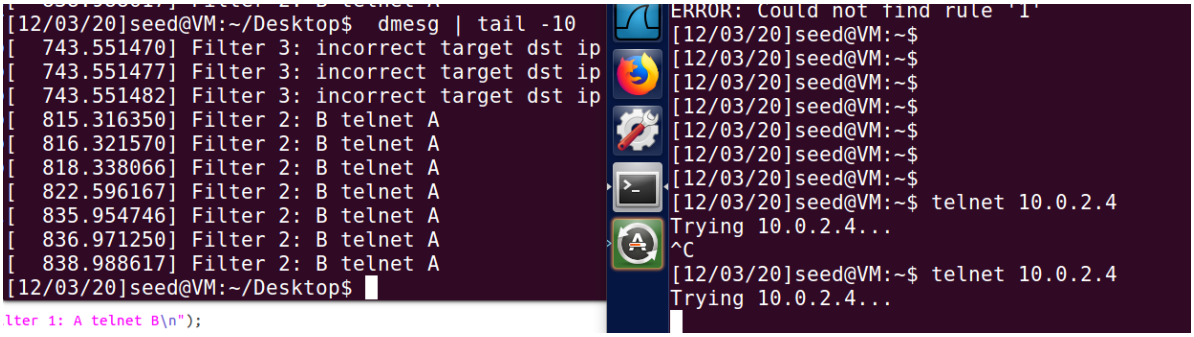
```
                printk(KERN_INFO "Filter 1: incorrect target src ip\n");
                return NF_ACCEPT;
            }
            if (isAddressEqual(ipHeader,1,10,0,2,7) == false){
                printk(KERN_INFO "Filter 1: incorrect target dst ip\n");
                return NF_ACCEPT;
            }
            printk(KERN_INFO "Filter 1: A telnet B\n");
            return NF_DROP;
        }

        // Filter 3: A http www.fudan.edu.cn
        if (dst == 80) { // http
            // check ip addr
            if (isAddressEqual(ipHeader,0,10,0,2,4) == false){
                printk(KERN_INFO "Filter 3: incorrect target src ip\n");
                return NF_ACCEPT;
            }
            if (isAddressEqual(ipHeader,1,202,120,224,81) == false){
                printk(KERN_INFO "Filter 3: incorrect target dst ip\n");
                return NF_ACCEPT;
            }
            printk(KERN_INFO "Filter 3: A http www.fudan.edu.cn\n");
            return NF_DROP;
        }
    }

    if (ipHeader->protocol == 1) { // ICMP
        icmpHeader = (struct icmphdr *)((__u32 *)ipHeader + ipHeader->ihl);
        // Filter 4: A ping B
        if (icmpHeader->type == 8){ // ping
            if (isAddressEqual(ipHeader,0,10,0,2,4) == false){
                printk(KERN_INFO "Filter 4: incorrect target src ip\n");
                return NF_ACCEPT;
            }
            if (isAddressEqual(ipHeader,1,10,0,2,7) == false){
                printk(KERN_INFO "Filter 4: incorrect target dst ip\n");
                return NF_ACCEPT;
            }
            printk(KERN_INFO "Filter 4: A ping B\n");
            return NF_DROP;
        }
}
    return NF_ACCEPT;
}

int init_module(){
    nfho_in.hook = (void *)hook_func_in;
    nfho_in.hooknum = NF_INET_PRE_ROUTING; // 收到的数据包
    nfho_in.pf = PF_INET;
    nfho_in.priority = NF_IP_PRI_FIRST;

    nf_register_hook(&nfho_in);

    nfho_out.hook = (void *)hook_func_out;
    nfho_out.hooknum = NF_INET_POST_ROUTING; // 转发的或者是本地发出的数据包
    nfho_out.pf = PF_INET;
    nfho_out.priority = NF_IP_PRI_FIRST;
```

```
    nf_register_hook(&nfho_out);
    printk(KERN_INFO "Welcome~\n");
    return 0;
}

void cleanup_module(){
    printk(KERN_INFO "See u~\n");
    nf_unregister_hook(&nfho_in);
    nf_unregister_hook(&nfho_out);
}
```

