



GitCode 开源社区 文章已被社区收录

加入社区



linux 同时被 2 个专栏收录 ▾

1 订阅 28 篇文章

订阅专栏

最近研究了一段时间的openvpn组网技术，也试着搭建了一个openvpn环境，大概理解了其中使用的一些技术原理，还是记录一下。本篇文章对专业搞网络的人也许用处不大，但是对于初次接触这些技术（比如vpn，代理技术，加密隧道，防火墙，路由，局域网组网）的人还是有一定价值的，便于理清整个vpn组网技术的脉络，也可以在遇到问题的时候自己排查。

openvpn是众多vpn种类的一种，是一个开源的产品，也是应用最广泛的一种vpn。支持的平台很多，我们常用的系统平台linux，window，安卓都支持。我搭建的openvpn服务端是运行在centos上，客户端是运行在安卓手机上的。其实不管运行在哪个平台，配置基本都是一样的。openvpn还有一个重要的功能是可以使用UDP协议传输数据，这相对于使用TCP传输数据的vpn，速度要快很多。

openvpn能构建一个虚拟专用网络（VPN），很多人也许不是很了解这句话包含的内容。一个是“虚拟的”，说的是这个网络内的各个主机在地理位置上，可能不是在相邻的地域，可能在地球的两端，可能横跨了整个互联网。这样看起来，这个网络就不像一个局域网，但是在逻辑上，是真正意义上的一个局域网。”虚拟“其实重点是指数据传输的通道上，这个通道是虚拟的，主机的网卡也是虚拟的，不是真实的网卡。还有一个形容词”专用“，这个就是指能和外部网络隔绝起来，为了不被外部网络影响，数据不被窥伺窃取篡改，VPN网络中两个主机之间的通信数据都会经过加密，这样的加密通信通道我们就称之为加密隧道。并且任何一台主机想要加入这个网络，都需要经过严格的认证授权。

VPN的数据包是在TCP/UDP数据包的基础上做了一层封装，这个封装的数据里就是VPN网络里面的局域网数据包，其中和正常的TCP/UDP数据包一样，也有源ip端口，目的ip端口。当然，这些局域网的数据包信息，只有解密了才能看到，外部即使截获了这个数据包，也解密不了其中的数据。这个数据包（也许是在互联网上）经过了漫长的传输后，到达最终目的地后才会被正确的解密，递交给上层应用。

安装VPN客户端的主机上，当认证授权成功后加入VPN网络，都会新增一个虚拟网卡，作为VPN网络通信的专用网卡，VPN服务器启动成功后也一样虚拟出这个专用网卡。网卡的名称一般为tun0,就像下面的：

```

[root@iZ0xihzpwxl982q3tns80wZ ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.22.95.164 netmask 255.255.240.0 broadcast 172.22.95.255
    inet6 fe80::216:3eff:fe00:c0e6 prefixlen 64 scopeid 0x20<link>
    ether 00:16:3e:00:c0:e6 txqueuelen 1000 (Ethernet)
    RX packets 14782500 bytes 11346524365 (10.5 GiB)
    RX errors 0 dropped 482 overruns 0 frame 0
    TX packets 10312456 bytes 10149267555 (9.4 GiB)
    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 1000 (Local Loopback)
    RX packets 4178173 bytes 7423671931 (6.9 GiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4178173 bytes 7423671931 (6.9 GiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

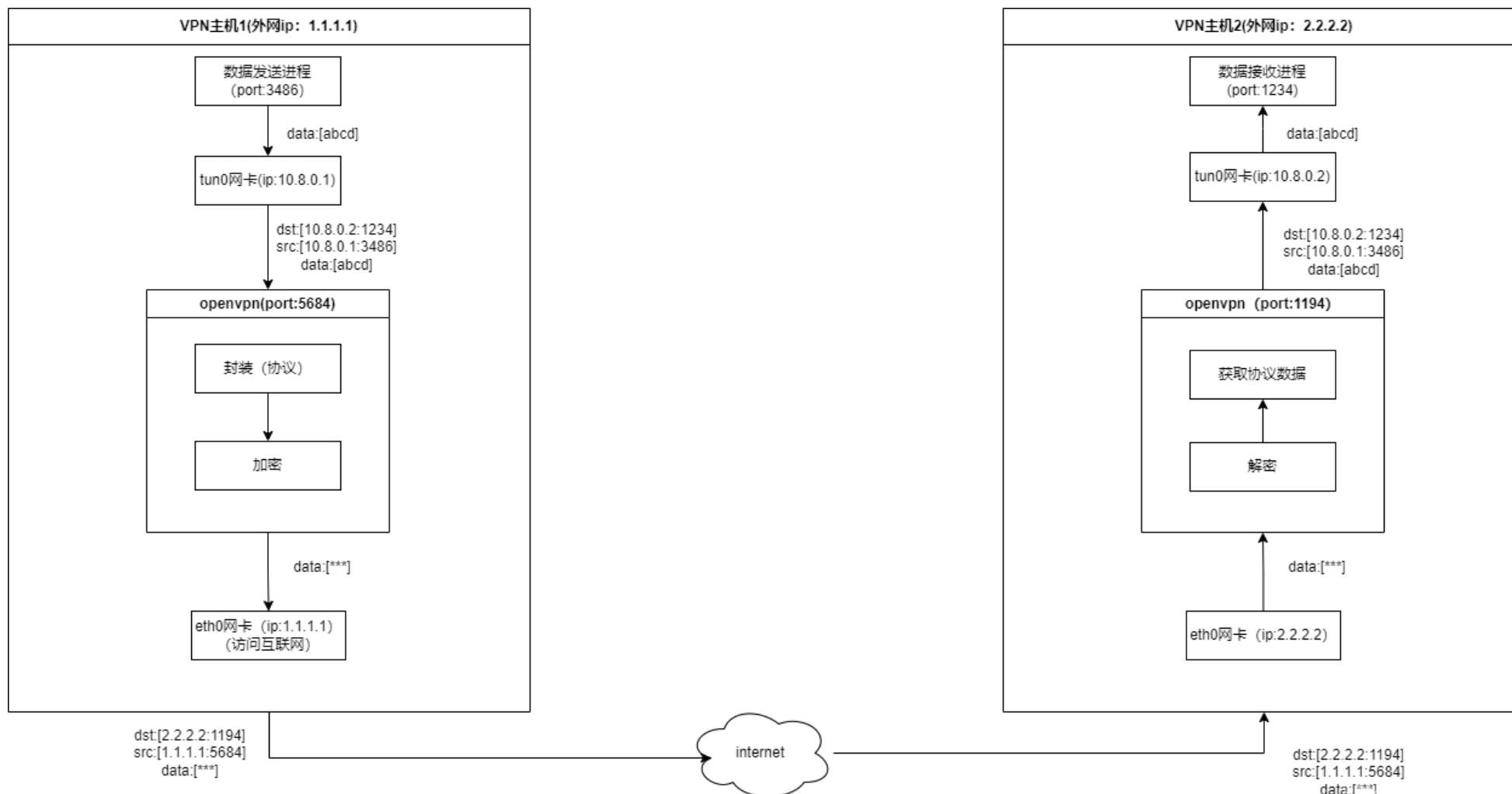
tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet 10.8.0.1 netmask 255.255.255.255 destination 10.8.0.2
    inet6 fe80::2430:224e:2242:324d prefixlen 64 scopeid 0x20<link>
    unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 100 (UNSPEC)
    RX packets 538682 bytes 78445700 (74.8 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1466603 bytes 1723727312 (1.6 GiB)
    TX errors 0 dropped 7595 overruns 0 carrier 0 collisions 0

