**实 验 报 告**

**实验4-虚拟专用网与反向代理技术**

|  |  |  |
| --- | --- | --- |
| **课程名称：** | **信息系统安全** | |
| **小组成员** | **姓名** | **学号** |
|  | **李林键** | **201630406969** |
|  | **刘一达** | **201630665229** |
|  | **吴峻羽** | **201630665892** |
|  | **叶苏航** | **201630666202** |
| **学生专业：** | **软件工程** | |
| **开课学期：** | **2018-2019 第二学期** | |

**软件学院**

**2019年3月**

**实验4 - 虚拟专用网与反向代理技术**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **地 点：** | B7 楼 | 333 房； | **实验台号：** |  |
| **实验日期与时间：** | 第9周 周四晚 | | **评 分：** |  |
| **预习检查纪录：** | 良好 | | **实验教师：** | 陈春华 |

1. 实验过程与结果

合理说明实验过程与结果

在实验3中我们已经签发了vpnserver的证书与密钥，`server.crt` 和 `server.key`。我们组有4个人，因此我们使用easy-rsa签发4个客户端的证书。

```

./easyrsa gen-req llj nopass # Common Name = llj

./easyrsa gen-req wjy nopass # Common Name = wjy

./easyrsa gen-req lyd nopass # Common Name = lyd

./easyrsa gen-req ysh nopass # Common Name = ysh

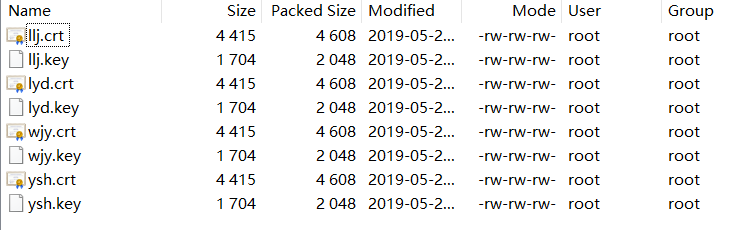
./easyrsa sign-req client llj

./easyrsa sign-req client wjy

./easyrsa sign-req client lyd

./easyrsa sign-req client ysh

```



然后把 `crt` 和 `key` 分配到对应小组成员的设备上。同时，还要把 `ca.crt` 也分配给每个小组成员的设备。

现在开始配置 OpenVPN 的服务器。

**注意：我尝试过用 Ubuntu 16.04/18.04 安装并配置，不能成功。尝试过用 CentOS 7 执行 `yum install openvpn` 安装并配置，也没能成功。唯一的成功尝试是使用 CentOS 从 OpenVPN 的源码使用 `rpmbuild` 编译成 rpm 包，然后以 rpm 包的方式进行安装。**

安装过程参考 [*https://cloud.tencent.com/developer/article/1026294*]

安装过程可以参考，前面的腾讯云链接，由于阿里云的源没有 OpenVPN 2.4.7，因此需要自己下载好源码然后上传到服务器。构建 CA 、签发证书等过程在实验 3 已经完成了，因此这些过程不需要参考前面的腾讯云链接。

在 CentOS 7 系统中，执行以下指令

`

wget -O /etc/yum.repos.d/CentOS-Base.repo \

http://mirrors.aliyun.com/repo/Centos-7.repo

yum update

yum clean all

yum makecache

yum install -y lzo lzo-devel openssl openssl-devel pam pam-devel

wget http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm

rpm -ivh epel-release-latest-7.noarch.rpm

yum install -y pkcs11-helper pkcs11-helper-devel

rpm -qa lzo lzo-devel openssl openssl-devel \

pam pam-devel pkcs11-helper pkcs11-helper-devel

# Upload the OpenVPN 2.4.7 source to the server.

yum install rpm-build

rpmbuild -tb openvpn-2.4.7.tar.gz

ls /root/rpmbuild/RPMS/x86\_64/

rpm -ivh /root/rpmbuild/RPMS/x86\_64/openvpn-2.4.7-1.x86\_64.rpm

# The processes of building PKI, building CA,

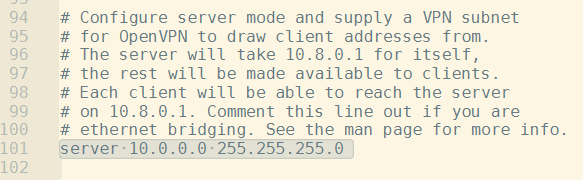
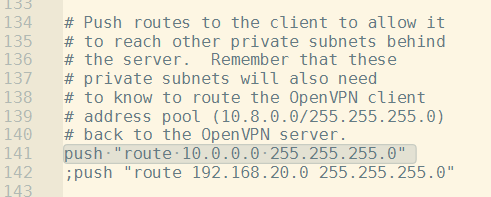
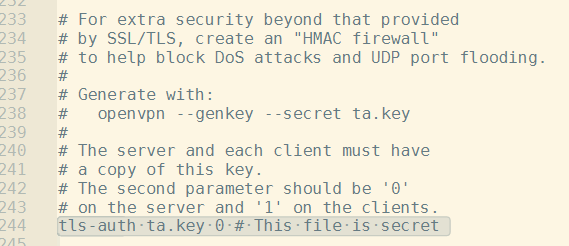
# generating/signing request and generating DH parameter have

