

Technical Approach.

Pictured above is a high level view of how everything works. Our VPN client program starts, configures, and then connects to the VPN server. The VPN server works by rewriting routing tables to push traffic along certain routes. That is done by OpenVPN server and client (and not by our program). Both server side and client side routing tables need to be rewritten so that traffic is funneled from the client, through the VPN server’s gateway, and then to the internet and back. This way the client is only communicating with the VPN server, and the internet only sees the VPN server communicating with it. We make use of OpenVPN’s client and server programs for this. To add hops, we start a new Linux server instance and configure the server for IP forwarding. This means the server will act as a router and forward all traffic on a certain port to another IP address. The very first hop server that we create will always point to the VPN server, and each following hop server will point to the one in front of it. Finally, we will use OpenVPN’s client to connect to the last hop server created, and the traffic will be funneled through each hop, to the running OpenVPN server, to the internet and back the same route.

A more detailed explanation of the code, and how this works:

We wanted the program to be completely autonomous. Everything needed will be created by our program. None of the routes will be static or preset-up, and everything will be created dynamically by the program. This way the user will not have to do anything but run the program. No complex set-up of keys, certificates, or anything that may be a hassle for the end user. In order to do this, we first need to hard code a key pair into the program that will be used for configuring the instances later on. We can import the key pair to the user’s AWS along with a created security group to allow access on the instance through our desired ports. After creating these two things, we can now create the instances of bare bones Linux servers and link them to the created security group and key pair. The instances created will be exactly the same, but be configured differently for the hops and for the VPN server. First we will configure the VPN server. OpenVPN makes use of SSL certificates and keys for authorization, so this is hard coded into the program. The program creates a folder on the local machine where it stores all the needed certificates and keys. We will need to make use of SCP to put these files onto our VPN server. This program makes use of PSCP to put the files onto the server. Then, using shell’s SSH capabilities, the program sends remote commands to the server to configure it for IP forwarding, which is necessary to funnel the traffic. Finally for the server, we start the system service of OpenVPN server. Now that the server is active and listening on our designated port, we could use our VPN client to connect to it, and everything will work, but we now want to add hops. This is done by simply starting up however many instances of hops we want, and then sending a remote command to enable and configure IP forwarding on each hop. Configuration of IP forwarding on each hop is done with just two rules. First, the pre routing rule will be a static route that will always point to the hop in front of it (or the VPN server if it is the first instance). The post routing rule will be a MASQUERADE NAT rule witch means to dynamically translate the return address back to the client that requested it. Now that the OpenVPN server is active and listening, and the hops are all pointing to the adjacent IPs, we can connect to the last hop created in order to funnel the traffic all the way through. Our program generates the needed SSL certificates for the client side, and an OpenVPN configuration file witch tells the client where to connect to. The IP address in the configuration file will be the last hop created, or the OpenVPN server if there are no hops. Then our program runs the client program with the created configuration file, and if you go to check your IP at a site like whatismyip.com, you will see that the IP address of the VPN server is showing. This is because like the diagram up top shows, only the VPN server has direct access to the internet. Each machine only has access to the machine in front of and behind it. All traffic is pushed along this route.

Finally, the disconnect button will terminate the OpenVPN client on the client machine, terminate all running instances on AWS, wait for them to be stopped, and then delete the created security groups and key pairs.