```

CSDN @hacker_lpy

一、openvpn数据收发工作原理

其中openvpn收发数据处理过程，我大致画了一下，如下图：



上面的VPN主机1就相当于openvpn的客户端，VPN主机2就相当于openvpn的服务器。其中eth0网卡是真实的网卡，对外连接外部网络，tun0是openvpn的虚拟出来的网卡，专门处理openvpn的数据。

数据发送进程，想要给VPN网络中的其他节点发送数据，就使用对端局域网的ip来发送，也就是10.8.0.2这个ip。这个数据发送进程发送的数据，先交给内核的协议栈进行TCP/IP数据包的封装，这个TCP/IP数据包经过本地路由后，本地的路由会直接交给tun0网卡（注意，此处还有一个重要的角色本地路由表），因为tun0网卡是openvpn这个进程虚拟出来的网卡，数据包交给tun0网卡后都会经过openvpn这个进程的处理，openvpn会把收到的数据用自己的openvpn协议进行封装，然后加密成密文（这个加密过程应该是使用开源的ssl或者tls协议），加密完成后就发送给对端（数据接收端）的openvpn进程。至于是怎么知道接收端的openvpn进程的ip端口的，是在开始的时候openvpn客户端和服务端的一些列认证完成后，这个连接才能建立成功的。发送端的openvpn进程把密文数据发出后，也会经过内核的协议栈，进行TCP/IP的协议组包，然后再次经过本地的路由表进行路由，路由表会把这个TCP/IP包交给外网出口网卡eth0发送出去。此时，整个数据的发送过程结束。

VPN主机1从网卡eth0发送的TCP/IP数据包，经过漫长的路由和转发，最终到达目的地，也就是，数据包通过VPN主机2的外网网卡eth0进入，内核协议栈解析这个TCP/IP数据包，知道这个数据包的目的ip端口号(2.2.2.2:1194),于是交给了openvpn进程（openvpn接收进程）处理。openvpn进程对接收到的数据先进行解密，得到openvpn协议数据，然后把VPN主机1数据发送进程发送的数据，转发到tun0网卡（这个转发的处理过程不是很清楚，究竟是重新组TCP/IP包还是原样把发送进程经过tun0网卡后的TCP/IP包转发，我们不去深究，最终的到达VPN主机2的tun0网卡时都是一样的），数据到达tun0网卡后也会经过内核去分发，内核协议栈把TCP/IP的包头都去掉，只把数据部分交给数据接收进程。

至此，数据收发过程完成。

二、openvpn认证（加密隧道建立）

上面的章节，介绍了openvpn网络中两个主机的数据收发过程，但是两个主机在能进行数据收发前，要先建立数据传输通道，这个就是加入openvpn网络的认证过程。数据通道的建立过程，也是openvpn客户端和服务端进行协商的过程。openvpn采用ssl协议通信，既然采用ssl，那么就会有证书验证，协商对称加密密钥的过程。需要生成ca证书，服务端证书客户端证书等。具体的原理这里不细说，可以自行去谷歌。openvpn客户端认证通过后就和服务端建立了一条加密隧道，也就是一条安全的数据通道。

三、路由表在openvpn组网中的作用

在第一部分介绍原理的时候，说发往数据接收端ip10.8.0.2的数据包，都转发给tun0网卡处理，这是怎么做到的？答案就是路由表。在openvpn客户端和服务端之间认证通过后，openvpn客户端就会修改我们本地的路由表，添加一条路由记录，指示本地路由收到发往ip10.8.0.2的数据包，都交给tun0网卡转发。

除了上面提到的第一个作用之外，在openvpn工作的过程中，路由表还有一个作用，就是把本地所有的数据包（流量）都路由到openvpn服务端的机器上，也就是把openvpn服务端的机器，作为本地机器的网关。把openvpn服务端作为网关的一个作用是，可以让openvpn局域网中的节点，通过openvpn服务端机器访问互联网，作为外网的出口。就类似我们平常用路由器上网的网络拓扑结构，路由器（openvpn服务端所在机器）作为NAT设备，代理接入互联网。真正起到NAT作用的是服务端所在主机的防火墙进程，而不是openvpn服务端进程。linux下的防火墙也是一门高深的技术，功能十分强大。

openvpn客户端认证成功后，路由表的信息类似以下的截图：

```
[root@localhost ~]# route -n
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
0.0.0.0          10.8.0.5         128.0.0.0        UG      0      0      0 tun0
0.0.0.0          10.0.2.2         0.0.0.0          UG      0      0      0 enp0s17
10.0.2.0         0.0.0.0          255.255.255.0    U       0      0      0 enp0s17
10.8.0.0         10.8.0.5         255.255.255.0    UG      0      0      0 tun0
10.8.0.5         0.0.0.0          255.255.255.255 UH      0      0      0 tun0
119.91.225.146  10.0.2.2         255.255.255.255 UGH     0      0      0 enp0s17
128.0.0.0        10.8.0.5         128.0.0.0        UG      0      0      0 tun0
169.254.0.0      0.0.0.0          255.255.0.0      U       1002   0      0 enp0s8
169.254.0.0      0.0.0.0          255.255.0.0      U       1003   0      0 enp0s17
172.17.0.0       0.0.0.0          255.255.0.0      U       0      0      0 docker0
192.168.56.0     0.0.0.0          255.255.255.0    U       0      0      0 enp0s8
[root@localhost ~]# ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
```

CSDN @hacker_lpy

tun0网卡的信息如下：

```
tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet 10.8.0.6 netmask 255.255.255.255 destination 10.8.0.5
    unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 100  (UNSPEC)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
[root@localhost ~]#
```

CSDN @hacker_lpy

即openvpn客户端认证接入openvpn服务器后，分配到的局域网ip是10.8.0.6，服务器本身的局域网ip是10.8.0.5，这是一个点对点的网络。红线圈住的那几条就是openvpn新增的路由规则，其中第一条就是把openvpn服务端作为网关的路由规则。

四、openvpn安装配置过程

1、服务器端 (centos)

我的服务端是部署在centos操作系统上的，安装openvpn就很简单了，直接用yum命令安装：

```
yum install -y openvpn
```

linux下安装的openvpn，客户端和服务端是一起的，运行的是服务端还是客户端，主要取决于配置文件。这个跟stunnel类似。

安装完成后，会有同时安装一些配置样例文件，用下面的命令进行查找：

```
find / -name "*openvpn*"
```

```
[root@localhost openvpn]# find / -name *openvpn*
/run/openvpn-client
/run/openvpn-server
/sys/fs/selinux/booleans/openvpn_can_network_connect
/sys/fs/selinux/booleans/openvpn_enable_homedirs
/sys/fs/selinux/booleans/openvpn_run_unconfined
/etc/selinux/targeted/active/modules/100/openvpn
/etc/selinux/targeted/tmp/modules/100/openvpn
/etc/openvpn
/var/lib/yum/yumdb/o/692cc57ec90929eee0bbbed9a12c3ece35c27f175-openvpn-2.4.12-1.el7-x86_64
/var/lib/openvpn
/var/log/openvpn.log
/root/docker/images/proxy/openvpn-server
/root/docker/images/proxy/openvpn-server/openvpn-status.log
/root/docker/images/proxy/openvpn-server/openvpn-shutdown.sh
/root/docker/images/proxy/openvpn-server/openvpnfile.tar.gz
/root/docker/images/proxy/openvpn-server/openvpn-startup.sh
/root/docker/images/proxy/openvpn-server/openvpn-unencrypt-startup.sh
/root/docker/images/proxy/openvpn-server/openvpn.log
/root/docker/images/proxy/openvpn-server/openvpn-client-static-key
/root/docker/images/proxy/openvpn-server/openvpn-client-static-key/run_openvpn_client.sh
/root/docker/images/test_proxy/stunnel-to-openvpn
/usr/sbin/openvpn
/usr/lib/systemd/system/openvpn-client@.service
/usr/lib/systemd/system/openvpn-server@.service
/usr/lib/systemd/system/openvpn@.service
/usr/lib/firewalld/services/openvpn.xml
/usr/lib/tmpfiles.d/openvpn.conf
/usr/lib64/openvpn
/usr/lib64/openvpn/plugins/openvpn-plugin-auth-pam.so
/usr/lib64/openvpn/plugins/openvpn-plugin-down-root.so
/usr/share/doc/openvpn-2.4.12
/usr/share/doc/openvpn-2.4.12/contrib/openvpn-fwmarkroute-1.00
/usr/share/doc/openvpn-2.4.12/sample/sample-config-files/openvpn-shutdown.sh
/usr/share/doc/openvpn-2.4.12/sample/sample-config-files/openvpn-startup.sh
```