# been done in Lab-3. I just need to copy the files in proper directories.

`

然后要把之前生成的服务器的公钥私钥、CA的公钥、Diffie-Hellman key、ta.key都复制到OpenVPN的安装目录下。

接下来把编辑好的服务端配置文件 `server.conf` 复制到 `/etc/openvpn/` 目录下

注意，`server.conf` 把通信端口设置为 1194，因此需要在阿里云控制台开启 1194 端口。



在 `server.conf` 文件中，指定了客户端配置目录

`

client-config-dir ccd # 新加入的客户端节点按ccd中对应文件中的规则进行ip分配

`

因此我们需要在 `/etc/openvpn/` 目录下新建一个 `ccd` 目录，并在 `ccd` 目录中编辑对应的配置文件。配置文件的文件名为客户的证书的Common Name。因此在本次实验中，有4个配置文件，分别对应小组4个成员。

按照如下规则设置ip

`

pwd

# output: /etc/openvpn/ccd

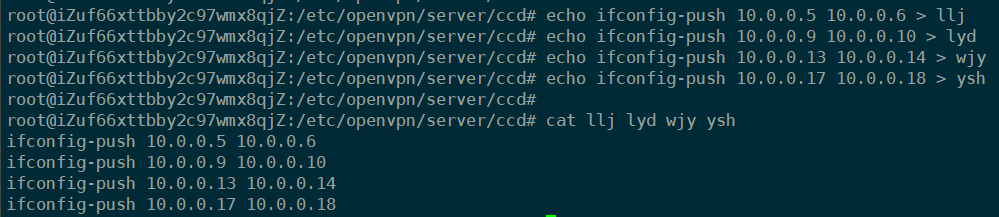
echo ifconfig-push 10.0.0.5 10.0.0.6 > llj

echo ifconfig-push 10.0.0.9 10.0.0.10 > lyd

echo ifconfig-push 10.0.0.13 10.0.0.14 > wjy

echo ifconfig-push 10.0.0.17 10.0.0.18 > ysh

`



然后在服务器设置iptables。

```

vim /etc/sysctl.conf # 打开文件并编辑

net.ipv4.ip\_forward = 1 # 添加这一行内容在文件中

:wq # 保存并退出

sysctl -p # 使内核参数生效

# 执行以下三行指令，添加iptables规则,确保服务器可以转发数据包到外网:

iptables -t nat -A POSTROUTING -s 10.0.0.0/24 -j MASQUERADE

service iptables save

iptables -t nat -L

# 启动 OpenVPN 并设置自启动

sudo systemctl start openvpn@server

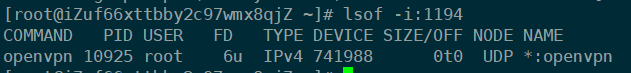
sudo systemctl enable openvpn@server

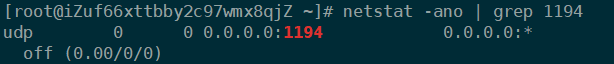
# 查看 1194 端口监听情况, 查看 OpenVPN

lsof -i:1194

netstat -ano | grep 1194

```





服务端配置到此就完成了，接下来就是配置客户端。在客户端系统（我使用Windows 10系统）安装OpenVPN之后，把用户的公钥、私钥、CA的公钥、ta.key复制安装目录下的 `config` 目录下。并且，在 `config` 目录下还要新建一个 `config.ovpn` 文件，打开并写入以下内容

```

client # 声明本程序属于客户端

dev tun # 使用TUN模式

proto udp # 使用udp协议

remote 47.100.57.125 1194 # 服务器公网IP

resolv-retry infinite # 不断重试vpnserver的域名

nobind # 客户端程序使用随机端口

persist-key # 保持允许状态，重启时更快

persist-tun

ca ca.crt # CA公钥证书

cert llj.crt # 本客户端使用的入网凭证，该证书的CN属性为llj，这里填写自己拿到的文件

key llj.key # 对应的私钥

tls-auth ta.key 1

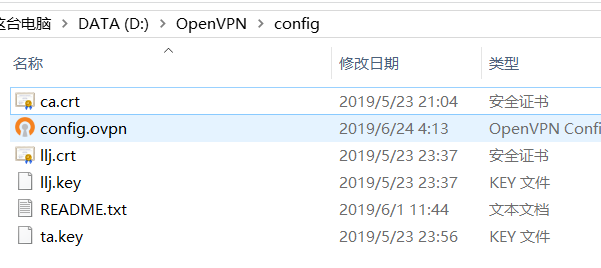
remote-cert-tls server # 入网时进行SSL/TLS过程，客户端认证服务器证书

cipher AES-256-CBC # 使用与vpnserver一样的对称加密算法

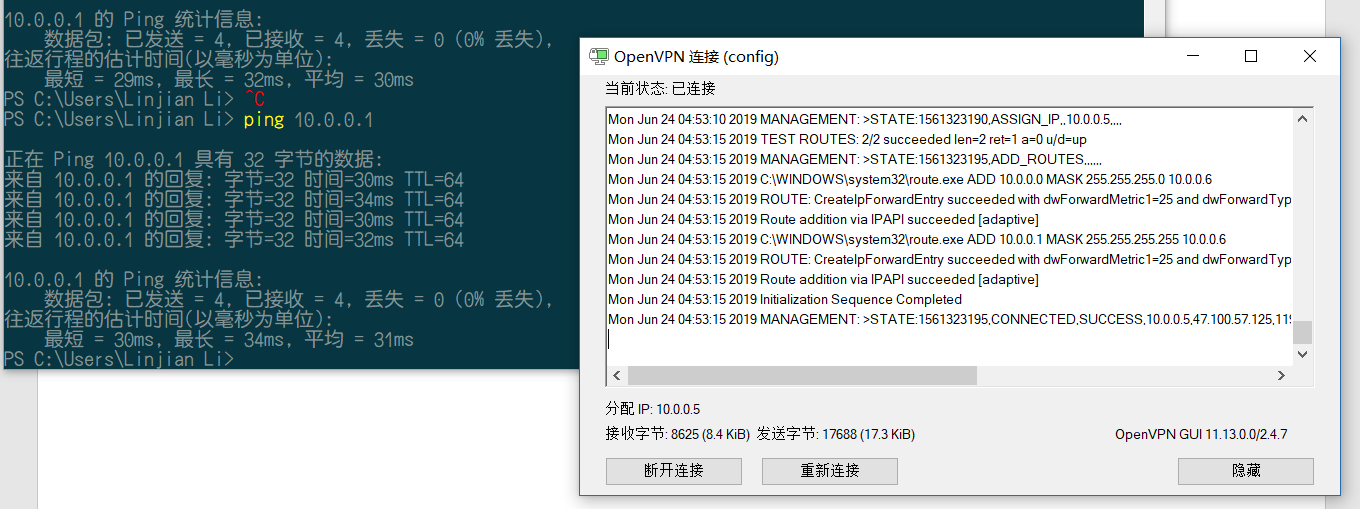
verb 3

