To: Steve VanDevender
From: Jerry Xie
Subject: Final Writeup for the Final Project
Date: Aug 7, 2019

Summary and Motivation

This project is intended to deploy a self-maintained VPN server that can be connected from any modern personal computing devices such as laptops and phones. For security reasons, I switch my VPN server from one cloud provider to another monthly. It is tedious to repeat the VPN server installation and configuration each time. Though I have been using a dockerized OpenConnect Server container for a while, and it is smooth and robust, however, not every type of Virtual Private Server, or VPS, from cloud providers support Docker. Typically, one needs a Kernel-based Virtual Machine, or KVM, type of VPS to run Docker, and this kind of VPS is more expensive than others. In order to save money, for this project, I would like to try using Puppet to deploy a VPN server, which means it could be deployed on most of cheap OpenVZ VPS. For a better consistency with the class, this project is going to be deployed on Amazon Web Services, or AWS.

Absolute Objectives

- Deploy a VPN server that is using any one of the following technology:
    - OpenConnect
    - Softether VPN
    - OpenVPN
- The VPN server is deployed on AWS.
- The VPN server can be connected from devices with iOS and macOS.
- Each VPN user has his own VPN user account credentials to connect.
- After the successful deployment, adding a new VPN user account should cost less than 5 minutes.

Optional Objectives

- Using Puppet to deploy and manage the VPN server.

Objectives Achievement

All objectives mentioned above have been implemented in the final project. The VPN technology I am using is OpenConnect due to its compatibility with Cisco's AnyConnect Protocol, which means one could easily connect to the VPN server with Cisco AnyConnect Clients across most of platforms such as Windows, macOS, iOS, and Android. This project has been tested fully functional with servers running on Ubuntu 18.04 and clients running on iOS and macOS.

Use Cases

A. Day-to-day usage on laptops/phones:
   Preconditions: I am connecting to a Wi-Fi network that is connected to the Internet; I have the VPN server's address and my user account credentials; the AnyConnect VPN software has been correctly installed on my systems.
    1. I simply use the AnyConnect VPN client to connect to the VPN server.

   PostConditions: After the successful connection, when I visit https://whoer.net/, I should be able to see my VPS's IP address instead of my actual IP address.

B. VPN user accounts updating usage:
   Preconditions: I want to perform an update, deletion, for example, to a VPN account on the VPN server. I have the root access to the VPS hosting the VPN server, and I know the VPN account name I wanted to delete.
    1. I use SSH to connect to the VPS.

2. I could find a text file stating all the VPN user account credentials in the corresponding puppet module directory.
3. I found the corresponding User resource definition in the file, changed “ensure => present” to “ensure => absent”, and save the file.

PostConditions: Puppet would automatically apply new changes in less than 5 minutes, or I could apply new changes manually. The user I wanted to delete is now no longer able to use the VPN service.

Deliverables

- A functional VPN server on AWS.
- A puppet module that deploys and manages the VPN server.

Deliverables Achievement

Both of the deliverables are achieved.

Typical user population

For this project, the user population is mostly me, but it would benefit anyone with the need to use VPN services. Since Puppet is capably to automatically deploy and manage the VPN server and Puppet can be run on most of modern OS, it would be flexible for consumers like me to switch VPN servers from one infrastructure provider to another in minutes at a lower cost. I have already used code from this project to deploy my VPN on AWS, and it is as smooth as using the prebuilt Docker container.

Possible Security Issue

Since, in all of the three VPN solutions, the user account credentials are hashed in the config file, and the traffic between the client and server is encrypted, the only security breach is the administrator of the VPS. If we assume the administrator is a good citizen who does not take notes of VPN users’ credentials and keeps the software in the VPS up-to-date, there should be no catastrophic security issue for this project. Since, after connecting to the VPN server, the users replace their trust in their Internet Service Provider, or ISP, with the trust in the VPN service provider, the users either trust the VPN server and its administrators or rely on encrypted information transfer protocols such as SSH and HTTPS. In this way, it would be difficult for users’ data to be compromised.

Life Cycle and Longevity

Puppet can ensure the package needed to run the VPN server updated. However, the administrator still needs to maintain config files such as certificates and user-account-management-text-file from time to time since certificates expire and accounts need to be updated. Other than these, the whole system for this project should be robust to use for years.

Documentation

An administrator’s documentation and a programmer’s documentation are attached to this report.

Project Plan and Timeline

- August 4 - August 5: Be familiar with building three VPN servers manually and pick one of them is the most feasible to be implemented for this project in one week.
- August 6 - August 11: Try integrating the deployment and management procedures with Puppet.

User's Documentation

Getting the Cisco AnyConnect Software

For Windows and macOS users, refer to https://service.uoregon.edu/TDClient/KB/ArticleDet?ID=31471; for iOS and Android users, search "AnyConnect" with the App Store or Play Store.