CSDN @hacker_lpy

一般是/usr/share/doc/openvpn-xxx目录下，xxx是版本号。其中我们需要的几个文件就在sample/sample-config-files目录下，如下图：

```

[root@localhost sample-config-files]# pwd
/usr/share/doc/openvpn-2.4.12/sample/sample-config-files
[root@localhost sample-config-files]# ll
总用量 80
-rw-r--r--. 1 root root 3585 3月 17 2022 client.conf
-rw-r--r--. 1 root root 3562 3月 17 2022 firewall.sh
-rw-r--r--. 1 root root 62 3月 17 2022 home.up
-rw-r--r--. 1 root root 672 3月 17 2022 loopback-client
-rw-r--r--. 1 root root 675 3月 17 2022 loopback-server
-rw-r--r--. 1 root root 62 3月 17 2022 office.up
-rw-r--r--. 1 root root 63 3月 17 2022 openvpn-shutdown.sh
-rw-r--r--. 1 root root 776 3月 17 2022 openvpn-startup.sh
-rw-r--r--. 1 root root 131 3月 17 2022 README
-rw-r--r--. 1 root root 820 3月 18 2022 roadwarrior-client.conf
-rw-r--r--. 1 root root 1498 3月 18 2022 roadwarrior-server.conf
-rw-r--r--. 1 root root 10784 3月 17 2022 server.conf
-rw-r--r--. 1 root root 1778 3月 17 2022 static-home.conf
-rw-r--r--. 1 root root 1724 3月 17 2022 static-office.conf
-rw-r--r--. 1 root root 1937 3月 17 2022 tls-home.conf
-rw-r--r--. 1 root root 1948 3月 17 2022 tls-office.conf
-rw-r--r--. 1 root root 199 3月 17 2022 xinetd-client-config
-rw-r--r--. 1 root root 989 3月 17 2022 xinetd-server-config
[root@localhost sample-config-files]#

```

CSDN @hacker_lpy

安装服务端的时候，我们复制firewall.sh、openvpn-shutdown.sh、openvpn-startup.sh、server.conf这几个文件到一个单独的目录下，比如/root/openvpn-server，如果你能看懂这几个配置文件里面的信息，就可以对应着修改一下，以符合你的部署要求。

下面是我修改以后的配置：

openvpn-startup.sh文件：

```

1 #!/bin/sh
2
3 # A sample OpenVPN startup script
4 # for Linux.
5
6 # openvpn config file directory
7 # 此处获取当前目录路径
8 dir=`pwd`

```



```
9 | 10 | # load the firewall
11 | # 调用firewall.sh脚本配置防火墙
12 | $dir/firewall.sh
13 | #iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
14 |
15 | # load TUN/TAP kernel module
16 | modprobe tun
17 |
18 | # enable IP forwarding 打开ipv4的转发功能
19 | echo 1 > /proc/sys/net/ipv4/ip_forward
20 |
21 | # Invoke openvpn for each VPN tunnel
22 | # in daemon mode. Alternatively,
23 | # you could remove "--daemon" from
24 | # the command line and add "daemon"
25 | # to the config file.
26 | #
27 | # Each tunnel should run on a separate
28 | # UDP port. Use the "port" option
29 | # to control this. Like all of
30 | # OpenVPN's options, you can
31 | # specify "--port 8000" on the command
32 | # line or "port 8000" in the config
33 | # file.
34 |
35 | # 启动openvpn服务端
36 | openvpn --cd $dir --daemon --config server.conf
```