```

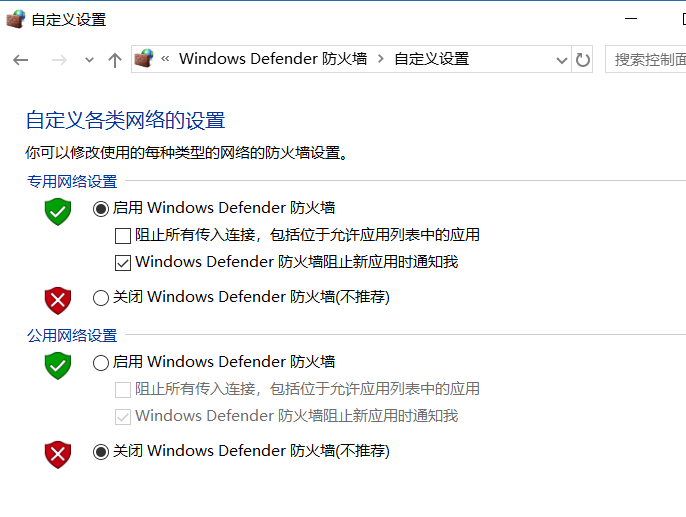
注意上面的第9、10、11行，对应的参数值应该填写用户自己所拿到的文件的名字。我构建CA时公钥名字就设为 `ca.crt`，并且我自己的公钥和私钥名字为 `llj.crt` 和 `llj.key`，因此在上面的内容我就写了这三个文件名。

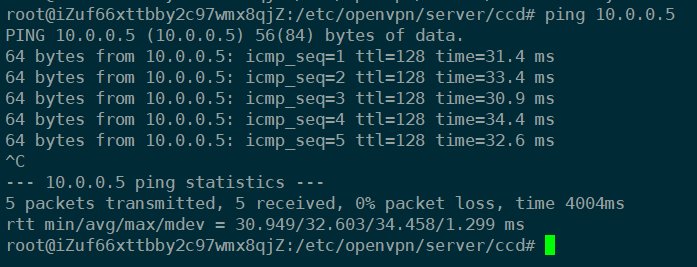


客户端配置好之后，**以管理员身份**运行OpenVPN，然后连接就能加入VPN网络了。我的身份信息是 `llj`，因此按照前面设置的服务器的配置，我得到的子网ip为 10.0.0.5。等其他组员也加入之后，就能够顺利ping到他们的子网地址。

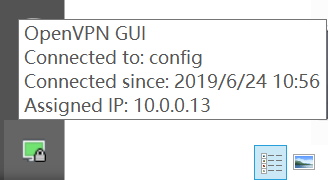


从客户端ping服务器可以成功，但是从服务器ping客户端是失败的。上网查找后发现是因为Windows客户端打开了防火墙。把防火墙关掉之后，服务器就能够ping到客户端了