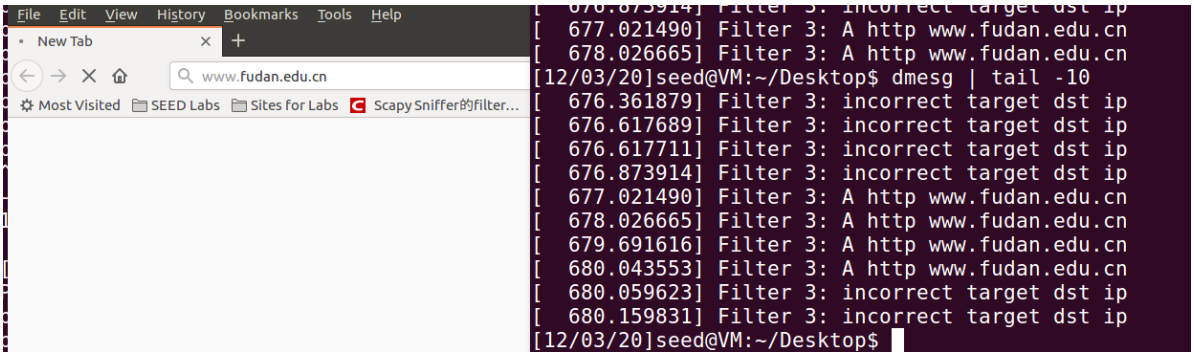
Filter 1: A telnet B



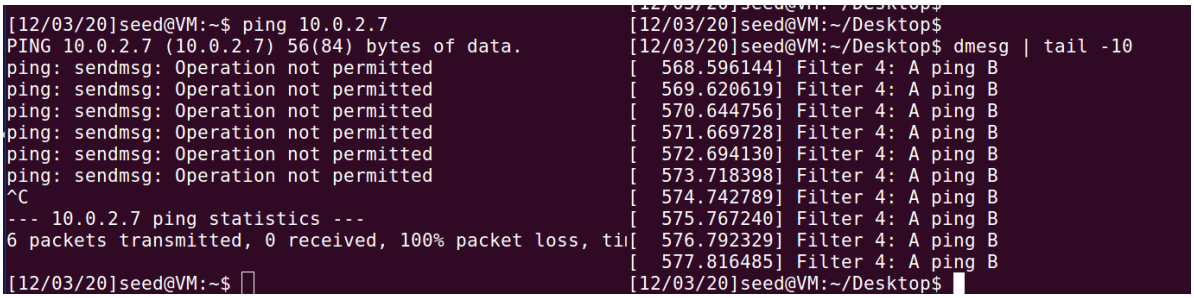
Filter 2: B telnet A



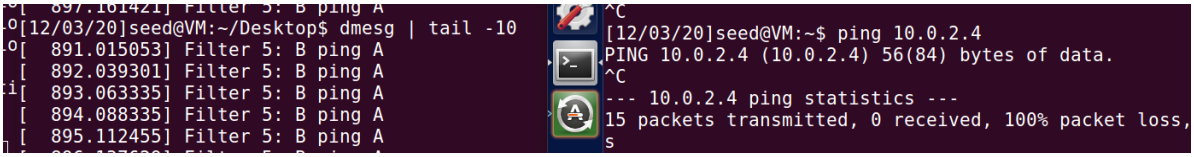
Filter 3: A http www.fudan.edu.cn



Filter 4: A ping B



Filter 5: B ping A



Task 3: Evading Egress Filtering

由于www.facebook.com由于一些众所周知的原因已经被block了，所以还是使用www.fudan.edu.cn代替

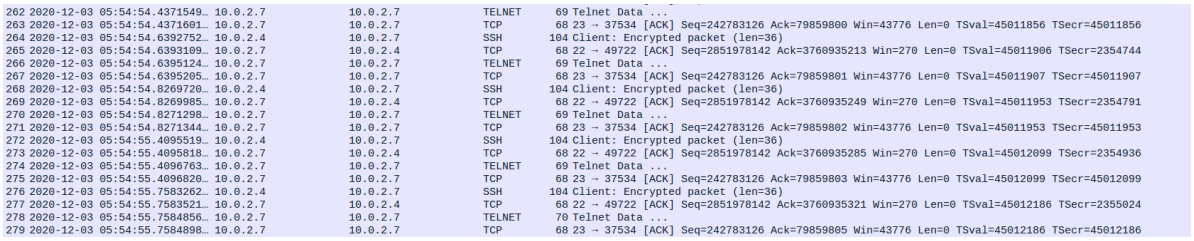
```
# Block all the outgoing traffic to external telnet servers
$ sudo ufw deny out from 10.0.2.4 to any port 23
# Block all the outgoing traffic to www.fudan.edu.cn
$ sudo ufw deny out from 10.0.2.4 to 202.120.224.81
```