firewall.sh文件:

```
1 | #!/bin/sh
2 |
3 | # A Sample OpenVPN-aware firewall.
4 |
5 | # eth0 is connected to the internet.
6 | # eth1 is connected to a private subnet.
7 |
8 | # Change this subnet to correspond to your private
9 | # ethernet subnet. Home will use HOME_NET/24 and
10 | # Office will use OFFICE_NET/24.
```

```
11 PRIVATE=10.8.0.0/24
12 |
13 # Loopback address
14 LOOP=127.0.0.1
15
16 # Delete old iptables rules
17 # and temporarily block all traffic.
18 iptables -P OUTPUT DROP
19 iptables -P INPUT DROP
20 iptables -P FORWARD DROP
21 # 这条是删除原有的防火墙规则，删除的是filter表的规则，filter表的规则部分是作用在INPUT链上的
22 iptables -F
23
24 # Set default policies 重新设置链的默认策略
25 iptables -P OUTPUT ACCEPT
26 iptables -P INPUT DROP
27 iptables -P FORWARD DROP
28
29 # Prevent external packets from using loopback addr 这里是避免外部发来的数据包的源地址和目的
30 # 地址是回环地址，从而错误的递交给本地的进程。这应该是避免攻击的一些手段
31 iptables -A INPUT -i eth0 -s $LOOP -j DROP
32 iptables -A FORWARD -i eth0 -s $LOOP -j DROP
33 iptables -A INPUT -i eth0 -d $LOOP -j DROP
34 iptables -A FORWARD -i eth0 -d $LOOP -j DROP
35
36 # Anything coming from the Internet should have a real Internet address
37 # 这里是对一些有可能是错误发送的包进行丢弃
38 iptables -A FORWARD -i eth0 -s 192.168.0.0/16 -j DROP
39 iptables -A FORWARD -i eth0 -s 172.16.0.0/12 -j DROP
40 iptables -A FORWARD -i eth0 -s 10.0.0.0/8 -j DROP
41 iptables -A INPUT -i eth0 -s 192.168.0.0/16 -j DROP
42 iptables -A INPUT -i eth0 -s 172.16.0.0/12 -j DROP
43 iptables -A INPUT -i eth0 -s 10.0.0.0/8 -j DROP
44
45 # Block outgoing NetBios (if you have windows machines running
46 # on the private subnet). This will not affect any NetBios
47 # traffic that flows over the VPN tunnel, but it will stop
48 # local windows machines from broadcasting themselves to
49 # the internet.
50 # 这里是对windows的一些服务端口进行屏蔽，我们是linux不用管
51 iptables -A FORWARD -p tcp --sport 137:139 -o eth0 -j DROP
52 iptables -A FORWARD -p udp --sport 137:139 -o eth0 -j DROP
```

```
53 iptables -A OUTPUT -p tcp --sport 137:139 -o eth0 -j DROP
54 | iptables -A OUTPUT -p udp --sport 137:139 -o eth0 -j DROP
55
56 # Check source address validity on packets going out to internet
57 #iptables -A FORWARD -s ! $PRIVATE -i eth1 -j DROP
58 #iptables -A FORWARD -m iprange ! --src-range 10.0.0.0-10.0.0.255 -i eth1 -j DROP
59
60 # Allow local loopback 这里是对回环地址放行
61 iptables -A INPUT -s $LOOP -j ACCEPT
62 iptables -A INPUT -d $LOOP -j ACCEPT
63
64 # Allow incoming pings (can be disabled) 这里是对icmp协议放行，也就是能响应ping命令
65 iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
66
67 # Allow services such as www and ssh (can be disabled) 这里是对常用的服务端口入栈放行
68 iptables -A INPUT -p tcp --dport http -j ACCEPT
69 iptables -A INPUT -p tcp --dport ssh -j ACCEPT
70 iptables -A INPUT -p tcp --dport 6679 -j ACCEPT
71
72 # Allow incoming OpenVPN packets
73 # Duplicate the line below for each
74 # OpenVPN tunnel, changing --dport n
75 # to match the OpenVPN UDP port.
76 #
77 # In OpenVPN, the port number is
78 # controlled by the --port n option.
79 # If you put this option in the config
80 # file, you can remove the leading '--'
81 #
82 # If you taking the stateful firewall
83 # approach (see the OpenVPN HOWTO),
84 # then comment out the line below.
85
86 # 这里是自己想要放行的端口和协议数据，有其他端口可以自行添加
87 iptables -A INPUT -p udp --dport 1194 -j ACCEPT
88 iptables -A INPUT -p udp --dport 9981 -j ACCEPT
89 iptables -A INPUT -p tcp --dport 9981 -j ACCEPT
90 iptables -A INPUT -p tcp --dport 6379 -j ACCEPT
91 iptables -A INPUT -p udp --dport 6379 -j ACCEPT
92 iptables -A INPUT -p udp --dport 6680 -j ACCEPT
93 iptables -A INPUT -p tcp --dport 6680 -j ACCEPT
94 iptables -A INPUT -p tcp --dport 1723 -j ACCEPT
```

```

95 |
96 | # Allow packets from TUN/TAP devices.
97 | # When OpenVPN is run in a secure mode,
98 | # it will authenticate packets prior
99 | # to their arriving on a tun or tap
100 | # interface. Therefore, it is not
101 | # necessary to add any filters here,
102 | # unless you want to restrict the
103 | # type of packets which can flow over
104 | # the tunnel.
105 |
106 | # 这里主要是对tun+网卡和tap+网卡的入栈和转发数据包放行，+号应该表示的是一个通配符
107 | iptables -A INPUT -i tun+ -j ACCEPT
108 | iptables -A FORWARD -i tun+ -j ACCEPT
109 | iptables -A INPUT -i tap+ -j ACCEPT
110 | iptables -A FORWARD -i tap+ -j ACCEPT
111 |
112 | # Allow packets from private subnets 这里是对eth1网卡的入栈和转发数据包放行
113 | iptables -A INPUT -i eth1 -j ACCEPT
114 | iptables -A FORWARD -i eth1 -j ACCEPT
115 |
116 | # Keep state of connections from local machine and private subnets
117 | # 这里是放行状态为NEW的出栈数据包
118 | iptables -A OUTPUT -m state --state NEW -o eth0 -j ACCEPT
119 | # 这里放行已经建立连接和关联的入栈数据包
120 | iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
121 | # 这里放行需要转发的包，有这两条，本机才能作为NAT代理客户端访问互联网
122 | iptables -A FORWARD -m state --state NEW -o eth0 -j ACCEPT
123 | iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
124 |
125 | # Masquerade local subnet
126 | # 对于需要通过openvpn服务端访问外网的客户端节点，这个防火墙配置是必需的，也是最重要的配置
127 | # 这个防火墙的配置是将路由后的源IP地址是10.8.0.0/24，输出网卡是eth0的数据包的源ip地址进行修改，改成本机的ip地址，这样响应数据包才能正确的给我们转发回来
128 | iptables -t nat -A POSTROUTING -s $PRIVATE -o eth0 -j MASQUERADE

```

这个防火墙配置，比较考验技术，新手一般看不懂，除非对防火墙的原理很熟悉，如果要介绍完全这些防火墙配置，需要另一篇长篇大论。这个配置是根据样例文件修改后的，大体还是不变，只做了小部分的修改。这个官方给的样例，是对防火墙规则做了严密的设计。如果你对安全性要求不高，可以将防火墙的INPUT、FORWARD、OUTPUT链的策略都改成ACCEPT,那么就只需要少许的防火墙配置就能达到目的，比如如下的配置（没有实际调试过，不保证正确性）：

```

2 | 3 | # 清空原有INPUT链的防火墙规则
4 | iptables -F
5 | # 放行所有输出包
6 | iptables -P OUTPUT ACCEPT
7 | # 放行所有输入包
8 | iptables -P INPUT ACCEPT
9 | # 放行所有需要转发的包
10 | iptables -P FORWARD ACCEPT
11 |
12 | # 对于需要通过openvpn服务端访问外网的客户端节点，这个防火墙配置是必需的，也是最重要的配置
13 | # 这个防火墙的配置是将路由后的源IP地址是10.8.0.0/24，输出网卡是eth0的数据包的源ip地址进行修改，改成本机的ip地址，这样响应数据包才能正确的给我们转发回来
14 | iptables -t nat -A POSTROUTING -s $PRIVATE -o eth0 -j MASQUERADE

```

server.conf文件:

```

1 | #####
2 | # Sample OpenVPN 2.0 config file for #
3 | # multi-client server. #
4 | # #
5 | # This file is for the server side #
6 | # of a many-clients <-> one-server #
7 | # OpenVPN configuration. #
8 | # #
9 | # OpenVPN also supports #
10 | # single-machine <-> single-machine #
11 | # configurations (See the Examples page #
12 | # on the web site for more info). #
13 | # #
14 | # This config should work on Windows #
15 | # or Linux/BSD systems. Remember on #
16 | # Windows to quote pathnames and use #
17 | # double backslashes, e.g.: #
18 | # "C:\\Program Files\\OpenVPN\\config\\foo.key" #
19 | # #
20 | # Comments are preceded with '#' or ';' #
21 | #####
22 |
23 | # Which local IP address should OpenVPN
24 | # listen on? (optional)
25 | ;local a.b.c.d

