# Setting Up a VPN into Amazon's Public Cloud VPC

**Amazon Web Services' VPC** (Virtual Private Cloud) is somewhat inconvenient for developers. The standard way to access it is through an IPsec "hardware VPN". In practice this means having to deal not just with IPsec, but also with BGP. This document simplifies things by using strongSwan to access the VPC instances. Neither hardware nor BGP are required.

## Scenario

We assume a single VPC subnet with Internet access (i.e., located behind an Internet Gateway). We have a small number of clients accessing the VPC remotely, all running on Linux machines. I believe the solution can be tweaked to allow for larger deployments. For example, you will want to replace preshared key authentication by certificate-based authentication to support a large number of clients.

## Solution Overview

We create a new, dedicated instance serving as a VPN gateway for the whole VPC. The solution uses tunnel-mode IPsec with IKEv2 and a virtual IP pool. For simplicity, we use preshared keys rather than certificates. strongSwan is deployed on both client and gateway.

## General Warnings

- Debugging IPsec is hard. Debugging networking on public cloud virtual machines is hard. Please **follow these instructions carefully**.
- Amazon's cloud is constantly changing, mostly for the better. This document has been validated with the current feature set, as of today (May 2012). It may not be valid tomorrow.

## Solution Steps

1. Create a new VPC instance (the minimal instance type in VPC is `m1.small`). This will become your VPN Gateway. We have used standard Ubuntu images, (Oneirc `ami-a562a9cc` and Precise `ami-a29943cb`). Instead of launching a new, dedicated instance, you can reuse an existing instance but that would be much less secure. Note that the VPN Gateway instance can be stopped when not in use, and later restarted.
2. Disable source/destination check on the VPN Gateway instance. (Right click on the instance in the Amazon Console. The option is in the pop up menu.)
3. Assign an Elastic IP for the instance. This will be the VPN gateway's public address, but first we will use it to access the gateway to install strongSwan.
4. Install strongSwan on the gateway (and on your client, too). We have used the version available in the repository, 4.5.2. Modify the configuration files per the next section.
5. Enable IP forwarding on the gateway (you need to do *both* of the following):
    1. Edit `/etc/sysctl.conf` and uncomment the line `net.ipv4.ip_forward=1`. The next time the system reboots, it will load these settings.
    2. For the current session, run `sysctl -p` to apply the changes to the running system.
6. Define the gateway's security group(s) to allow incoming TCP/22, UDP/500 and UDP/4500.
    - Once the gateway is fully set up, you will be able to disable TCP/22 in the security group and tunnel SSH through IPsec instead of directly accessing the VPN gateway.
7. Define a subnet for the virtual IP pool in ipsec.conf. It doesn't need to be inside the VPC. In our example the VPC encompasses 10.10.0.0/16, and the virtual IP pool will be drawn from 10.100.0.0/16.
8. The strongSwan Gateway will assign addresses for IPsec clients from the virtual address pool. But it needs a bit of routing help: First, note the instance ID of the VPN gateway. Then locate the routing table associated with the subnet of protected instances (this may or may not be the main routing table), and add a routing rule that routes all traffic destined to the pool's subnet (10.100.0.0/16) through the VPN gateway.
9. Allow any incoming traffic from the pool's subnet into all VPC instances. For example, by adding an "all traffic" rule to the `default` security group of your VPC.
10. For each instance that is in the VPC, disable the instance's Source/Dest. check (from the EC2 Instances page). **Edit:** this is only required for step **~~#4~~** in the Bonus section below. Do not make this change unless you implement the guest instance routing-table changes. **Edit2:** exception, you need to disable source/destination check (on affected instances; namely all instances in the routing path; ex: VPN <--> proxy instances) if you want to do any kind of policy based routing because this spoofing check will supersede OS level spoofing check and you will not be able to catch the packet loss through tshark or tcpdump.

1. Finally, `sudo ipsec restart` on the gateway and the client, and you are good to go!

## Bonus

The only job of Amazon's NAT instance is to run a single iptables rule. You can deploy that rule on your VPN gateway and save the price of a dedicated NAT instance.

1. Add the "masquerade" NAT rule for the entire VPC: `sudo iptables --table nat --append POSTROUTING --source 10.10.0.0/16 -j MASQUERADE`
2. Save the iptables configuration so that it will survive a reboot.
    - RHEL/CentOS machines
        - `/sbin/service iptables save`
        - This will write the rules to /etc/sysconfig/iptables.
    - Debian/Ubuntu machines
        - Install iptables-persistent, if not already installed: `apt-get install iptables-persistent`
        - This during install it will prompt to save current rules. Say yes, to have it create /etc/iptables/rules.v4 and /etc/iptables/rules.v6 for you.
        - In the future, after making changes, run `iptables-save > /etc/iptables/rules.v4` to save.
3. Change the EC2 routing table so that the VPN gateway (rather than the NAT instance) becomes the default route of your private subnet.
4. As an alternative to the previous step, update the /etc/network/interfaces file on your non-accessible VPC instances to use the VPN gateway as their default route:

```
# static routes
up route del -net 0.0.0.0/0 gw 10.0.10.1 dev eth0
up route add -net 0.0.0.0/0 gw 10.0.10.10 dev eth0
```