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

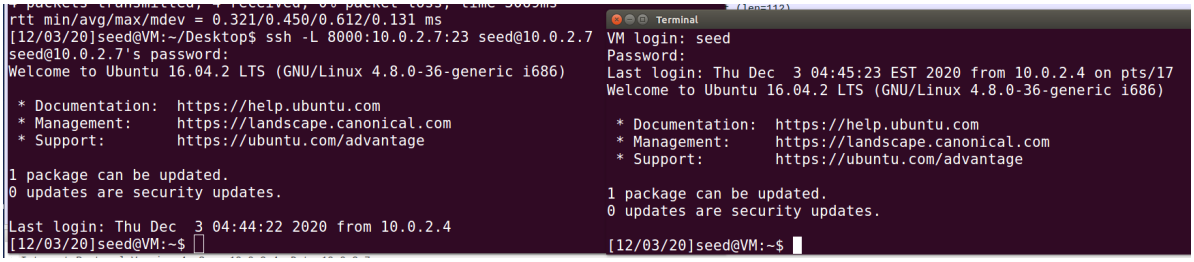
这一步需要先关闭10.0.2.7的防火墙

```
$ sudo ufw disable
```

```
$ ssh -L 8000:10.0.2.7:23 seed@10.0.2.7
$ telnet localhost 8000
```

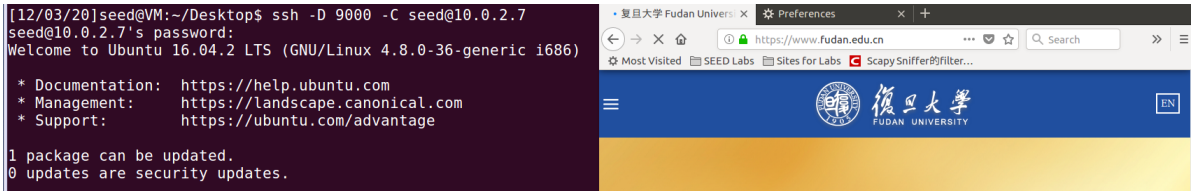


在Wireshark中抓到的数据包可以看到ssh连接是作为真正的telnet连接（10.0.2.4<-->10.0.2.7）的桥梁（10.0.2.4<-->10.0.2.7(acting as ssh server, "apollo")<-->10.0.2.7）