```

```
26
27 # Which TCP/UDP port should OpenVPN listen on?
28 # If you want to run multiple OpenVPN instances
29 # on the same machine, use a different port
30 # number for each one. You will need to
31 # open up this port on your firewall.
32 # 这个是监听端口
33 port 6379
34
35 # TCP or UDP server?
36 # 使用的协议，推荐使用udp协议，速度会快两三倍
37 proto tcp
38 ;proto udp
39
40 # "dev tun" will create a routed IP tunnel,
41 # "dev tap" will create an ethernet tunnel.
42 # Use "dev tap0" if you are ethernet bridging
43 # and have precreated a tap0 virtual interface
44 # and bridged it with your ethernet interface.
45 # If you want to control access policies
46 # over the VPN, you must create firewall
47 # rules for the the TUN/TAP interface.
48 # On non-Windows systems, you can give
49 # an explicit unit number, such as tun0.
50 # On Windows, use "dev-node" for this.
51 # On most systems, the VPN will not function
52 # unless you partially or fully disable
53 # the firewall for the TUN/TAP interface.
54 # 这里我们就使用tun, tap是以太网桥接方式，没有深究
55 ;dev tap
56 dev tun
57
58 # Windows needs the TAP-Win32 adapter name
59 # from the Network Connections panel if you
60 # have more than one. On XP SP2 or higher,
61 # you may need to selectively disable the
62 # Windows firewall for the TAP adapter.
63 # Non-Windows systems usually don't need this.
64 ;dev-node MyTap
65
66 # SSL/TLS root certificate (ca), certificate
```

```
67 # (cert), and private key (key). Each client 68 # and the server must have their own cert and
69 # key file. The server and all clients will
70 # use the same ca file.
71 #
72 # See the "easy-rsa" directory for a series
73 # of scripts for generating RSA certificates
74 # and private keys. Remember to use
75 # a unique Common Name for the server
76 # and each of the client certificates.
77 #
78 # Any X509 key management system can be used.
79 # OpenVPN can also use a PKCS #12 formatted key file
80 # (see "pkcs12" directive in man page).
81 # 这里配置ca证书, 服务端证书, 服务端密钥
82 ca ./crt/ca.crt
83 cert ./crt/server.crt
84 key ./crt/server.key # This file should be kept secret
85
86 # Diffie hellman parameters.
87 # Generate your own with:
88 # openssl dhparam -out dh2048.pem 2048
89 # 都告诉你了, 直接使用命令openssl dhparam -out dh2048.pem 2048生成, 具体作用可以自行谷歌
90 dh ./crt/dh2048.pem
91
92 # Network topology
93 # Should be subnet (addressing via IP)
94 # unless Windows clients v2.0.9 and lower have to
95 # be supported (then net30, i.e. a /30 per client)
96 # Defaults to net30 (not recommended)
97 ;topology subnet
98
99 # Configure server mode and supply a VPN subnet
100 # for OpenVPN to draw client addresses from.
101 # The server will take 10.8.0.1 for itself,
102 # the rest will be made available to clients.
103 # Each client will be able to reach the server
104 # on 10.8.0.1. Comment this line out if you are
105 # ethernet bridging. See the man page for more info.
106 # 这里配置openvpn网络, 包括ip和子网掩码, 我们就用默认的
107 server 10.8.0.0 255.255.255.0
108
```

```
109 # Maintain a record of client <-> virtual IP address110 # associations in this file. If OpenVPN goes down or
111 # is restarted, reconnecting clients can be assigned
112 # the same virtual IP address from the pool that was
113 # previously assigned.
114 # 这里是为了保证同一个客户端多次连接后分配给相同的局域网ip地址
115 ifconfig-pool-persist ipp.txt
116
117 # Configure server mode for ethernet bridging.
118 # You must first use your OS's bridging capability
119 # to bridge the TAP interface with the ethernet
120 # NIC interface. Then you must manually set the
121 # IP/netmask on the bridge interface, here we
122 # assume 10.8.0.4/255.255.255.0. Finally we
123 # must set aside an IP range in this subnet
124 # (start=10.8.0.50 end=10.8.0.100) to allocate
125 # to connecting clients. Leave this line commented
126 # out unless you are ethernet bridging.
127 ;server-bridge 10.8.0.4 255.255.255.0 10.8.0.50 10.8.0.100
128
129 # Configure server mode for ethernet bridging
130 # using a DHCP-proxy, where clients talk
131 # to the OpenVPN server-side DHCP server
132 # to receive their IP address allocation
133 # and DNS server addresses. You must first use
134 # your OS's bridging capability to bridge the TAP
135 # interface with the ethernet NIC interface.
136 # Note: this mode only works on clients (such as
137 # Windows), where the client-side TAP adapter is
138 # bound to a DHCP client.
139 ;server-bridge
140
141 # Push routes to the client to allow it
142 # to reach other private subnets behind
143 # the server. Remember that these
144 # private subnets will also need
145 # to know to route the OpenVPN client
146 # address pool (10.8.0.0/255.255.255.0)
147 # back to the OpenVPN server.
148 ;push "route 192.168.10.0 255.255.255.0"
149 ;push "route 192.168.20.0 255.255.255.0"
150
```



```
151 # To assign specific IP addresses to specific152 # clients or if a connecting client has a private
153 # subnet behind it that should also have VPN access,
154 # use the subdirectory "ccd" for client-specific
155 # configuration files (see man page for more info).
156
157 # EXAMPLE: Suppose the client
158 # having the certificate common name "Thelonious"
159 # also has a small subnet behind his connecting
160 # machine, such as 192.168.40.128/255.255.255.248.
161 # First, uncomment out these lines:
162 ;client-config-dir ccd
163 ;route 192.168.40.128 255.255.255.248
164 # Then create a file ccd/Thelonious with this line:
165 #   iroute 192.168.40.128 255.255.255.248
166 # This will allow Thelonious' private subnet to
167 # access the VPN. This example will only work
168 # if you are routing, not bridging, i.e. you are
169 # using "dev tun" and "server" directives.
170
171 # EXAMPLE: Suppose you want to give
172 # Thelonious a fixed VPN IP address of 10.9.0.1.
173 # First uncomment out these lines:
174 ;client-config-dir ccd
175 ;route 10.9.0.0 255.255.255.252
176 # Then add this line to ccd/Thelonious:
177 #   ifconfig-push 10.9.0.1 10.9.0.2
178
179 # Suppose that you want to enable different
180 # firewall access policies for different groups
181 # of clients. There are two methods:
182 # (1) Run multiple OpenVPN daemons, one for each
183 # group, and firewall the TUN/TAP interface
184 # for each group/daemon appropriately.
185 # (2) (Advanced) Create a script to dynamically
186 # modify the firewall in response to access
187 # from different clients. See man
188 # page for more info on learn-address script.
189 ;learn-address ./script
190
191 # If enabled, this directive will configure
192 # all clients to redirect their default
```

```
193 # network gateway through the VPN, causing 194 # all IP traffic such as web browsing and
195 # and DNS lookups to go through the VPN
196 # (The OpenVPN server machine may need to NAT
197 # or bridge the TUN/TAP interface to the internet
198 # in order for this to work properly).
199 # 这里是将客户端的所有流量定向到服务器端，也就是让客户端修改自己的路由表
200 # 把服务器端的局域网ip地址作为网关，在上面的《三、路由表在openvpn组网中的作用》中有提到
201 push "redirect-gateway def1 bypass-dhcp"
202
203 # Certain Windows-specific network settings
204 # can be pushed to clients, such as DNS
205 # or WINS server addresses. CAVEAT:
206 # http://openvpn.net/faq.html#dhcpcaveats
207 # The addresses below refer to the public
208 # DNS servers provided by opendns.com.
209 # 这里是给客户端推送DNS服务器地址，如果没有这个推送，客户端还是会用本地默认的DNS地址，有可能导致
210 # DNS解析失败，这个我是使用命令查看服务器本地的DNS服务器：cat /etc/resolv.conf
211 # 你也可以使用其他的DNS服务器，只要客户端能正确解析出域名
212 push "dhcp-option DNS 100.100.2.136"
213 push "dhcp-option DNS 100.100.2.138"
214
215 # Uncomment this directive to allow different
216 # clients to be able to "see" each other.
217 # By default, clients will only see the server.
218 # To force clients to only see the server, you
219 # will also need to appropriately firewall the
220 # server's TUN/TAP interface.
221 # 这里是配置让接入到服务器的多个客户端彼此能看到对方，即能正常通信
222 client-to-client
223
224 # Uncomment this directive if multiple clients
225 # might connect with the same certificate/key
226 # files or common names. This is recommended
227 # only for testing purposes. For production use,
228 # each client should have its own certificate/key
229 # pair.
230 #
231 # IF YOU HAVE NOT GENERATED INDIVIDUAL
232 # CERTIFICATE/KEY PAIRS FOR EACH CLIENT,
233 # EACH HAVING ITS OWN UNIQUE "COMMON NAME",
234 # UNCOMMENT THIS LINE OUT.
```

```
235 # 这里允许使用同一个证书的多个客户端登录
236 duplicate-cn
237
238 # The keepalive directive causes ping-like
239 # messages to be sent back and forth over
240 # the link so that each side knows when
241 # the other side has gone down.
242 # Ping every 10 seconds, assume that remote
243 # peer is down if no ping received during
244 # a 120 second time period.
245 keepalive 10 120
246
247 # For extra security beyond that provided
248 # by SSL/TLS, create an "HMAC firewall"
249 # to help block DoS attacks and UDP port flooding.
250 #
251 # Generate with:
252 #   openssl genpkey --genkey --secret ta.key
253 #
254 # The server and each client must have
255 # a copy of this key.
256 # The second parameter should be '0'
257 # on the server and '1' on the clients.
258 # 这里的ta.key文件也是根据命令openssl genpkey --genkey --secret ta.key直接生成
259 tls-auth ./crt/ta.key 0 # This file is secret
260
261 # Select a cryptographic cipher.
262 # This config item must be copied to
263 # the client config file as well.
264 # Note that v2.4 client/server will automatically
265 # negotiate AES-256-GCM in TLS mode.
266 # See also the ncp-cipher option in the manpage
267 # 这里是指定对称加密算法，服务端和客户的配置要一样
268 cipher AES-256-GCM
269
270 # Enable compression on the VPN link and push the
271 # option to the client (v2.4+ only, for earlier
272 # versions see below)
273 # 这里是允许压缩，并且把这个压缩配置推送给了客户端，客户端可以不用配置
274 compress lz4-v2
275 push "compress lz4-v2"
276
```

```
277 # For compression compatible with older clients use comp-lzo
278 # If you enable it here, you must also
279 # enable it in the client config file.
280 ;comp-lzo
281
282 # The maximum number of concurrently connected
283 # clients we want to allow.
284 ;max-clients 100
285
286 # It's a good idea to reduce the OpenVPN
287 # daemon's privileges after initialization.
288 #
289 # You can uncomment this out on
290 # non-Windows systems.
291 # 这里设置openvpn允许使用的角色和用户组
292 user root
293 group root
294
295 # The persist options will try to avoid
296 # accessing certain resources on restart
297 # that may no longer be accessible because
298 # of the privilege downgrade.
299 # 这里不是很懂，直接保持原样，没有这个有时会报错
300 persist-key
301 persist-tun
302
303 # Output a short status file showing
304 # current connections, truncated
305 # and rewritten every minute.
306 # 这里是每分钟记录客户端的状态
307 status openvpn-status.log
308
309 # By default, log messages will go to the syslog (or
310 # on Windows, if running as a service, they will go to
311 # the "%Program Files%\OpenVPN\log" directory).
312 # Use log or log-append to override this default.
313 # "log" will truncate the log file on OpenVPN startup,
314 # while "log-append" will append to it. Use one
315 # or the other (but not both).
316 # 这里是日志输出文件，log是每次启动覆盖之前的日志，log-append会往后追加，不要同时配置两个
317 log ./openvpn.log
318 ;log-append ./openvpn.log
```

```

319 |
320 | # Set the appropriate level of log
321 | # file verbosity.
322 | #
323 | # 0 is silent, except for fatal errors
324 | # 4 is reasonable for general usage
325 | # 5 and 6 can help to debug connection problems
326 | # 9 is extremely verbose
327 | # 这里是日志输出等级，数字越高，日志越详细
328 | verb 3
329 |
330 | # Silence repeating messages. At most 20
331 | # sequential messages of the same message
332 | # category will be output to the log.
333 | ;mute 20
334 |
335 | # Notify the client that when the server restarts so it
336 | # can automatically reconnect.
337 | ;explicit-exit-notify 1

