网络安全课外实验
实验报告

学　　院：	信息科学与工程学院
专业班级：	信息安全 1401
指导老师：	王伟平
学　　号：	0906140129
姓　　名：	郭一涵

目　录

基于 OpenSSL 的 VPN 网络环境搭建

1. 概要介绍

VPN 是常见的一种数据安全传输方式，主要用来利用公共网络假设专用网络，实现一个虚拟的专用网络环境。

2. 实验环境

装有 CentOS7 系统的服务器一台
装有 OpenVPN 客户端，能够联网的计算机一台

3. 实验内容

3.1 CentOS 搭建 OpenVPN 服务器端

由于 OpenVPN Server 不在默认源，所以需要安装 EPEL（Extra Packages for Enterprise Linux）仓库，其中包含有 OpenVPN 包，命令如下

```
wget
http://dl.fedoraproject.org/pub/epel/beta/7/x86_64/ep
el-release-7-0.2.noarch.rpm
rpm -Uvh epel-release-7-0.2.noarch.rpm
```

接着从 Yum 安装 OpenVPN

```
yum install openvpn -y
```

然后从示例配置文件复制一份配置文件到/etc/openvpn

```
cp
/usr/share/doc/openvpn-*/sample/sample-config-files/s
erver.conf /etc/openvpn
```

打开/etc/openvpn/server.conf 编辑：

```
vim /etc/openvpn/server.conf
```

在这里边取消以下的注释

```
push "redirect-gateway def1 bypass-dhcp"
push "dhcp-option DNS 8.8.8.8"
push "dhcp-option DNS 8.8.4.4"
user nobody
group nobody
```

在配置文件更改完毕之后，开始生成Keys和Certifications

首先安装easy-rsa

```
yum install easy-rsa
```

将相关文件复制到OpenVPN的目录

```
cp -R /usr/share/easy-rsa/ /etc/openvpn
```

这些完成之后，开始调整Keys的生成配置

首先修改vars文件

```
vim /etc/openvpn/easy-rsa/2.0/vars
```

将一下内容修改为自己的值

```
export KEY_COUNTRY="US"
export KEY_PROVINCE="NY"
export KEY_CITY="New York"
export KEY_ORG="Organization Name"
export KEY_EMAIL="administrator@example.com"
export KEY_CN=droplet.example.com
export KEY_NAME=server
export KEY_OU=server
```

修改完成之后，清空vars

```
cd /etc/openvpn/easy-rsa/2.0
source ./vars
```

完成之后可以开始修改密钥了

在/etc/openvpn/easy-rsa/2.0 路径下执行

```
./clean-all
./build-ca
```

之后为 Server 生成密钥

```
./build-key-server server
```

然后生成 Diffie Hellman key exchange 文件

```
./build-dh
```

接着将这四个文件复制到 OpenVPN 的配置目录中

```
cd /etc/openvpn/easy-rsa/2.0/keys
cp dh1024.pem ca.crt server.crt server.key /etc/openvpn
```

接着就可以在/etc/openvpn/easy-rsa/2.0目录中生成客户端

Certifications和Keys了

```
./build-key client
```

之后，需要配置一下路由以及启动 Server 了

首先需要配置一下防火墙，由于 CentOS7 中 iptables 已经被

Firewalld 取代，故需要使用 Firewalld

```
systemctl status firewalld.service
```

然后允许 openvpn 服务通过

```
firewall-cmd --add-service openvpn
firewall-cmd --permanent --add-service openvpn
```

接着添加masquerade

```
firewall-cmd --add-masquerade
firewall-cmd --permanent --add- masquerade
```

接着要允许IP转发

```
vim /etc/sysctl.conf
```

在这里边添加以下内容

```
net.ipv4.ip_forward = 1
```

然后启动服务，并添加自启动项

```
sysctl -p
systemctl start openvpn@server
systemctl enable openvpn@server
```

