Max Manders

a bit of dev, a bit of ops

(/)

AWS VPN Solutions with StrongSWAN (/2014/05/05/aws-vpn-solutions-with-strongswan.html)

May 05, 2014

Overview

In this article, I'll discuss how to connect two or more VPCs which are in different AWS regions. Using static routing, routes cannot be re-advertised and as such it's not possible to write routing tables that allow a packet to traverse from one VPC to another via an intermediary VPC, where all VPCs are connected by IPSec tunnels. Because of this limitation, if you want complete connectivity between all peered regions, there must exist an IPSec tunnel between each and every region. Of course, if you want to join VPCs within the same region, you can use the recently released VPC Peering (http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-peering.html) feature. Pay particular attention to the documentation about unsupported configurations (http://docs.aws.amazon.com/AmazonVPC/latest/PeeringGuide/invalid-peering-configurations.html).

Introduction

Connecting AWS VPCs across regions is not currently supported using the recently released VPC Peering feature. VPC Peering only allows you to connect VPCs within the *same* region. If you want to securely connect VPCs across different regions, you have two options

- run an AWS VPN in one VPC and connect it to a software based IPSec solution on an instance in another region
- run a software based IPSec solution on an instance in each region and connect the two instances

The two most popular software based IPSec solutions for Linux seem to be FreeS/WAN and its fork strongSwan. I've opted for strongSwan on the basis of what I believe to be better documentation, and the fact that it is still under active development. In order to terminate a tunnel to an AWS VPN, I found that strongSwan 5.1 is required. If you're using Ubuntu, you'll either need to upgrade to Ubuntu 14.04, or use an appropriate backports repository; versions of Ubuntu prior to 14.04 only provide strongSwan 4.5.

Assumptions

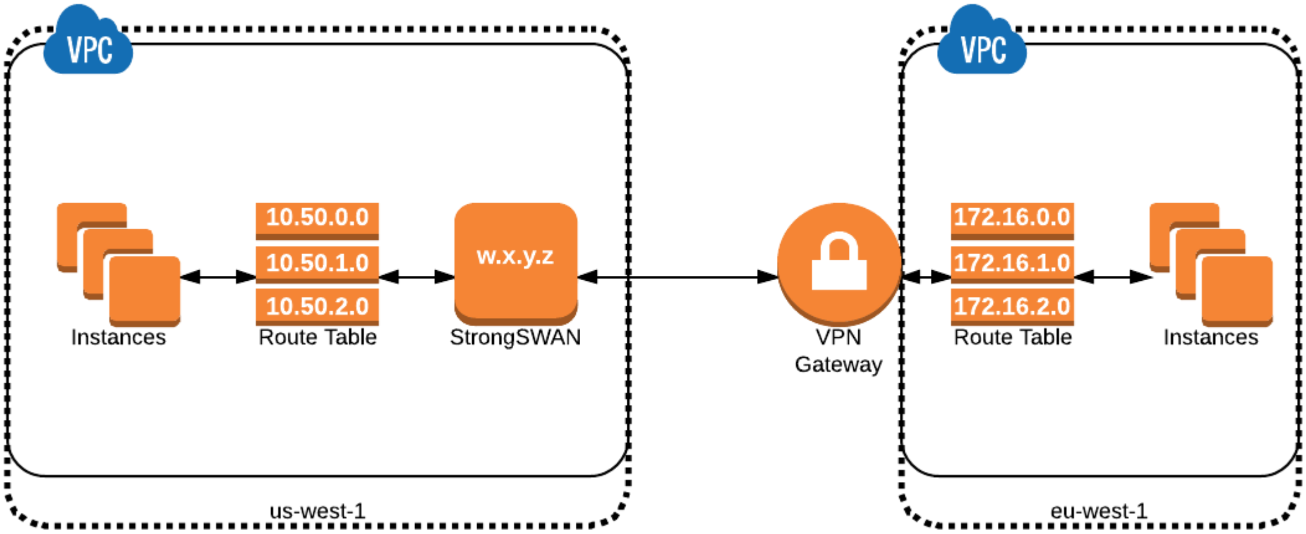
This article assumes a certain familiarity with AWS. Specifically I assume that you have two VPCs in different regions, with the following configurations

- non-overlapping networks in each VPC
- public (and optionally private) subnets
- a routing table to route internet traffic for your public subnets via

an Internet Gateway

- a routing table to route internet traffic for your private subnets via a public NAT instance
- a public NAT instance
- appropriate security groups and attachments

Scenario 1: Connect VPC_1 with VPC_2 using strongSwan and AWS VPN



Step 1: Configure a strongSwan server in VPC_1

For this discussion, we shall assume that VPC_1 is located in *us-west-1* and has a network block of `172.50.0.0/16`.