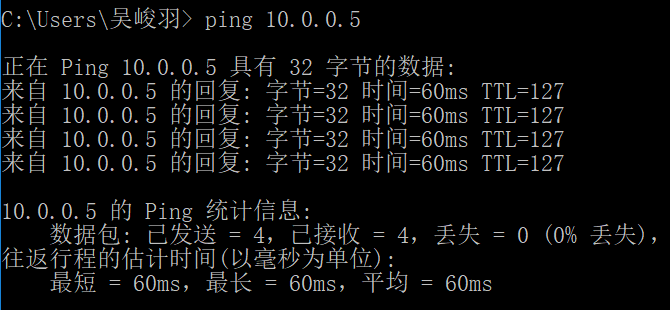
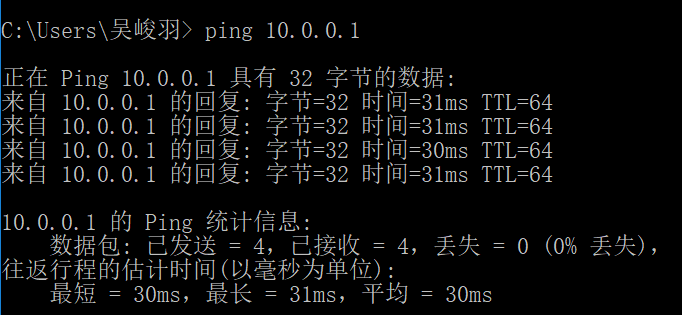
 