3.2 在客户端配置

取回之前在/etc/openvpn/easy-rsa/2.0/keys 里边的三个文件
ca.crt
client.crt
client.key

然后创建一个配置文件：client.ovpn

```
client
dev tun
proto udp
remote xxx.xxx.xxx.xxx 1194
resolv-retry infinite
nobind
persist-key
persist-tun
comp-lzo
verb 3
ca ca.crt
cert client.crt
key client.key
```

3.3 运行结果

通过搭建的 VPN，可以正常访问 Google ， Youtube 等网站

4. 实验收获

4.1 TCP 与 UDP

为什么 OpenVPN 首选了 UDP 而不是 TCP？主要是因为在有大流量的情况下，TCP 隧道的性能将严重下降，最终导致网络不可用，这一点 CIPE 的作者已经通过实测证明了。不过 OpenVPN 还是支持 TCP 的。

4.2 SSL 协议问题

为什么 OpenVPN 使用 SSL 的握手协议而不需要 SSL 的记录协议？SSL 首选了 TCP，而 OpenVPN 首选了 UDP，设计时 DTLS 还没有产生，故 OpenVPN 不能直接使用 SSL 记录协议，但是 SSL 的握手协议是良好的，故 OpenVPN 将 SSL 握手协议和自己的记录协议适配在了一起，适配的方法是增加了一相当于“多路复用码”的感觉，通过对一些字段的解析，识别协议包是握手包还是记录包。

4.3 SSL 记录协议与 OpenVPN 记录协议差别

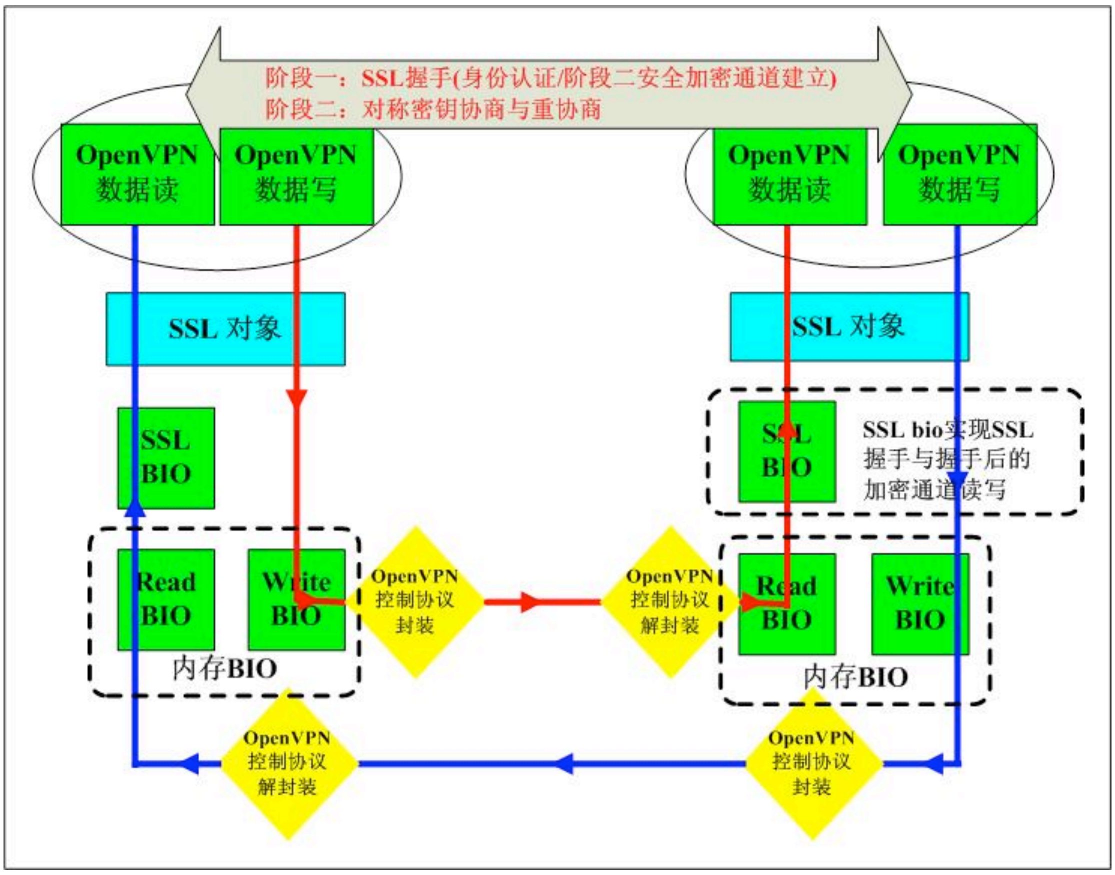
SSL 直接接于传输层协议之上，它在保证安全的同时一定不能更改应用的语义，比如 http 是 tcp 的，那么 https 也需要是 tcp 的，SSL 原本就是设计用于加密应用数据而不是构建三层隧道的。而 OpenVPN 的记录协议保护的只是一个 IP 数据报的再封装，最终的端系统应用才负责传输层语义，因此 OpenVPN 不需要可靠传输。