```

这个server.conf服务端配置文件也是根据样例文件做的一些修改。一些比较重要的配置我已经标注出来了。证书的话需要自己制作，使用openssl命令就可以生成。参考：[OPenSSL-生成证书-CSDN博客](#)

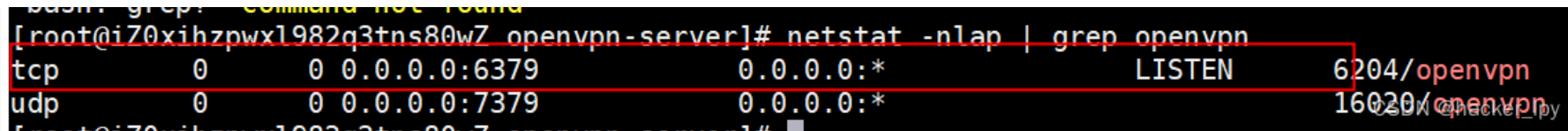
openvpn-shutdown.sh文件就不用改了，所有配置都配好就可以启动openvpn服务器：

./openvpn-startup.sh

如果有报错就看当前目录下的openvpn.log文件查看错误

检查是否启动成功：

netstat -nltp | grep openvpn



```

[root@i70xihzpxl982q3tns80wZ openvpn-server]# netstat -nltp | grep openvpn
tcp        0      0 0.0.0.0:6379          0.0.0.0:*           LISTEN     6204/openvpn
udp        0      0 0.0.0.0:7379          0.0.0.0:*           LISTEN     16020/openvpn

```

看到6379端口在监听就说明启动成功

2、客户端 (centos)