现在尝试从客户端ping客户端。在吴峻羽同学的设备上接入OpenVPN，按照前面的分配规则，他应该拿到10.0.0.13这个IP。

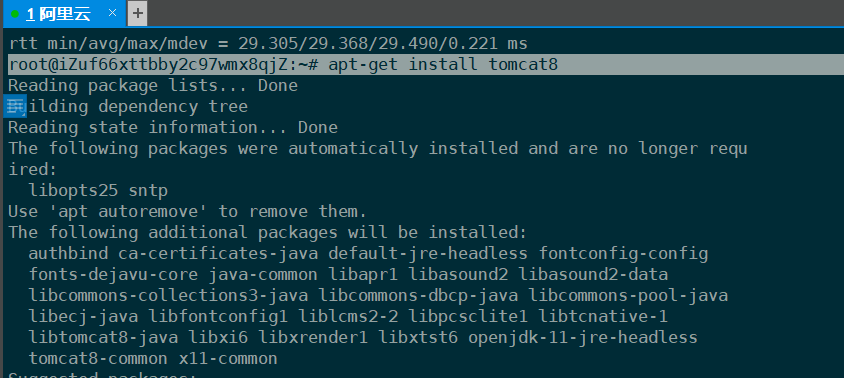


现在ping李林键的设备，IP为10.0.0.5，结果如下。

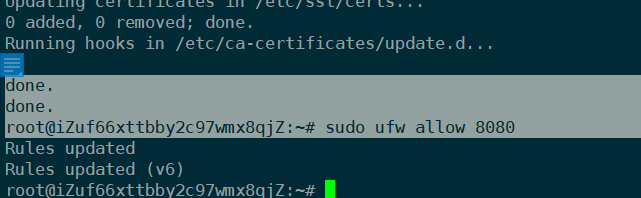
 

接下来进行反向代理的实验内容。

首先在服务器上安装Tomcat。由于我只有一台服务器，因此就只能在之前的OpenVPN服务器上安装了。



然后更新防火墙规则

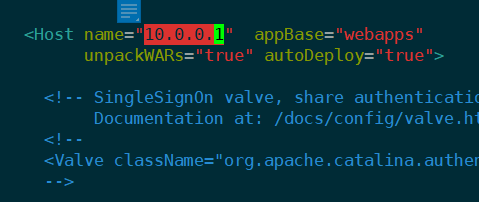


同时也要在阿里云的控制台上允许8080端口的通信。



然后修改Tomcat的服务器配置，使服务器的IP为10.0.0.1，端口不变，仍为默认的8080。





这里原本是

name=localhost

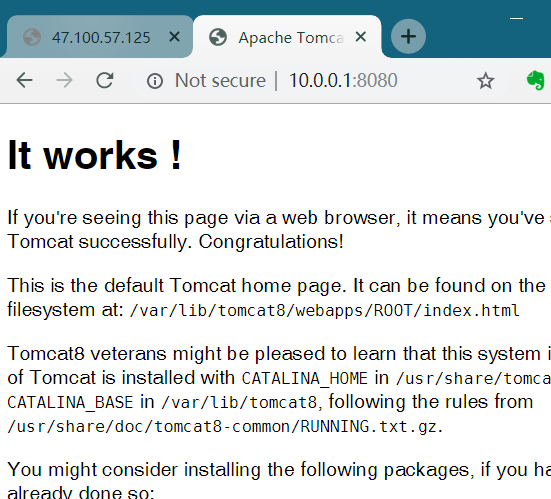
改为

name=10.0.0.1

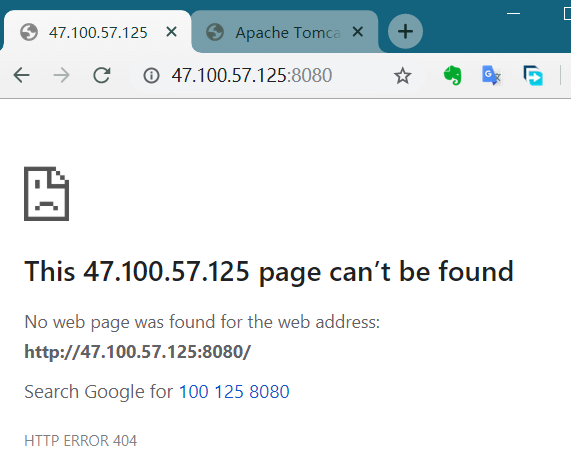
然后

sudo service tomcat8 restart

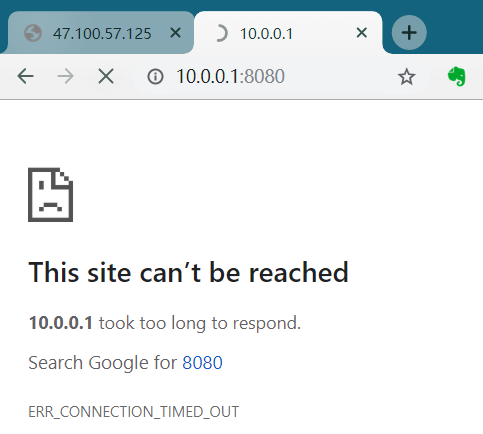
配置好之后，在客户端上访问10.0.0.1:8080，即访问Tomcat服务器的内网地址。



然后从公网IP访问Tomcat服务器是不行的（因为现在还没有设置反向代理）。



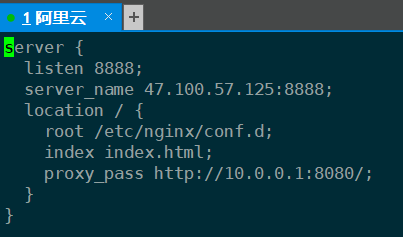
因为我现在的客户端连接着VPN，因此可以用内网地址访问。断开后就无法访问了，如下图。



现在我们配置nginx的反向代理。安装了nginx之后，在conf.d目录下加入47.100.57.125.conf文件（前面的IP地址是自己服务器的公网IP）。

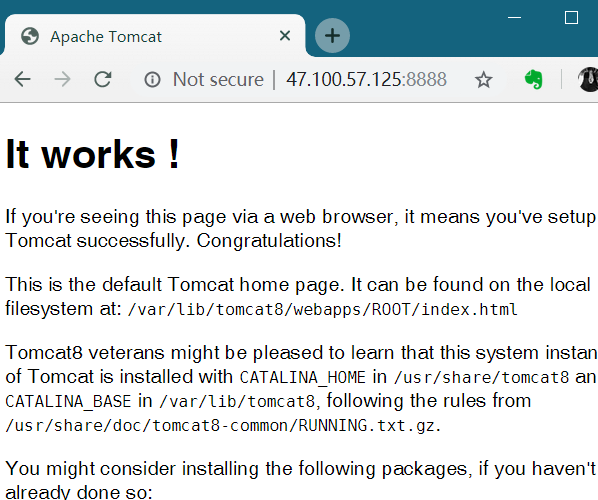


然后写入以下内容





这时候我们重新使用公网的IP和端口8888访问



发现可以访问Tomcat服务器了。说明nginx已经把我的访问反向代理到了10.0.0.1:8080。

1. 回答如下问题
2. 说明实验中VPN服务器的具体作用

答：作为中转站。

1. 收到客户端发来的（经过OpenVPN客户端程序封装的）IP数据包

2. 解封数据包，识别数据包目的地（VPN地址）

3. 通过查询VPN地址与公网地址的对应表格

4. 重新封装，使用目的地的公网IP把数据包发送出去

1. 说明实验中Nginx服务器具体作用

答：

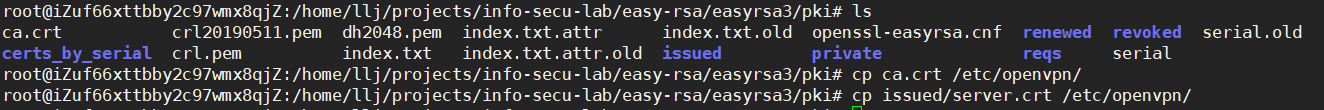
收到访问请求的时候，在用户不知情的情况下，把访问请求转发到内部服务器上。

Tomcat服务器本来设置的只允许内网访问，但用户需要从外网进行访问，就需要反向代理。我们在没有连接OpenVPN的时候，是没有办法访问10.0.0.1:8080的，但是通过反向代理，当用户在访问<http://47.100.57.125:8888/>这个服务的时候，Nginx把请求转发到内网Tomcat服务器10.0.0.1:8080上，但用户并不知道自己的请求被转发了，仍认为自己的请求是被<http://47.100.57.125:8888/>处理的，实质是被10.0.0.1:8080处理了。

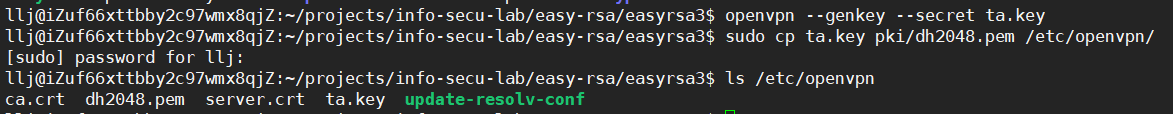
# 附录：之前尝试过的所有失败的方法

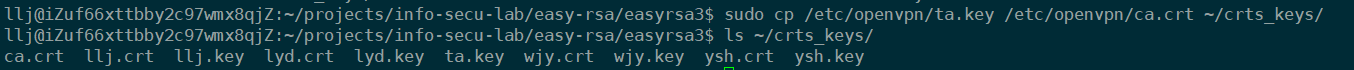
使用 `apt-get install openvpn` 在服务器上安装OpenVPN，然后把CA和vpnserver的证书都复制到openvpn的目录下。