4.4 SSL 握手协议与 OpenVPN 握手协议

OpenVPN 的连接建立使用了 SSL 握手协议，可是却抓不到 SSL 握手包，实际上 SSL 握手协议是作为一种载荷封装在了 OpenVPN 的协议包里面了，SSL 握手协议数据实际上是写入了一块内存，然后 OpenVPN 从该内存中读出这些数据包，然后封装后通过 reliable 层发出，数据到达对端后，解封装后写入一块内存，然后 SSL 从内存中读出 SSL 握手载荷，一切都是通过 BIO 实现的

4.5 BIO 封装 SSL 握手消息

之前说过 OpenVPN 通过 BIO 封装的 SSL 握手协议，那么他是如何去封装的，在这里用一副简要的图来展示：



5. 附 VPN 一键安装程序源码（针对 Linux 系统）

由于 OpenSSL 需要专门的客户端，在使用的时候并不方便，故我后来又专门搭建了基于 Ikev2 的 VPN 环境，目前大部分系统平台均直接支持，一下为自动搭建源码，来源于 Github：

```
#!
/bin/b
ash
          PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
          export PATH
          #========================================================================
          ========================
          #   System Required:  CentOS6.x/7 (32bit/64bit) or Ubuntu
          #   Description:  Install IKEV2 VPN for CentOS and Ubuntu
          #   Author: quericy
          #   Intro:  https://quericy.me/blog/699
          #========================================================================
          ========================
          clear
          VER=1.2.0
          echo "#############################################################"
          echo "# Install IKEV2 VPN for CentOS6.x/7 (32bit/64bit) or Ubuntu or
          Debian7/8.*"
          echo "# Intro: https://quericy.me/blog/699"
          echo "#"
          echo "# Author:quericy"
          echo "#"
          echo "# Version:$VER"
          echo "#############################################################"
          echo ""


          __INTERACTIVE=""
          if [ -t 1 ] ; then
              __INTERACTIVE="1"
          fi


          __green(){
              if [ "$__INTERACTIVE" ] ; then
                  printf '\033[1;31;32m'
              fi
              printf -- "$1"
              if [ "$__INTERACTIVE" ] ; then
                  printf '\033[0m'
              fi
          }


          __red(){
              if [ "$__INTERACTIVE" ] ; then
                  printf '\033[1;31;40m'
              fi
              printf -- "$1"
              if [ "$__INTERACTIVE" ] ; then
                  printf '\033[0m'
              fi
          }


          __yellow(){
              if [ "$__INTERACTIVE" ] ; then
                  printf '\033[1;31;33m'
              fi
              printf -- "$1"
              if [ "$__INTERACTIVE" ] ; then
                  printf '\033[0m'
              fi
          }


          # Install IKEV2
          function install_ikev2(){
              rootness
              disable_selinux
              get_system
              yum_install
              get_my_ip
              pre_install
              download_files
              setup_strongswan
              get_key
              configure_ipsec
              configure_strongswan
              configure_secrets
              SNAT_set
              iptables_check
              ipsec restart
              success_info
          }


          # Make sure only root can run our script
          function rootness(){
          if [[ $EUID -ne 0 ]]; then
             echo "Error:This script must be run as root!" 1>&2
             exit 1
          fi
          }


          # Disable selinux
          function disable_selinux(){
          if [ -s /etc/selinux/config ] && grep 'SELINUX=enforcing'
          /etc/selinux/config; then
```

```
    sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config
    setenforce 0
fi
}


# Ubuntu or CentOS
function get_system(){
    if grep -Eqi "CentOS" /etc/issue || grep -Eq "CentOS" /etc/*-release;
then
        system_str="0"
    elif  grep -Eqi "Ubuntu" /etc/issue || grep -Eq "Ubuntu" /etc/*-release;
then
        system_str="1"
    elif  grep -Eqi "Debian" /etc/issue || grep -Eq "Debian" /etc/*-release;
then
        system_str="1"
    else
        echo "This Script must be running at the CentOS or Ubuntu or Debian!"
        exit 1
    fi
}


#install necessary lib
function yum_install(){
    if [ "$system_str" = "0" ]; then
    yum -y update
    yum -y install pam-devel openssl-devel make gcc curl
    else
    apt-get -y update
    apt-get -y install libpam0g-dev libssl-dev make gcc curl
    fi
}


# Get IP address of the server
function get_my_ip(){
    echo "Preparing, Please wait a moment..."
    IP=`curl -s checkip.dyndns.com | cut -d' ' -f 6  | cut -d'<' -f 1`
    if [ -z $IP ]; then
        IP=`curl -s ifconfig.me/ip`
    fi
}
```

```
# Pre-installation settings
function pre_install(){
    echo "#############################################################"
    echo "# Install IKEV2 VPN for CentOS6.x/7 (32bit/64bit) or Ubuntu or
Debian7/8.*"
    echo "# Intro: https://quericy.me/blog/699"
    echo "#"
    echo "# Author:quericy"
    echo "#"
    echo "# Version:$VER"
    echo "#############################################################"
    echo "please choose the type of your VPS(Xen、KVM: 1  ,  OpenVZ: 2):"
    read -p "your choice(1 or 2):" os_choice
    if [ "$os_choice" = "1" ]; then
        os="1"
        os_str="Xen、KVM"
        else
            if [ "$os_choice" = "2" ]; then
                os="2"
                os_str="OpenVZ"
                else
                echo "wrong choice!"
                exit 1
            fi
    fi
    echo "please input the ip (or domain) of your VPS:"
    read -p "ip or domain(default_value:${IP}):" vps_ip
    if [ "$vps_ip" = "" ]; then
        vps_ip=$IP
    fi


    echo "Would you want to import existing cert? You NEED copy your cert
file to the same directory of this script"
    read -p "yes or no?(default_value:no):" have_cert
    if [ "$have_cert" = "yes" ]; then
        have_cert="1"
    else
        have_cert="0"
        echo "please input the cert country(C):"
        read -p "C(default value:com):" my_cert_c
        if [ "$my_cert_c" = "" ]; then
```

```
            my_cert_c="com"
        fi
        echo "please input the cert organization(O):"
        read -p "O(default value:myvpn):" my_cert_o
        if [ "$my_cert_o" = "" ]; then
            my_cert_o="myvpn"
        fi
        echo "please input the cert common name(CN):"
        read -p "CN(default value:VPN CA):" my_cert_cn
        if [ "$my_cert_cn" = "" ]; then
            my_cert_cn="VPN CA"
        fi
    fi


    echo "####################################"
    get_char(){
        SAVEDSTTY=`stty -g`
        stty -echo
        stty cbreak
        dd if=/dev/tty bs=1 count=1 2> /dev/null
        stty -raw
        stty echo
        stty $SAVEDSTTY
    }
    echo "Please confirm the information:"
    echo ""
    echo -e "the type of your server: [$(__green $os_str)]"
    echo -e "the ip(or domain) of your server: [$(__green $vps_ip)]"