服务端启动成功后，可以使用linux的客户端连接一下，看能不能正常工作，一般都不会一次成功，都会有各种各样的问题。

centos系统上的openvpn客户端和服务端是不区分的，前面讲服务的部署的时候提到有一个client.conf的样例配置文件，我也是拿这个来改了一下。

client.conf文件如下：

```
1 #####
2 # Sample client-side OpenVPN 2.0 config file #
3 # for connecting to multi-client server.      #
4 #                                              #
5 # This configuration can be used by multiple #
6 # clients, however each client should have   #
7 # its own cert and key files.                #
8 #                                              #
9 # On Windows, you might want to rename this  #
10 # file so it has a .ovpn extension           #
11 #####
12
13 # Specify that we are a client and that we
14 # will be pulling certain config file directives
15 # from the server.
16 # 这里需要指明是客户端才行
17 client
18
19 # Use the same setting as you are using on
20 # the server.
21 # On most systems, the VPN will not function
22 # unless you partially or fully disable
23 # the firewall for the TUN/TAP interface.
24 # 这里跟服务端的配置一样
25 ;dev tap
26 dev tun
27
28 # Windows needs the TAP-Win32 adapter name
29 # from the Network Connections panel
30 # if you have more than one.  On XP SP2,
31 # you may need to disable the firewall
32 # for the TAP adapter.
33 ;dev-node MyTap
```

```
34 35 | # Are we connecting to a TCP or
36 # UDP server? Use the same setting as
37 # on the server.
38 # 这里是使用协议，必须跟服务端一致
39 proto tcp
40 ;proto udp
41
42 # The hostname/IP and port of the server.
43 # You can have multiple remote entries
44 # to load balance between the servers.
45 # 这里填服务器端的IP端口
46 remote 1.1.1.1 6679
47 #remote my-server-2 1194
48
49 # Choose a random host from the remote
50 # list for load-balancing. Otherwise
51 # try hosts in the order specified.
52 ;remote-random
53
54 # Keep trying indefinitely to resolve the
55 # host name of the OpenVPN server. Very useful
56 # on machines which are not permanently connected
57 # to the internet such as laptops.
58 # 这里就用默认的配置
59 resolv-retry infinite
60
61 # Most clients don't need to bind to
62 # a specific local port number.
63 # 这里指明通信绑定的端口，一般都不指定，系统随机分配
64 nobind
65
66 # Downgrade privileges after initialization (non-Windows only)
67 ;user nobody
68 ;group nobody
69
70 # Try to preserve some state across restarts.
71 # 这里不是很懂用来干嘛的，看注释是重启的时候维持一些状态，我们就使用默认配置
72 persist-key
73 persist-tun
74
75 # If you are connecting through an
```

```
76 # HTTP proxy to reach the actual OpenVPN 77 | # server, put the proxy server/IP and
78 # port number here. See the man page
79 # if your proxy server requires
80 # authentication.
81 ;http-proxy-retry # retry on connection failures
82 ;http-proxy [proxy server] [proxy port #]
83
84 # Wireless networks often produce a lot
85 # of duplicate packets. Set this flag
86 # to silence duplicate packet warnings.
87 ;mute-replay-warnings
88
89 # SSL/TLS parms.
90 # See the server config file for more
91 # description. It's best to use
92 # a separate .crt/.key file pair
93 # for each client. A single ca
94 # file can be used for all clients.
95 # 这里就配置ssl通信用的ca证书，客户端证书，客户端私钥，可以指定路径，默认是当前目录
96 ca ca.crt
97 cert client.crt
98 key client.key
99
100 # Verify server certificate by checking that the
101 # certicate has the correct key usage set.
102 # This is an important precaution to protect against
103 # a potential attack discussed here:
104 # http://openvpn.net/howto.html#mitm
105 #
106 # To use this feature, you will need to generate
107 # your server certificates with the keyUsage set to
108 # digitalSignature, keyEncipherment
109 # and the extendedKeyUsage to
110 # serverAuth
111 # EasyRSA can do this for you.
112 # 这里需要注册默认的配置，默认是打开的，因为我制作的服务端证书没有它指定的几个信息，我就不做校验了，当然你也可以按他的指示重新制作服务端证书，这个是为了避免潜在的攻击的
113 #remote-cert-tls server
114
115 # If a tls-auth key is used on the server
116 # then every client must also have the key.
117 # 这里配置tls认证key，这个就是服务器使用的ta.key，两端是同一个文件
```



```
118 | tls-auth ta.key 1 119 |
120 | # Select a cryptographic cipher.
121 | # If the cipher option is used on the server
122 | # then you must also specify it here.
123 | # Note that v2.4 client/server will automatically
124 | # negotiate AES-256-GCM in TLS mode.
125 | # See also the ncp-cipher option in the manpage
126 | # 加密算法，跟服务端要一致，但是我发现服务端配置的是AES-256-GCM，这里配置的是AES-256-CBC，也一样能正常加解密，比较奇怪，可能最终协商一致就可以
127 | cipher AES-256-CBC
128 |
129 | # Enable compression on the VPN link.
130 | # Don't enable this unless it is also
131 | # enabled in the server config file.
132 | #comp-lzo
133 |
134 | # Set log file verbosity.
135 | # 日志等级
136 | verb 3
137 |
138 | # Silence repeating messages
139 | ;mute 20
140 | # 服务端已经配置，这里不需要重新配置
141 | ;redirect-gateway autolocal
```

配置完成后开始启动客户端：openvpn --config client.conf

如果打印如下的日志，就是连接成功了

```
Sat Feb 24 22:34:00 2024 TLS: Initial packet from [AF_INET] 192.168.43.0:6679, sid=9dc9802f ff5578f2
Sat Feb 24 22:34:00 2024 VERIFY OK: depth=1, C=CN, ST=shanghai, L=shanghai, O=bx, OU=bx, CN=CA
Sat Feb 24 22:34:00 2024 VERIFY OK: depth=0, C=CN, ST=shanghai, O=bx, OU=bx, CN=8.211.144.206
Sat Feb 24 22:34:01 2024 WARNING: 'link-mtu' is used inconsistently, local='link-mtu 1559', remote='link-mtu 1552'
Sat Feb 24 22:34:01 2024 WARNING: 'cipher' is used inconsistently, local='cipher AES-256-CBC', remote='cipher AES-256-GCM'
Sat Feb 24 22:34:01 2024 WARNING: 'auth' is used inconsistently, local='auth SHA1', remote='auth [null-digest]'
Sat Feb 24 22:34:01 2024 WARNING: 'comp-lzo' is present in remote config but missing in local config, remote='comp-lzo'
Sat Feb 24 22:34:01 2024 Control Channel: TLSv1.2, cipher TLSv1/SSLv3 ECDHE-RSA-AES256-GCM-SHA384, 2048 bit RSA
Sat Feb 24 22:34:01 2024 [AF_INET] 192.168.43.0:6679 Peer Connection Initiated with [AF_INET] 10.8.0.6:6679
Sat Feb 24 22:34:02 2024 SENT CONTROL [AF_INET] 192.168.43.0:6679: 'PUSH_REQUEST' (status=1)
Sat Feb 24 22:34:03 2024 PUSH: Received control message: 'PUSH_REPLY,route 192.168.43.0 255.255.255.0,redirect-gateway def1 bypass-dhcp,dhcp-option DNS 1138,compress lz4-v2,route 10.8.0.0 255.255.255.0,topology net30,ping 10,ping-restart 120,ifconfig 10.8.0.6 10.8.0.5,peer-id 0,cipher AES-256-GCM'
Sat Feb 24 22:34:03 2024 OPTIONS IMPORT: timers and/or timeouts modified
Sat Feb 24 22:34:03 2024 OPTIONS IMPORT: compression parms modified
Sat Feb 24 22:34:03 2024 OPTIONS IMPORT: --ifconfig/up options modified
Sat Feb 24 22:34:03 2024 OPTIONS IMPORT: route options modified
Sat Feb 24 22:34:03 2024 OPTIONS IMPORT: --ip-win32 and/or --dhcp-option options modified
Sat Feb 24 22:34:03 2024 OPTIONS IMPORT: peer-id set
Sat Feb 24 22:34:03 2024 OPTIONS IMPORT: adjusting link_mtu to 1626
Sat Feb 24 22:34:03 2024 OPTIONS IMPORT: data channel crypto options modified
Sat Feb 24 22:34:03 2024 Data Channel: using negotiated cipher 'AES-256-GCM'
Sat Feb 24 22:34:03 2024 Outgoing Data Channel: Cipher 'AES-256-GCM' initialized with 256 bit key
Sat Feb 24 22:34:03 2024 Incoming Data Channel: Cipher 'AES-256-GCM' initialized with 256 bit key
Sat Feb 24 22:34:03 2024 ROUTE_GATEWAY 192.168.58.2/255.255.255.0 IFACE=ens33 HWADDR=00:0c:29:f4:13:23
Sat Feb 24 22:34:03 2024 TUN/TAP device tun0 opened
Sat Feb 24 22:34:03 2024 TUN/TAP TX queue length set to 100
Sat Feb 24 22:34:03 2024 /sbin/ip link set dev tun0 up mtu 1500
Sat Feb 24 22:34:03 2024 /sbin/ip addr add dev tun0 local 10.8.0.6 peer 10.8.0.5
Sat Feb 24 22:34:03 2024 ROUTE remote_host is NOT LOCAL
Sat Feb 24 22:34:03 2024 /sbin/ip route add 0.0.0.0/1 via 192.168.58.2
Sat Feb 24 22:34:03 2024 /sbin/ip route add 0.0.0.0/1 via 10.8.0.5
Sat Feb 24 22:34:03 2024 /sbin/ip route add 128.0.0.0/1 via 10.8.0.5
Sat Feb 24 22:34:03 2024 /sbin/ip route add 192.168.43.0/24 via 10.8.0.5
Sat Feb 24 22:34:03 2024 /sbin/ip route add 10.8.0.0/24 via 10.8.0.5
Sat Feb 24 22:34:03 2024 WARNING: this configuration may cache passwords in memory -- use the auth-nocache option to prevent this
Sat Feb 24 22:34:03 2024 Initialization Sequence Completed
```

