前面的文章：Linux下用GRE隧道直接联通两个私网里讲了怎样用在两个独立的私网间打洞(GRE tunnel)来直接连通两个不同的私网，进阶需求就是希望能把这个tunnel加密一下。否则，有心的坏人可能会在外部线路上窃听到很多敏感的信息。因此，本文将要讲的，算是前一文章的进阶部分：**给打洞的隧道加密**！

Why "GRE over IPSec"

光是说加密隧道的话，方案有好几种：

- IPSec tunnel
- GRE over IPSec
- IPSec over GRE
- .....

我们为什么选用**GRE over IPSec**呢？
跟单纯的**IPSec tunnel**比，优势在于：

- 方案更灵活，我们可以灵活的把要加密的流量路由到GRE隧道上
- 而且IPSec不支持多播，像OSPF或其他高大上的路由协议没法玩儿

跟**IPSec over GRE**比：

- 更安全。整个上公网的流量都是加密的，但从外部根本都不知道跑的是GRE协议。

接下来，我们主要就是要讲的方案是：**GRE over IPSec**

环境

对象	备注
NETA	10.0.0.0/24的一个私网，网关是GWA
NETB	10.0.1.0/24的一个私网，网关是GWB
NETC	节点都能互通的一个网络，可以认为是公网
GWA	ip:10.0.0.1(NETA)、1.1.1.1(NETC)，CentOS6.x
GWB	ip:10.0.1.1(NETB)、2.2.2.2(NETC)，CentOS6.x
greB	GWA上的虚拟网络接口，GRE隧道名
greA	GWB上的虚拟网络接口，GRE隧道名
eth0	GWA连NETA、GWB连NETB的网络设备名
eth1	GWA和GWB连NETC的网络设备名

具体步骤

准备工作

分别在GWA和GWB两台机上执行：

```
yum -y install libreswan iptables;
rm -rf /etc/ipsec.d/*db; # 删除原有自带的db文件
ipsec initnss; # 重新初始化db
chkconfig ipsec on; # 这一步也可以后面再做
```

```
checconfig ipsec on; # 这一步也可以后面再做
```

正常打洞

GWA上执行：

```
cat << EOF | tee /etc/sysconfig/network-scripts/ifcfg-greB
DEVICE=greB
ONBOOT=yes
TYPE=GRE
PEER_OUTER_IPADDR=2.2.2.2
PEER_INNER_IPADDR=10.0.1.0/24
MY_OUTER_IPADDR=1.1.1.1
MY_INNER_IPADDR=10.0.0.1
KEY=http://haw-haw.org
BOOTPROTO=none
EOF

ifup greB;
```

同样，在GWB上执行：

```
cat << EOF | tee /etc/sysconfig/network-scripts/ifcfg-greA
DEVICE=greA
ONBOOT=yes
TYPE=GRE
PEER_OUTER_IPADDR=1.1.1.1
```

```
PEER_INNER_IPADDR=10.0.0.0/24
MY_OUTER_IPADDR=2.2.2.2
MY_INNER_IPADDR=10.0.1.1
KEY=http://haw-haw.org
BOOTPROTO=none

EOF

ifup greA;
```

这样，其实打洞就基本上已经完成了，现在从NETA和NETB的网络里随便找两台机器，都应该能互通了。

GRE over IPSEC

配置GWA

在GWA上执行：

```
ipsec newhostkey \
--configdir /etc/ipsec.d \
--random /dev/urandom \
--output /etc/ipsec.d/GWA.secrets \
--verbose;
# 上面的"--random /dev/urandom"的参数比用缺省的效率要高很多！
ipsec showhostkey --left;
# 记下输出中“leftrsasigkey=”这一行
# 这将用于本机(GWA)的/etc/ipsec.d/greB.conf文件中
ipsec showhostkey --right;
```

```
ipsec showhostkey --right
# 记下输出中"rightrsasigkey="这一行
# 这将用于对端机器(GWB)的/etc/ipsec.d/greA.conf文件中
vim /etc/ipsec.d/greB.conf # 建立配置文件greB.conf
```