    if [ "$have_cert" = "1" ]; then
        echo -e "$(__yellow "These are the certificate you MUST be
prepared:")"
        echo -e "[$(__green "ca.cert.pem")]:The CA cert or the chain cert."
        echo -e "[$(__green "server.cert.pem")]:Your server cert."
        echo -e "[$(__green "server.pem")]:Your  key of the server cert."
        echo -e "[$(__yellow "Please copy these file to the same directory
of this script before start!")]"
    else
        echo -e "the cert_info:[$(__green "C=${my_cert_c},
O=${my_cert_o}")]"
    fi
    echo ""
    echo "Press any key to start...or Press Ctrl+C to cancel"
    char=`get_char`
    #Current folder
    cur_dir=`pwd`
    cd $cur_dir
}




# Download strongswan
function download_files(){
    strongswan_version='strongswan-5.5.1'
    strongswan_file="$strongswan_version.tar.gz"
    if [ -f $strongswan_file ];then
        echo -e "$strongswan_file [$(__green "found")]"
    else
        if ! wget --no-check-certificate
https://download.strongswan.org/$strongswan_file;then
            echo "Failed to download $strongswan_file"
            exit 1
        fi
    fi
    tar xzf $strongswan_file
    if [ $? -eq 0 ];then
        cd $cur_dir/$strongswan_version/
    else
        echo ""
        echo "Unzip $strongswan_file failed! Please visit
https://quericy.me/blog/699 and contact."
        exit 1
    fi
}


# configure and install strongswan
function setup_strongswan(){
    if [ "$os" = "1" ]; then
        ./configure  --enable-eap-identity --enable-eap-md5 \
--enable-eap-mschapv2 --enable-eap-tls --enable-eap-ttls --enable-eap-peap
\
--enable-eap-tnc --enable-eap-dynamic --enable-eap-radius
--enable-xauth-eap  \
--enable-xauth-pam  --enable-dhcp  --enable-openssl  --enable-addrblock
--enable-unity  \
--enable-certexpire --enable-radattr --enable-swanctl --enable-openssl
```

```
--disable-gmp


    else
        ./configure  --enable-eap-identity --enable-eap-md5 \
--enable-eap-mschapv2 --enable-eap-tls --enable-eap-ttls --enable-eap-peap
\
--enable-eap-tnc --enable-eap-dynamic --enable-eap-radius
--enable-xauth-eap  \
--enable-xauth-pam  --enable-dhcp  --enable-openssl  --enable-addrblock
--enable-unity  \
--enable-certexpire --enable-radattr --enable-swanctl --enable-openssl
--disable-gmp --enable-kernel-libipsec


    fi
    make; make install
}


# configure cert and key
function get_key(){
    cd $cur_dir
    if [ ! -d my_key ];then
        mkdir my_key
    fi
    if [ "$have_cert" = "1" ]; then
        import_cert
    else
        create_cert
    fi


    echo "####################################"
    get_char(){
        SAVEDSTTY=`stty -g`
        stty -echo
        stty cbreak
        dd if=/dev/tty bs=1 count=1 2> /dev/null
        stty -raw
        stty echo
        stty $SAVEDSTTY
    }
    cp -f ca.cert.pem /usr/local/etc/ipsec.d/cacerts/
    cp -f server.cert.pem /usr/local/etc/ipsec.d/certs/
    cp -f server.pem /usr/local/etc/ipsec.d/private/
    cp -f client.cert.pem /usr/local/etc/ipsec.d/certs/
    cp -f client.pem  /usr/local/etc/ipsec.d/private/
    echo "Cert copy completed"
}


# import cert if user has ssl certificate
function import_cert(){
   cd $cur_dir
   if [ -f ca.cert.pem ];then
        cp -f ca.cert.pem my_key/ca.cert.pem
        echo -e "ca.cert.pem [$(__green "found")]"