**Edit:** changing the routing table on a guest is **discouraged by Amazon**. Therefore the preceding method (modification of the EC2 routing table) is preferred.
**Edit2:** you can't change default routing (<VPC CIDR block> to local) but you can certainly add your custom routing in the instance (or console) on top of the default ones (ex: if you want to do policy based routing). In VPC private IP doesn't change across reboot so adding routing rules should be safe (you don't use elastic IP to do routing). Amazon console's routing changes supersede OS level routing (I am guessing they happen at higher up router level). In the console you are able to specify <VPC CIDR block>. In a VPC instance your are residing under VPC CIDR block - in a subnet. Static routing would be for advanced users under VPC instance (there is a potential to lock yourself out; from the console Amazon guarantees you can't remove the default routing so you can't lock yourself out).

## Debugging

These instructions didn't work for me out of the box, because I set my network up a little differently. To debug, I found it useful to do the following:

- Ping the IP address of the next closest interface.
- If you don't get replies, you need to find out what's wrong - if there is a wrong route, IP Tables rule, or something else. Use tshark to capture and display ICMP packets (replace 192.168.254.10 with the IP address of the host you are interested in):
  - To capture:

    ```
    sudo tshark -f "host 192.168.254.10" -i eth0 -w /tmp/capture.cap
    ```

  - To display:

    ```
    sudo tshark -R "icmp and host 192.168.254.10" -r /tmp/capture.cap
    ```

- You may need to install tshark on the client, VPN gateway, and VPC box to get a good view of what is happening.
- if you are doing for example policy based routing use this to see if mac addresses are changed correctly (or not changed at all, which would indicate routing issues). Because policy based routing triggers spoofing violation make sure you disable these checks everywhere it matters (source and destination servers). You will not see packets drop in sniff tools, I can't stress this enough, all you will see is packet not going somewhere but not packet loss at destination server (nothing will show up in sniff tool) so it will be difficult to determine the reason (hardware/software router, OS, and 3rd party interception software like a proxy can drop packets):
  - To verify routing changes (check source/destination mac addresses are correct):

    ```
    tcpdump -eqni eth0 'port 80' , if you routed port 80 traffic to a remote proxy server do this on both VPN/Proxy servers
    ```

## Configuration Files

### /etc/ipsec.conf on the Client

```
# ipsec.conf - strongSwan IPsec configuration file

# basic configuration

config setup
    # nat_traversal=yes
    charonstart=yes
    plutostart=no
    # charondebug="ike 2, knl 2, cfg 2, mgr 2, chd 2, net 2"

# Connections into AWS VPC
conn %default
    ikelifetime=60m
    keylife=20m
    rekeymargin=3m
    keyingtries=1
    keyexchange=ikev2
    authby=secret

conn us-east-1-vpc
    left=%any
    leftsourceip=%config
    leftid=<my-email-address>
    leftfirewall=yes
    right=<gateway's elastic IP>
    rightsubnet=10.10.0.0/16
    rightid=@us-east-gw.example.com
```

```
    auto=start

# Add connections here.

# include /var/lib/strongswan/ipsec.conf.inc
```

### /etc/ipsec.secrets on the Client

```
us-east-gw.example.com : PSK "aa82c7a776e2175114213acc02dda9951a6bc25deb433e6d5d6ef7058626c589"
```

### /etc/ipsec.conf on the Gateway

```
# ipsec.conf - strongSwan IPsec configuration file

# basic configuration

config setup
    # nat_traversal=yes
    charonstart=yes
    plutostart=no
    # charondebug="ike 2, knl 2, cfg 2, mgr 3, chd 2, net 2"

# /etc/ipsec.conf - strongSwan IPsec configuration file

conn %default
    ikelifetime=60m
    keylife=20m
    rekeymargin=3m
    keyingtries=1
    keyexchange=ikev2
    authby=secret

conn client
    # The leftid parameter is not a real DNS name
    leftid=us-east-gw.example.com
    # The "left" parameter is the gateway's private IP
    left=10.10.0.10
    # We are protecting the entire VPC, not just this subnet
    leftsubnet=10.10.0.0/16
    leftfirewall=yes
    right=%any
    # The virtual IP pool is outside the VPC!
    rightsourceip=10.100.255.0/28
    auto=add

# Add connections here.

# include /var/lib/strongswan/ipsec.conf.inc
```

### /etc/ipsec.secrets on the Gateway

```
<my-email-address> : PSK "aa82c7a776e2175114213acc02dda9951a6bc25deb433e6d5d6ef7058626c589"
```