连接成功后我们查看一下网卡配置：

```
[root@localhost ~]# ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
    ether 02:42:f3:26:31:b9 txqueuelen 0 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.58.135 netmask 255.255.255.0 broadcast 192.168.58.255
    inet6 fe80::4a7f:fd50:6c04:87ff prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:f4:13:23 txqueuelen 1000 (Ethernet)
    RX packets 450 bytes 42913 (41.9 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 315 bytes 40469 (39.5 KiB)
    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 1000 (Local Loopback)
    RX packets 68 bytes 5916 (5.7 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 68 bytes 5916 (5.7 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet 10.8.0.6 netmask 255.255.255.255 destination 10.8.0.5
    inet6 fe80::987d:a83:6958:a474 prefixlen 64 scopeid 0x20<link>
    unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 100 (UNSPEC)
    RX packets 19 bytes 1444 (1.4 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 23 bytes 1664 (1.6 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[root@localhost ~]#
```

CSDN @hacker_lpy

看到tun0网卡已经被创建好，服务端分配给我们客户端的ip是10.8.0.6,因为是点对点的网络，对端显示的ip是10.8.0.5，但是我们试着ping一下这个ip，发现ping不通，似乎这个ip是不存在的。但是我们此时去服务端查看一下服务端的tun0网卡的ip发现是10.8.0.1，于是尝试ping这个ip证明我们已经和服务端建立了通信：

```
[root@localhost ~]# ping 10.8.0.1
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=288 ms
64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=269 ms
64 bytes from 10.8.0.1: icmp_seq=3 ttl=64 time=497 ms
```

看到能ping通，说明已经能和服务端通信，vpn网络已经组网完成。

在测试过程中我发现客户端和服务端是能进行通信，但是客户端主机解析不了域名，用命令：cat /etc/resolv.conf 查看了一下域名服务器地址，发现域名服务器地址还是本地的ip，没有自动改成服务器端给我们推送的DNS地址。我手动把这个文件的DNS地址改成服务端推送的DNS地址就可以正常进行域名解析了。此时就可以通过openvpn访问外网，客户端所有流量都会通过服务器端转发。

3、客户端 (Android)

对于openvpn的安卓客户端，有两款，一款叫**OpenVPN Connect**，一款叫**OpenVPN For Android**，两款都可以用，可能需要一点技术才能下载到。

openvpn的安卓客户，在制作配置文件的时候会有点麻烦，但是如果掌握了方法也很简单。刚开始我认为配置文件在各个平台都通用，我已经在linux上配置好配置文件，直接拷贝到安卓上就行，但是发现导入配置文件的时候只能导入一个文件，那还有ca证书，客户端证书，客户端私钥，还有ta.key文件怎么导入？没办法只能求助谷歌，找了很久才在openvpn的社区中找到安卓配置文件的制作方法。

原来openvpn的配置文件支持把这些证书文件内联到配置文件中。就拿以上centos客户端的配置文件来举例，需要以下四个配置项

```
1 | ca ca.crt
2 | cert client.crt
3 | key client.key
4 | tls-auth ta.key 1
```

即需要内联四个证书，内联的时候只需要将证书文件内容包含在<configItemName></configItemName>中就可以，例如ca证书内联后即为

```
1 | <ca>
2 | ...
3 | </ca>
```

于是就可以拿上面的centos客户端配置来制作，先把client.conf改成文件名client.ovpn,安卓只支持这个后缀名文件。然后执行下面的命令将证书内联进配置文件:

```
1 | echo "<ca>" >> client.ovpn
2 | cat ca.crt >> client.ovpn
3 | echo "</ca>" >> client.ovpn
4 | echo "<cert>" >> client.ovpn
5 | cat client.crt >> client.ovpn
6 | echo "</cert>" >> client.ovpn
7 | echo "<key>" >> client.ovpn
8 | cat client.key >> client.ovpn
9 | echo "</key>" >> client.ovpn
10 | echo "key-direction 1" >> client.ovpn
11 | echo "<tls-auth>" >> client.ovpn
12 | cat ta.key >> client.ovpn
13 | echo "</tls-auth>" >> client.ovpn
```

因为tls-auth ta.key 1这个配置除了配置ta.key证书文件，还有一个配置参数，所以内联了ta.key文件后，还需要添加key-direction 1这个配置参数。

OpenVPN Connect导入配置文件，就是点右下角的加号->Upload File->BROWSE,选择配置文件后导入。**OpenVPN For Android**导入配置，就是点击右上角的加号->IMPORT导入。这两款App的配置文件都通用

导入配置文件后，就可以点连接，如果没连上可以点右上角的按钮查看日志，然后再排查文件。如果显示连接成功，那么恭喜你，可以使用自己的openvpn了。


文章知识点与官方知识档案匹配，可进一步学习相关知识

网络技能树 MPLS VPN MPLS VPN 47735 人正在系统学习中

2024 FinTechathon 深圳国际金融科技大赛公开课：产品经理专场

成为金融科技领域的产品经理，你需要哪些技能？10月15日，我们的线上公开课将为你揭晓。行业专家、学院副院长、赛道冠军团队队长齐聚，分享他们的独到见解和实战经验。还有多轮抽奖等你来，丰富奖品等...

8 条评论

 阿J~ 热评 强呀强呀，好文支持!

写评论

2、OpenVPN搭建

搭建OpenVPN，centos7搭建VPN，centos7部署OpenVPN，linux如何搭建VPN服务，cetnos7如何部署OpenVPN，centos7如何部署VPN服务

weixin_46371752的博客 6775

【亲测能用！OpenVPN实验教程】Win11主机连CentOS7服务器（用户名密码模式）

经过无数个日日夜夜，无数次调试、崩溃、再调试，甚至急得胃病复发。看到成功连接的那一瞬间，竟然无比平静。写这篇文章是希望其他使用OpenVPN的同志能够少走弯路，提供一些我的经验。和的帮助！

IT_GIRL_XYX的博客 2007

win10openvpn搭建与安卓客户端使用（仅用于内网穿透，不可非法使用）

在有公网动态ip，域名，和端口映射的电脑上进行搭建openvpn服务端，安卓作为客户端

IT赵云的博客 1万+

史上最详细保姆级教程部署OpenVPN

提供给公司与子公司或者公司个人与公司之间建立安全的数据传输

XLB 9774

OpenVN客户端添加路由配置（流量分流）

我们因为某些原因需要特定的流量不进VPN隧道或者进VPN隧道转发，我们就可以通过定义路由实现。路由控制需要由三个参数进行定义： 1、route-nopull 如果在客户端配置文件中配route-nopull，openvpn连接...

zzchances的博客 1万+

openvpn原理

然后使用对方的CA证书，把自己目前使用的数据加密方法加密后发送给对方，由于使用的是对方CA证书加密，所以只有对方CA证书对应的Private key才能解密该数据，这样就保证了此密钥的安全性，并且此密钥...

qq_31532979的博客 1071

openvpn搭建与简单配置

openvpn详解

YFHCSDN的博客 1万+

OpenVPN 简介及部署

OpenVPN 是一个健全且高效的 VPN 守护进程，它支持 SSL/TLS 安全、以太网桥，支持TCP 或者 UDP 代理或者是 NAT 通道传输，支持动态 IP 地址和 DHCP，可支持成百上千的用户，并且可以移植到大多数主...

wang11876的博客 1万+