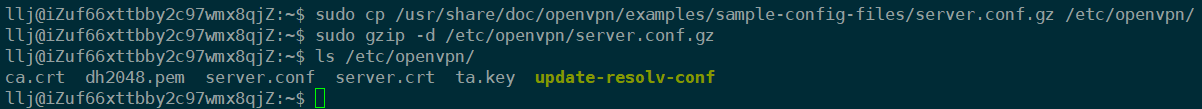


生成一个HMAC签名，然后把HMAC签名和之前生成的Diffie-Hellman key也复制到openvpn目录下。

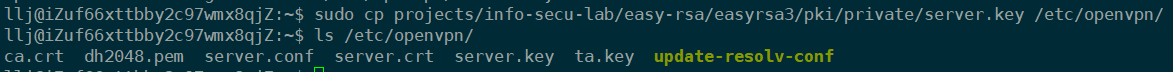




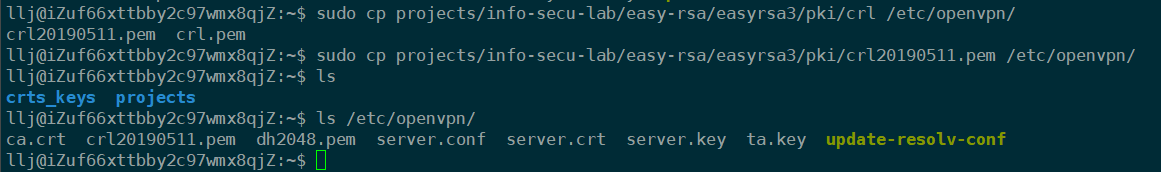
接下来需要配置 OpenVPN 服务器。



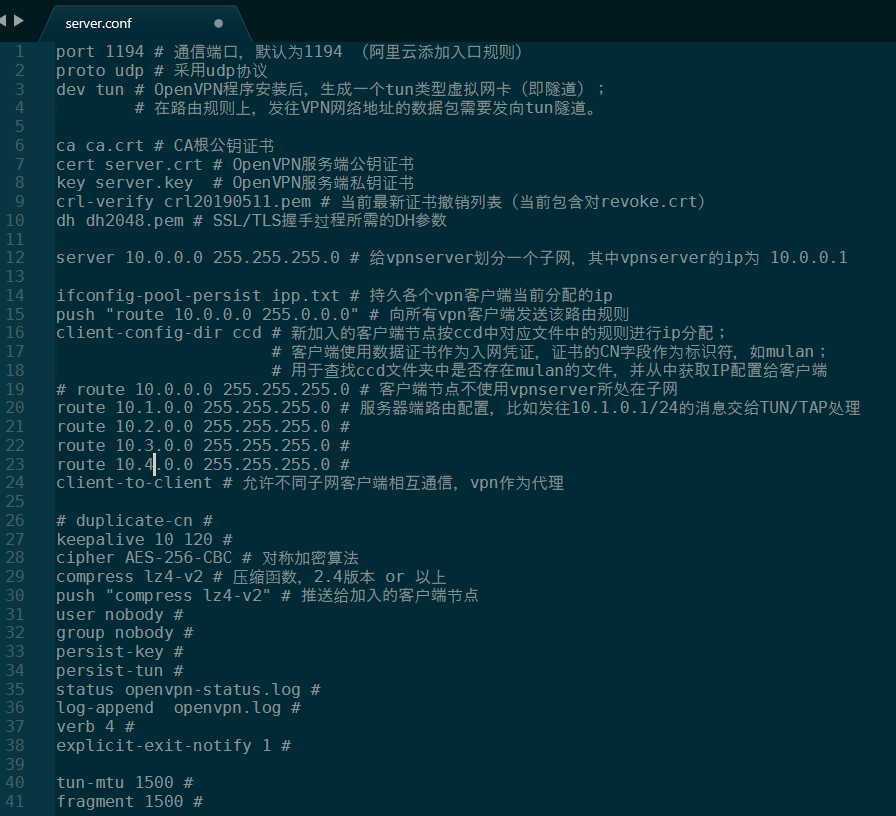
前面忘记复制 server.key 到openvpn的目录下，现在补回来。

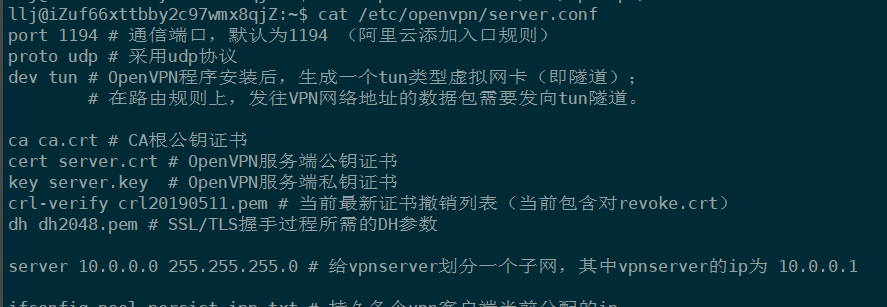


同时，也要复制一份CRL到openvpn的目录



按照实验指南与我之前操作的实际情况，对 `server.conf` 做出修改。





我们的VPN共分为5个subnet，分配方案如下

然后输入以下指令启动OpenVPN服务

`

sudo systemctl start openvpn@server.service

`

我在输入指令后，无法启动服务。出现了如下错误消息

`

Job for openvpn@server.service failed because the control process exited with error code. See "systemctl status openvpn@server.service" and "journalctl -xe" for details.