1. Launch a new `m1.small` instance in a public subnet of your VPC
   - Choose an Ubuntu 14.04 x86_64 AMI
   - For now, ensure that your Security Groups are configured to permit SSH from your local workstation. *We'll need to reconfigure them later once both endpoints have been created.*
   - Attach a new EIP to the instance
   - Right-click the instance and disable the source/destination check to allow this server to act as a router

2. Install strongSwan and its dependencies:

   ```
   sudo apt-get install strongswan
   ```

3. Enable IP forwarding by doing the following as root

   ```
   echo 1 > /proc/sys/net/ipv4/ip_forward && sudo sysctl -p
   ```

Step 2: Create AWS VPN in VPC_2

Having created a strongSwan instance with a known public IP address, it's time to create the other end of our tunnel in another VPC. For this discussion, we shall assume that VPC_2 is located in *eu-west-1* and has a network block of `172.60.0.0/16`.

1. Navigate to the VPC Dashboard in the AWS Console
2. Make sure you are in the correct region
3. Select the 'Virtual Private Gateways' menu item
4. Click 'Create Virtual Private Gateway'
   - Give your new VGW a name
5. Select 'Yes, Create'
6. Once your VGW has been created, select it, and then click 'Attach to VPC'
   - Select the target VPC from the VPC drop-down list
7. Select the 'Customer Gateways' menu item
8. Click 'Create Customer Gateway'
   - Give your CGW a name
   - Set routing to *Static*
   - Set the public IP address of the remote end of the VPN connection, i.e. the EIP of the strongSwan instance.
9. Click 'Yes, Create'

10. Select the 'VPN Connections' menu item
11. Click 'Create VPN Connection'
    - Name: Give your VPN connection a name
    - Virtual Private Gateway: Select the VGW you created in step 4
    - Customer Gateway: Select 'Existing' and choose the CGW you created in step 8
    - Routing Options: Static
    - Static IP Prefixes: Set the CIDR block of your VPC you wish to make available over the VPN tunnel, e.g. 172.60.0.0/16 for the entire VPC network
12. Click 'Yes, Create'

Once the VPN has been created, right-click on the entry and select the option to download the configuration file. Select the generic configuration option. This will save a text file to your local workstation with the information that will be needed to configure the strongSwan instance in the other region.

Step 3: Update strongSwan Security Groups

Now that we have both ends of the tunnel created, we need to ensure that they can talk to each other. At this stage, you should have the public IP addresses for the AWS VPN from the configuration file you downloaded earlier. You should also have the public IP address of the strongSwan instance. Now, we'll create a Security Group (or modify an existing one) to contain rules to permit the public traffic necessary to set up the IPSec tunnel. Configure your security group as per the table below.

Type	Protocol	Port Range	Source
Custom UDP Rule	UDP	4500	AWS VPN Tunnel 1 Public IP
Custom UDP Rule	UDP	500	AWS VPN Tunnel 1 Public IP
All Traffic	All	All	CIDR of Remote VPC
All Traffic	All	All	CIDR of Local VPC

You don't need to do anything particular on the AWS side to permit traffic; this was addressed implicitly behind the scenes when you created the Customer Gateway earlier on. You'll just need to make sure that any security groups on either side of the tunnel explicitly allow traffic from the opposite network.

Step 4: Configure Routing

So, our tunnel end points can talk to each other from a Security Group perspective, but we need to tell our VPCs how to use the tunnel when it's up. That means changes to one or more routing tables. If you have a functioning VPC, as described in the assumptions section above, then you should have two different flavours of routing table. Both should have a routing table that like the one below; the second row will only be present for private subnets, while the third row will only be present for public subnets.

Destination	Target
VPC CIDR	local
0.0.0.0/0	eni-XXX
0.0.0.0/0	igw-XXX

We need to add one more route to our routing tables in each region to say how to hop across the tunnel into the other network. For VPC_1, which has the strongSwan instance, adding the new route will result in the table below, substituting the ENI ID of the strongSwan instance's primary ethernet adapter.

Destination	Target
⋮	⋮
VPC_2 CIDR	eni-XXX

For VPC_2, which has an AWS VPN, adding the new route will result in the table below, substituting the VGW ID of the VGW you created earlier.