    else
        echo -e "ca.cert.pem [$(__red "Not found!")]"
        exit
    fi
    if [ -f server.cert.pem ];then
        cp -f server.cert.pem my_key/server.cert.pem
        cp -f server.cert.pem my_key/client.cert.pem
        echo -e "server.cert.pem [$(__green "found")]"
        echo -e "client.cert.pem [$(__green "auto create")]"
    else
        echo -e "server.cert.pem [$(__red "Not found!")]"
        exit
    fi
    if [ -f server.pem ];then
        cp -f server.pem my_key/server.pem
        cp -f server.pem my_key/client.pem
        echo -e "server.pem [$(__green "found")]"
        echo -e "client.pem [$(__green "auto create")]"
    else
        echo -e "server.pem [$(__red "Not found!")]"
        exit
    fi
    cd my_key
}


# auto create certificate
function create_cert(){
    cd $cur_dir
    cd my_key
    ipsec pki --gen --outform pem > ca.pem
    ipsec pki --self --in ca.pem --dn "C=${my_cert_c}, O=${my_cert_o},
CN=${my_cert_cn}" --ca --outform pem >ca.cert.pem
    ipsec pki --gen --outform pem > server.pem
    ipsec pki --pub --in server.pem | ipsec pki --issue --cacert ca.cert.pem
\
    --cakey ca.pem --dn "C=${my_cert_c}, O=${my_cert_o}, CN=${vps_ip}" \
    --san="${vps_ip}" --flag serverAuth --flag ikeIntermediate \
    --outform pem > server.cert.pem
    ipsec pki --gen --outform pem > client.pem
    ipsec pki --pub --in client.pem | ipsec pki --issue --cacert ca.cert.pem
--cakey ca.pem --dn "C=${my_cert_c}, O=${my_cert_o}, CN=VPN Client"
--outform pem > client.cert.pem
    echo "configure the pkcs12 cert password(Can be empty):"
    openssl pkcs12 -export -inkey client.pem -in client.cert.pem -name
"client" -certfile ca.cert.pem -caname "${my_cert_cn}"  -out
client.cert.p12
}


# configure the ipsec.conf
function configure_ipsec(){
 cat > /usr/local/etc/ipsec.conf<<-EOF
config setup
    uniqueids=never
conn iOS_cert
    keyexchange=ikev1
    fragmentation=yes
    left=%defaultroute
    leftauth=pubkey
    leftsubnet=0.0.0.0/0
    leftcert=server.cert.pem
    right=%any
    rightauth=pubkey
    rightauth2=xauth
    rightsourceip=10.31.2.0/24
    rightcert=client.cert.pem
    auto=add
conn android_xauth_psk
    keyexchange=ikev1
    left=%defaultroute
    leftauth=psk
    leftsubnet=0.0.0.0/0
    right=%any
    rightauth=psk
    rightauth2=xauth
    rightsourceip=10.31.2.0/24
    auto=add
conn networkmanager-strongswan
    keyexchange=ikev2
    left=%defaultroute
    leftauth=pubkey
    leftsubnet=0.0.0.0/0
    leftcert=server.cert.pem
    right=%any
    rightauth=pubkey
    rightsourceip=10.31.2.0/24
    rightcert=client.cert.pem
    auto=add
conn ios_ikev2
    keyexchange=ikev2
    ike=aes256-sha256-modp2048,3des-sha1-modp2048,aes256-sha1-modp2048!
    esp=aes256-sha256,3des-sha1,aes256-sha1!
    rekey=no
    left=%defaultroute
    leftid=${vps_ip}
    leftsendcert=always
    leftsubnet=0.0.0.0/0
    leftcert=server.cert.pem
    right=%any
    rightauth=eap-mschapv2
    rightsourceip=10.31.2.0/24
    rightsendcert=never
    eap_identity=%any
    dpdaction=clear
    fragmentation=yes
    auto=add
conn windows7
    keyexchange=ikev2
    ike=aes256-sha1-modp1024!
    rekey=no
    left=%defaultroute
    leftauth=pubkey
    leftsubnet=0.0.0.0/0
    leftcert=server.cert.pem
    right=%any
    rightauth=eap-mschapv2