`

于是我输入

`

sudo systemctl status openvpn@server.service

`

得到如下内容

`

**Options error: Unrecognized option or missing parameter(s) in /etc/openvpn/server.conf:4:   (2.3.10)**

`

我认为可能是前面的截图中，我的 `server.conf` 文件中第4、17、18行的开头有许多空格，导致软件读取配置文件的时候出了错。于是我把这些行开头的空格删除。

再尝试启动OpenVPN服务，这时还是有错误，查看状态得到以下消息

`

Options error: Unrecognized option or missing parameter(s) in /etc/openvpn/server.conf:29: compress (

`

这个在实验材料里最下面有说明，OpenVPN从2.3升级到2.4存在压缩算法兼容的问题。

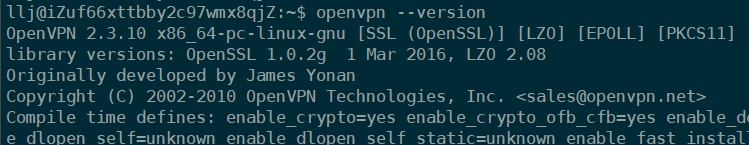
在 `server.conf` 里有两行

`

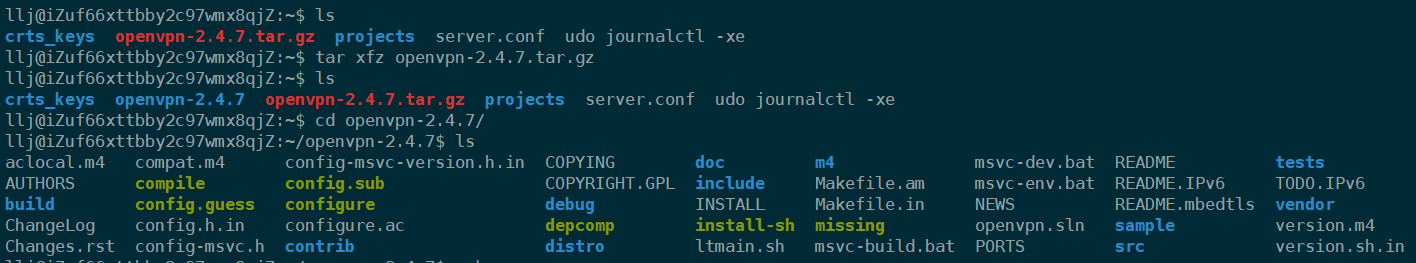
compress lz4-v2 # 压缩函数，2.4版本 or 以上

push "compress lz4-v2" # 推送给加入的客户端节点

`



可以看到，我的服务器的OpenVPN版本为2.3.10



进入到openvpn-2.4.7的目录在执行

`

./config

`

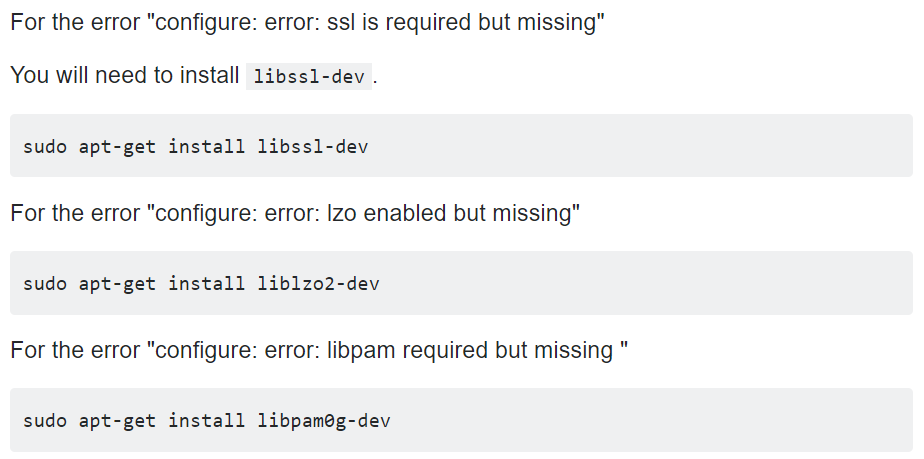
的时候，出现了

`

configure: error: openssl check failed

`

上网搜索，发现了如下解答



于是

`

apt-get install libssl-dev

`

再执行

`

./config

`

的时候出现了

`

configure: error: lzo enabled but missing

`

于是执行

`

sudo apt-get install liblzo2-dev

sudo apt-get install libpam0g-dev

`

接下来再执行

`

./configure

`

就能够通过了。

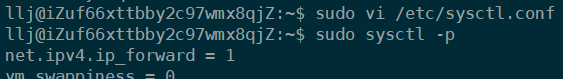
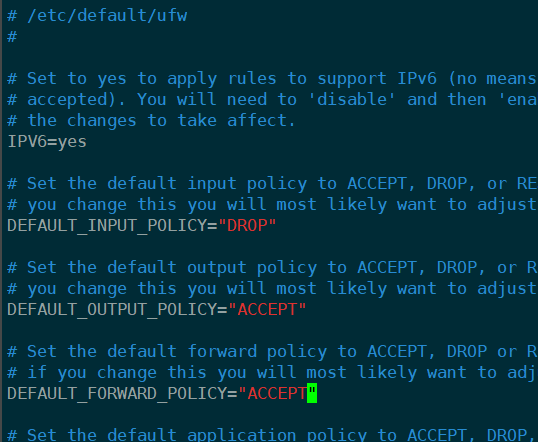
然后