Destination	Target
⋮	⋮
VPC_1 CIDR	vgw-XXX

Step 5: Configure strongSwan

At this point, we have an endpoint in each region, between which we can connect our IPSec tunnel. We have configured Security Groups to permit the passing of IPSec along the tunnel. Lastly, we have configured our routing table so that each VPC knows how to direct traffic destined for the other network. The AWS VPN is configured and ready to go, we just need to configure IPSec on the strongSwan instance.

Firstly, we'll need to edit `/var/lib/strongswan/ipsec.conf.inc` to add a pre-shared key. This key can be found in the AWS VPN config file we downloaded earlier. Look under the IPSec Tunnel #1 heading for *Pre-Shared Key*. We'll also need the Outside IP Address for the Virtual Private Gateway. It's worth noting that within any region in the AWS network, the public Virtual Private Gateway IP addresses for the two tunnels are always the same. For our purposes, the secrets file should look like:

```
87.238.85.42 : PSK "put_your_psk_here"
```

Now we need to configure the IPSec tunnel, by editing `/etc/ipsec.conf`

```
config setup

conn %default
        ikelifetime=28800s
        keylife=3600s
        rekeymargin=3m
        keyingtries=3
        authby=secret
        keyexchange=ikev2
        mobike=no

conn name_of_connection
        left={private_ip_of_instance}
        leftsubnet={local_vpc_CIDR}
        leftid=name_of_connection
        right=87.238.85.42
        rightsubnet={remote_vpc_CIDR}
        dpdaction=restart
        auto=route

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

Step 6: Start the Tunnel

You should now be able to restart the IPSec service (`service strongswan restart`) and send traffic over the tunnel to the remote VPC, and vice versa. You can look at the tunnel status in both the AWS console, as well as using the `ipsec status name_of_connection` command.

Summary

Hopefully you have been able to follow the steps above to bring up your own IPSec tunnel between two different VPCs in two different regions. You may have noticed that AWS VPNs provide two tunnels for redundancy. AWS may perform maintenance on one of the tunnels from time to time, and having two established tunnels allows you to keep communicating over the VPN during these maintenance windows. Unfortunately I have had trouble implementing two tunnels via strongSwan, but I'm working on it. I'll post back when I have more information.

In the next post, we'll look at a few small changes that can be made that will allow you to create an IPSec tunnel between two strongSwan instances, without the need for an AWS VPN tunnel.

Categories: aws (/categories/aws.html)
Tags: aws (/tags/aws.html), networking (/tags/networking.html), ipsec (/tags/ipsec.html), vpn (/tags/vpn.html), strongswan (/tags/strongswan.html), linux (/tags/linux.html)

0 Comments Max Manders 1 Login

Recommend Share Sort by Best

Start the discussion…

Be the first to comment.

ALSO ON **MAX MANDERS** WHAT'S THIS?

Using New And Old AWS CLI Tools Together

3 comments • 2 years ago

aughban — I've always prefered using the sdk's in development consoles like irb or pry. Lets me due fun stuff like: asg.groups.select {|g| g.name

My Puppet VimRC

1 comment • 21 days ago

Owen Morgan — I do find myself missing the configuration file tips shared over tea and the low barrier between desks. Time for me to look

Subscribe Add Disqus to your site Add Disqus Add Privacy

About Me

I'm (/about) an Edinburgh based ops/dev engineer working for FanDuel (http://www.fanduel.com). An AWS (https://aws.amazon.com) aficianado and recovering PHP developer, I've learned to love Ruby and Chef (http://www.getchef.com). I'm a fan of open source software (http://en.wikipedia.org/wiki/Open_source) and open-data (http://en.wikipedia.org/wiki/Open_data). I live in Edinburgh, UK with my wife Jo (http://twitter.com/mrsjmanders) and our two cats, Ziggy and Maggie. I'm a co-organiser of Whisky Web (http://www.whiskyweb.co.uk) and an amateur trombonist.

[...] (/about)

- Pinboard (http://www.pinboard.in/u:maxmanders/)
- Flickr (http://www.flickr.com/photos/maxmanders)
- Last.FM (http://www.last.fm/user/mmanders)
- Twitter (http://www.twitter.com/maxmanders)
- Github (http://www.github.com/maxmanders)
- LinkedIn (https://uk.linkedin.com/in/maxmanders)
- RSS (http://maxmanders.co.uk/feed/rss.xml)

Contact

If you want to get touch, you can send an email to my-first-name [at] this domain.