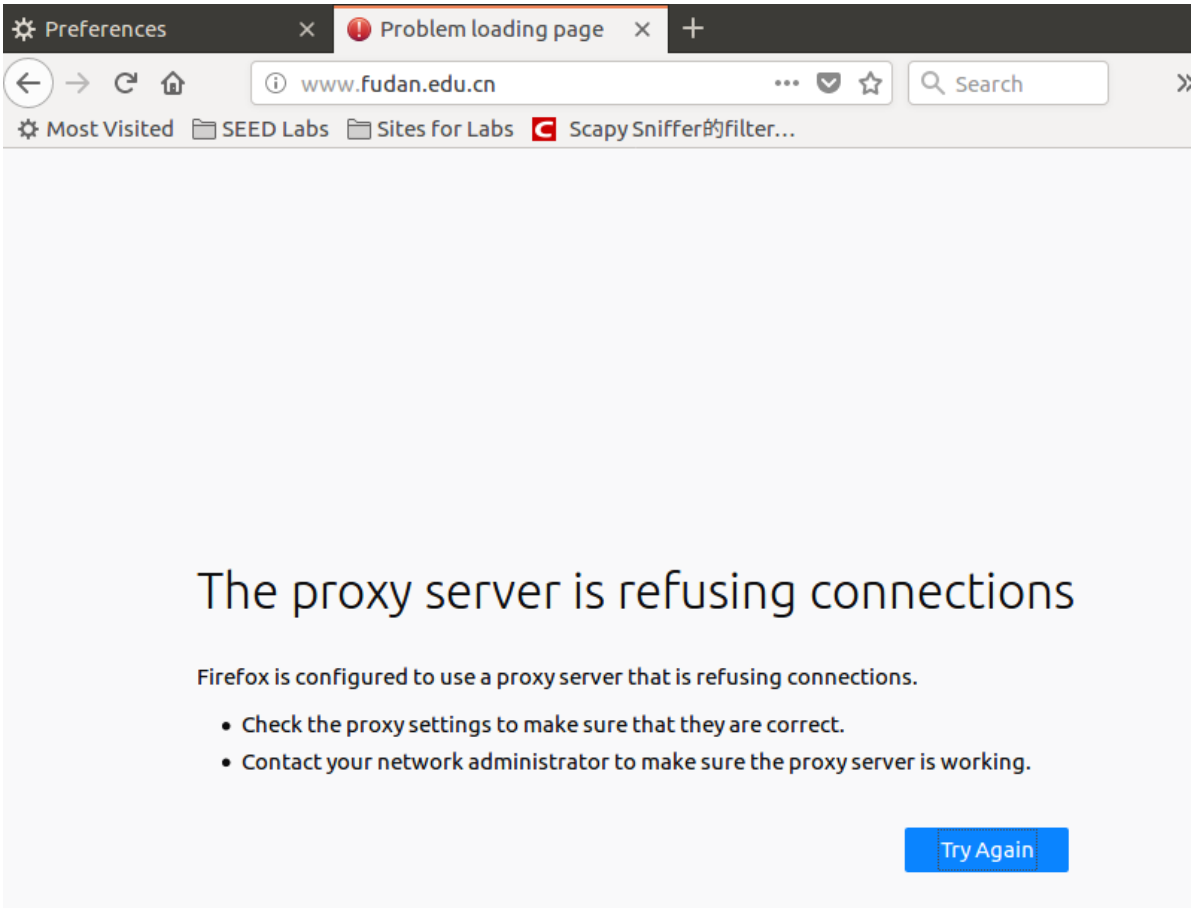


Task 3.b: Connect to Facebook using SSH Tunnel

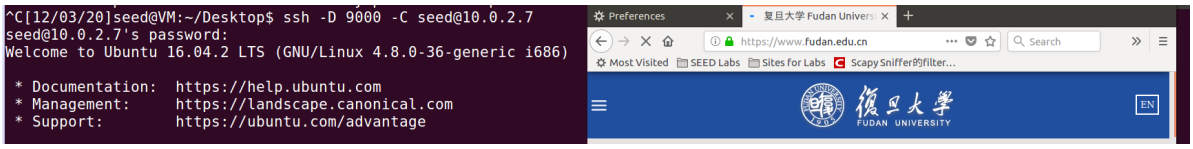
```
$ ssh -D 9000 -C seed@10.0.2.7
# -D 绑定端口
# -C 请求压缩所有数据
```



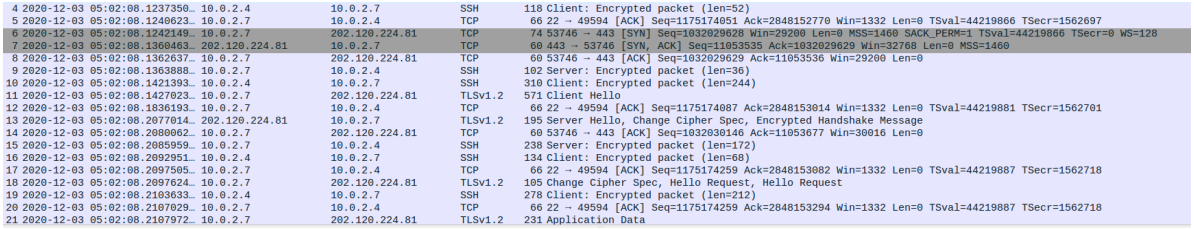
断开ssh连接并清空浏览器的cache之后：



再次建立ssh连接之后刷新浏览器页面：



在这个过程中10.0.2.7充当了proxy的角色。实际上是由10.0.2.7向www.fudan.edu.cn发送请求并将收到的响应通过建立的ssh连接发给10.0.2.4。