    rightsourceip=10.31.2.0/24
    rightsendcert=never
    eap_identity=%any
    auto=add
EOF
}


# configure the strongswan.conf
function configure_strongswan(){
 cat > /usr/local/etc/strongswan.conf<<-EOF
 charon {
        load_modular = yes
        duplicheck.enable = no
        compress = yes
        plugins {
                include strongswan.d/charon/*.conf
        }
        dns1 = 8.8.8.8
        dns2 = 8.8.4.4
        nbns1 = 8.8.8.8
        nbns2 = 8.8.4.4
}
include strongswan.d/*.conf
EOF
}


# configure the ipsec.secrets
function configure_secrets(){
    cat > /usr/local/etc/ipsec.secrets<<-EOF
: RSA server.pem
: PSK "myPSKkey"
: XAUTH "myXAUTHPass"
myUserName %any : EAP "myUserPass"
EOF
}


function SNAT_set(){
    echo "Use SNAT could implove the speed,but your server MUST have static
ip address."
    read -p "yes or no?(default_value:no):" use_SNAT
    if [ "$use_SNAT" = "yes" ]; then
        use_SNAT_str="1"
        echo -e "$(__yellow "ip address info:")"
        ip address | grep inet
        echo "Some servers has elastic IP (AWS) or mapping IP.In this case,you
should input the IP address which is binding in network interface."
        read -p "static ip or network interface ip (default_value:${IP}):"
static_ip
    if [ "$static_ip" = "" ]; then
        static_ip=$IP
    fi
    else
        use_SNAT_str="0"
    fi
}


# iptables check
function iptables_check(){
    cat > /etc/sysctl.d/10-ipsec.conf<<-EOF
net.ipv4.ip_forward=1
EOF
    sysctl --system
    echo "Do you use firewall in CentOS7 instead of iptables?"
    read -p "yes or no?(default_value:no):" use_firewall
    if [ "$use_firewall" = "yes" ]; then
        firewall_set
    else
        iptables_set
    fi
}


# firewall set in CentOS7
function firewall_set(){
    if ! systemctl is-active firewalld > /dev/null; then
        systemctl start firewalld
    fi
    firewall-cmd --permanent --add-service="ipsec"
    firewall-cmd --permanent --add-port=500/udp
    firewall-cmd --permanent --add-port=4500/udp
    firewall-cmd --permanent --add-masquerade
    firewall-cmd --reload
}
```

```
# iptables set
function iptables_set(){
    echo -e "$(__yellow "ip address info:")"
    ip address | grep inet
    echo "The above content is the network card information of your VPS."
    echo "[$(__yellow "Important")]Please enter the name of the interface
which can be connected to the public network."
    if [ "$os" = "1" ]; then
            read -p "Network card interface(default_value:eth0):" interface
        if [ "$interface" = "" ]; then
            interface="eth0"
        fi
        iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
        iptables -A FORWARD -s 10.31.0.0/24  -j ACCEPT
        iptables -A FORWARD -s 10.31.1.0/24  -j ACCEPT
        iptables -A FORWARD -s 10.31.2.0/24  -j ACCEPT
        iptables -A INPUT -i $interface -p esp -j ACCEPT
        iptables -A INPUT -i $interface -p udp --dport 500 -j ACCEPT
        iptables -A INPUT -i $interface -p tcp --dport 500 -j ACCEPT
        iptables -A INPUT -i $interface -p udp --dport 4500 -j ACCEPT
        iptables -A INPUT -i $interface -p udp --dport 1701 -j ACCEPT
        iptables -A INPUT -i $interface -p tcp --dport 1723 -j ACCEPT
        #iptables -A FORWARD -j REJECT
        if [ "$use_SNAT_str" = "1" ]; then