`

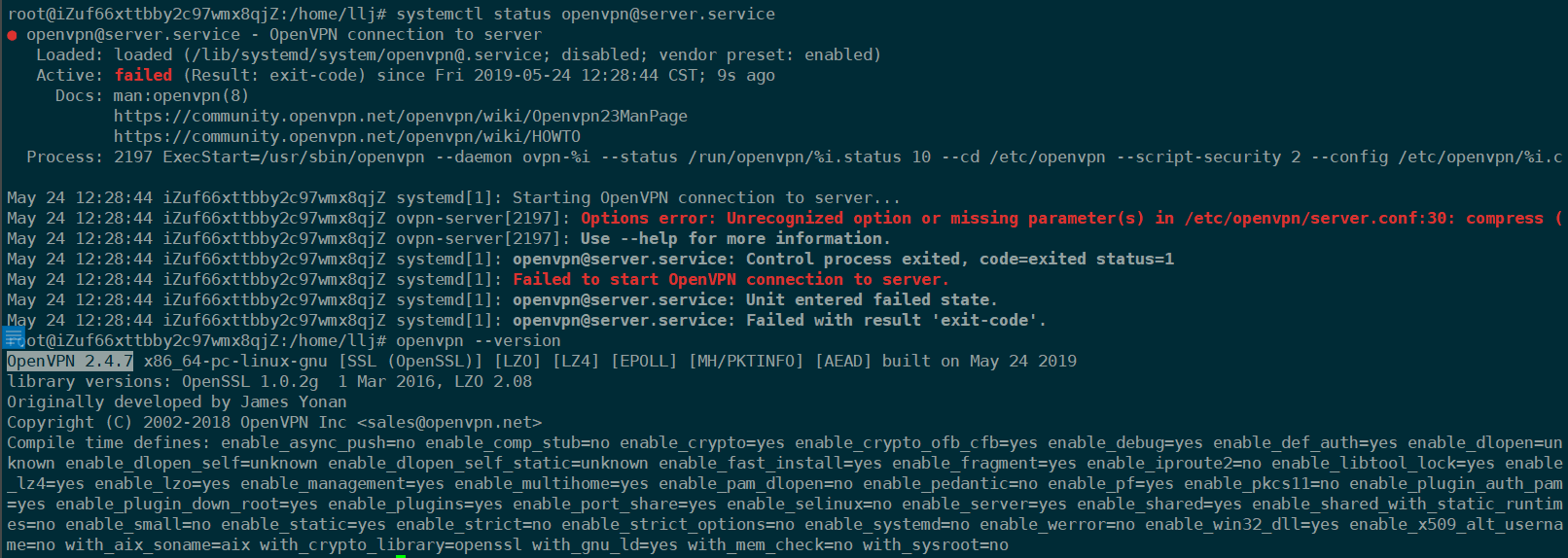
make

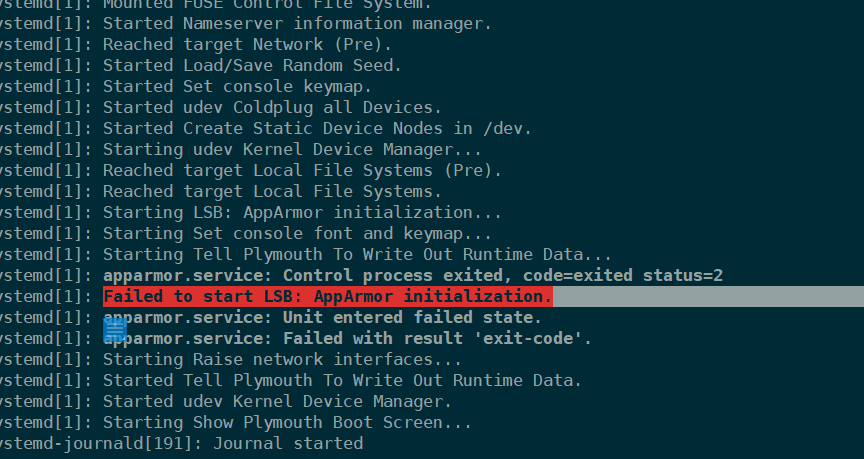
make install

`

想办法更新到了 OpenVPN 2.4.7 ，依然没能解决问题。



重新安装了 Ubuntu 18.04，然后安装OpenVPN，版本确认为2.4.4。但是启动服务的时候又不能成功，错误信息有

`

error: Could not load host key: /etc/ssh/ssh\_host\_ed25519\_key

pam\_systemd(sshd:session): Failed to release session: Interrupted system call

`