Task 4: Evading Ingress Filtering

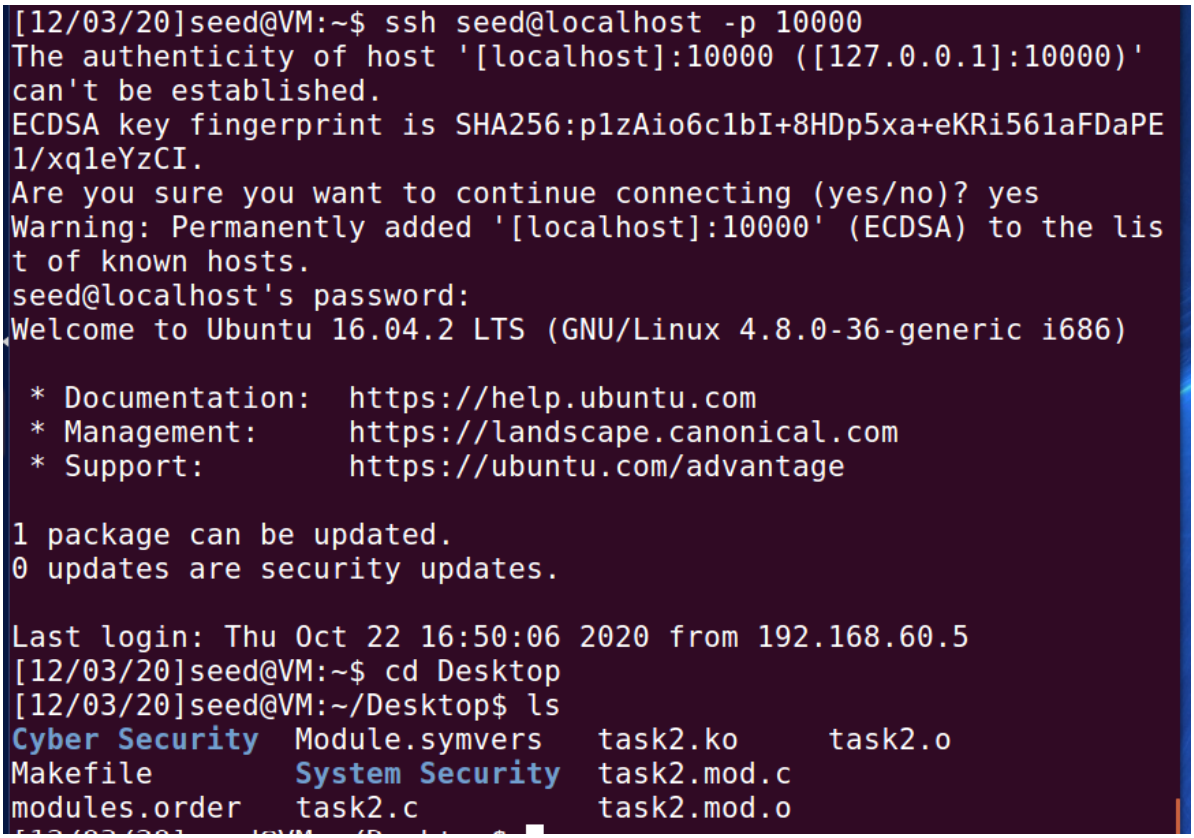
```
# 删除之前的task设置的防火墙规则
$ sudo ufw delete 1
# block Machine B from accessing its port 80 (web server) and 22 (SSH server)
$ sudo ufw deny out from 10.0.2.7 to 10.0.2.4 port 80
$ sudo ufw deny out from 10.0.2.7 to 10.0.2.4 port 22
```

首先在10.0.2.4上开启反向连接隧道[2]：

```
$ ssh -f -N -R 10000:localhost:22 seed@10.0.2.7
# -f 后台执行ssh指令
# -N 不执行远程指令
# -R listen-port:host:port 指派远程上的 port 到本地地址上的 port
```

然后在10.0.2.7上通过10000端口就可以成功建立ssh连接：

```
$ ssh seed@localhost -p 10000
```



1. https://www.cnblogs.com/EasonJim/p/6851241.html ↩
2. https://www.cnblogs.com/x_wukong/p/5997872.html ↩