            iptables -t nat -A POSTROUTING -s 10.31.0.0/24 -o $interface -j
SNAT --to-source $static_ip
            iptables -t nat -A POSTROUTING -s 10.31.1.0/24 -o $interface -j
SNAT --to-source $static_ip
            iptables -t nat -A POSTROUTING -s 10.31.2.0/24 -o $interface -j
SNAT --to-source $static_ip
        else
            iptables -t nat -A POSTROUTING -s 10.31.0.0/24 -o $interface -j
MASQUERADE
            iptables -t nat -A POSTROUTING -s 10.31.1.0/24 -o $interface -j
MASQUERADE
            iptables -t nat -A POSTROUTING -s 10.31.2.0/24 -o $interface -j
MASQUERADE
        fi
    else
        read -p "Network card interface(default_value:venet0):" interface
        if [ "$interface" = "" ]; then
            interface="venet0"
        fi
        iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
        iptables -A FORWARD -s 10.31.0.0/24  -j ACCEPT
        iptables -A FORWARD -s 10.31.1.0/24  -j ACCEPT
        iptables -A FORWARD -s 10.31.2.0/24  -j ACCEPT
        iptables -A INPUT -i $interface -p esp -j ACCEPT
        iptables -A INPUT -i $interface -p udp --dport 500 -j ACCEPT
        iptables -A INPUT -i $interface -p tcp --dport 500 -j ACCEPT
        iptables -A INPUT -i $interface -p udp --dport 4500 -j ACCEPT
        iptables -A INPUT -i $interface -p udp --dport 1701 -j ACCEPT
        iptables -A INPUT -i $interface -p tcp --dport 1723 -j ACCEPT
        #iptables -A FORWARD -j REJECT
        if [ "$use_SNAT_str" = "1" ]; then
            iptables -t nat -A POSTROUTING -s 10.31.0.0/24 -o $interface -j
SNAT --to-source $static_ip
            iptables -t nat -A POSTROUTING -s 10.31.1.0/24 -o $interface -j
SNAT --to-source $static_ip
            iptables -t nat -A POSTROUTING -s 10.31.2.0/24 -o $interface -j
SNAT --to-source $static_ip
        else
            iptables -t nat -A POSTROUTING -s 10.31.0.0/24 -o $interface -j
MASQUERADE
            iptables -t nat -A POSTROUTING -s 10.31.1.0/24 -o $interface -j
MASQUERADE
            iptables -t nat -A POSTROUTING -s 10.31.2.0/24 -o $interface -j
MASQUERADE
        fi
    fi
    if [ "$system_str" = "0" ]; then
        service iptables save
    else
        iptables-save > /etc/iptables.rules
        cat > /etc/network/if-up.d/iptables<<-EOF
#!/bin/sh
iptables-restore < /etc/iptables.rules
EOF
        chmod +x /etc/network/if-up.d/iptables
    fi
}


# echo the success info
function success_info(){
    echo "#############################################################"
    echo -e "#"
    echo -e "# [$(__green "Install Complete")]"
    echo -e "# Version:$VER"
    echo -e "# There is the default login info of your IPSec/IkeV2 VPN Service"
    echo -e "# UserName:$(__green " myUserName")"
    echo -e "# PassWord:$(__green " myUserPass")"
    echo -e "# PSK:$(__green " myPSKkey")"
    echo -e "# you should change default username and password in$(__green
" /usr/local/etc/ipsec.secrets")"
    echo -e "# you cert:$(__green " ${cur_dir}/my_key/ca.cert.pem ")"
    if [ "$have_cert" = "1" ]; then
    echo -e "# you don't need to install cert if it's be trusted."
    else
    echo -e "# you must copy the cert to the client and install it."
    fi
    echo -e "#"
    echo -e
"#################################################################"
    echo -e ""
}


# Initialization step
install_ikev2
```