内容如下：

```
conn greB
    type=transport
    left=10.0.0.1
    leftrsasigkey=......
    leftprotoport=gre
    right=10.0.1.1
    rightrsasigkey=......
    rightprotoport=gre
    authby=rsasig
    auto=start
```

注意：

- 这里的greB是随便取的，只是因为GWA上的隧道设备名为greB，所以就沿用了这个名字
- leftrsasigkey=是来自于上面*ipsec showhostkey --left*命令
- rightrsasigkey=是来自于GWB上执行命令*ipsec showhostkey --right*的结果

配置GWB

依葫芦画瓢，在GWB上执行：

```
ipsec newhostkey \
--configdir /etc/ipsec.d \
--random /dev/urandom \
--output /etc/ipsec.d/GWB.secrets \
--verbose;
ipsec showhostkey --left;
# 记下输出中“leftrsasigkey=”这一行
# 这将用于本机(GWB)的/etc/ipsec.d/greA.conf文件中
ipsec showhostkey --right;
# 记下输出中“rightrsasigkey=”这一行
# 这将用于对端机器(GWA)的/etc/ipsec.d/greB.conf文件中
vim /etc/ipsec.d/greA.conf
# 建立配置文件greA.conf，因为tunnel设备名叫greA
```

内容如下：

```
conn greA
    type=transport
    left=10.0.1.1
    leftrsasigkey=......
    leftprotoport=gre
    right=10.0.0.1
    rightrsasigkey=......
    rightprotoport=gre
    authby=rsasig
```

```
    auto=start
```

注意：

- 这里的greA是随便取的，只是因为GWB上的隧道设备名为greA，所以就沿用了这个名字
- leftrsasigkey=是来自于上面*ipsec showhostkey --left*命令
- rightrsasigkey=是来自于GWA上执行命令*ipsec showhostkey --right*的结果

Iptables

本来在前面正常GRE打洞测步骤里其实也有iptables相关设置，这里就都整合到这一部分统一说了

在GWA和GWB上分别执行：

```
iptables -A INPUT -i eth1 -p gre -j ACCEPT;
iptables -A INPUT -i eth1 -p udp \
    -m state --state NEW \
    -m udp \
    -m multiport --dports 50,51,500,4500 \
    -j ACCEPT;
iptables -A INPUT -i eth1 -p tcp \
    -m state --state NEW \
    -m tcp \
    -m multiport --dports 50,51 \
    -j ACCEPT;
iptables -t mangle -A FORWARD \
```

```
    -p tcp -m tcp --tcp-flags SYN,RST SYN \
    -j TCPMSS --clamp-mss-to-pmtu;
/etc/init.d/iptables save; # 将iptables规则存入配置文件
```

启动服务

分别在GWA和GWB的机器上执行：

```
/etc/init.d/ipsec start;
ipsec auto --add greB; # 仅GWA上执行
ipsec auto --up greB; # 仅GWA上执行
ipsec auto --add greA; # 仅GWB上执行
ipsec auto --up greA; # 仅GWB上执行
chkconfig ipsec on;
# 如果前面没有执行这句的这里执行一下，
# 以后ipsec就会随机器启动起来，
# 而且不再需要ipsec auto --add和ipsec auto --up了
```

简单测试

要检验是否成功设置，可以分别在GWA和GWB上听包：

```
tcpdump -nn -i eth1 host 1.1.1.1 and host 2.2.2.2;
```

会发现包都是ESP加密过的了。

然后分别在GWA和GWB上干掉ipsec对greB和greA的加密干掉

```
ipsec auto --delete greB; # 仅在GWA上执行
ipsec auto --delete greA; # 仅在GWB上执行
```

然后再重复上面的听包命令：

```
tcpdump -nn -i eth1 host 1.1.1.1 and host 2.2.2.2;
```

会发现加密包没有了，取而代之是GRE包，而且明显能看到GRE包里封装的内容。