Connecting to My Demo Server

The following would demonstrate how to connect to the demo server on macOS, connection on other platforms should be very similar:

1. Allow connections to untrusted server, see Figure-1.(Skip this step if the server is using a certificate f rom a trusted authority)

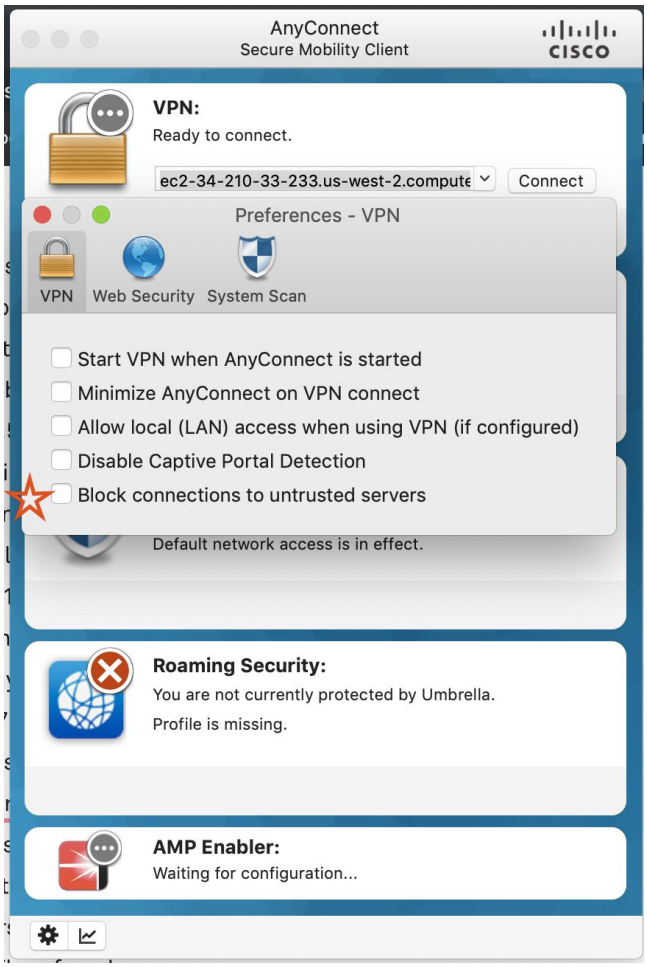

Figure-1

2. Copy-paste the following server address to the input field and click connect: “ec2-34-210-33-233.us-west-2.compute.amazonaws.com”; the instance would be terminated in a week after the class ended.
3. If you are using Keychain, the following window might be prompted, see Figure-2; click “Deny”:

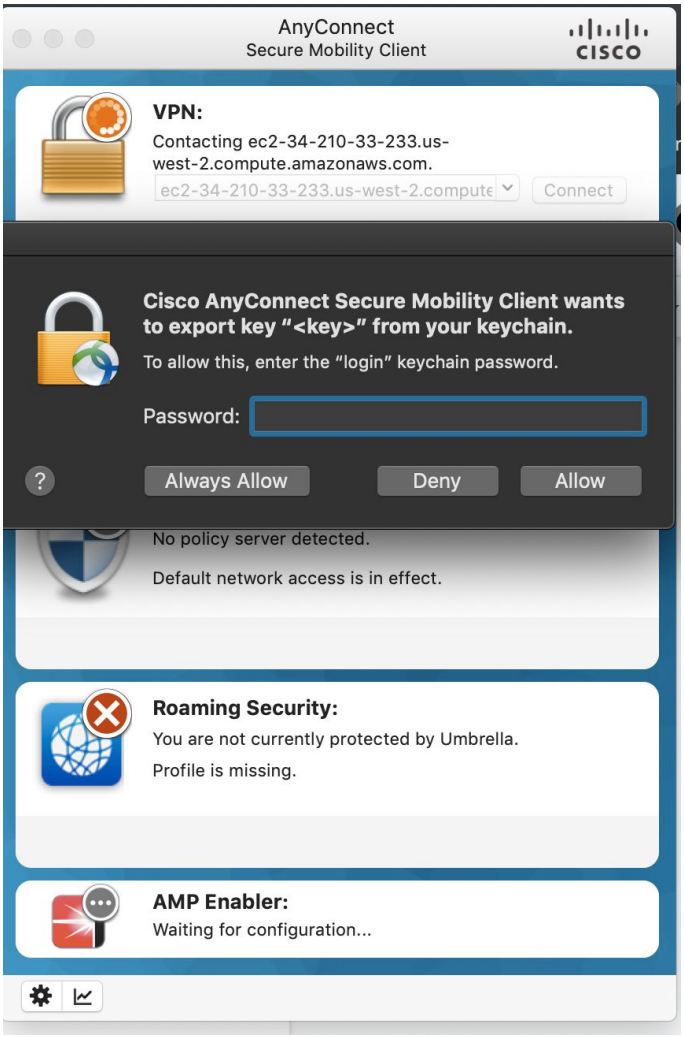

Figure-2

4. If the server does not have a trusted certificate from an authority, the following window will be prompted, see Figure-3; do not worry, the connection is still encrypted, click “Connect Anyway”:

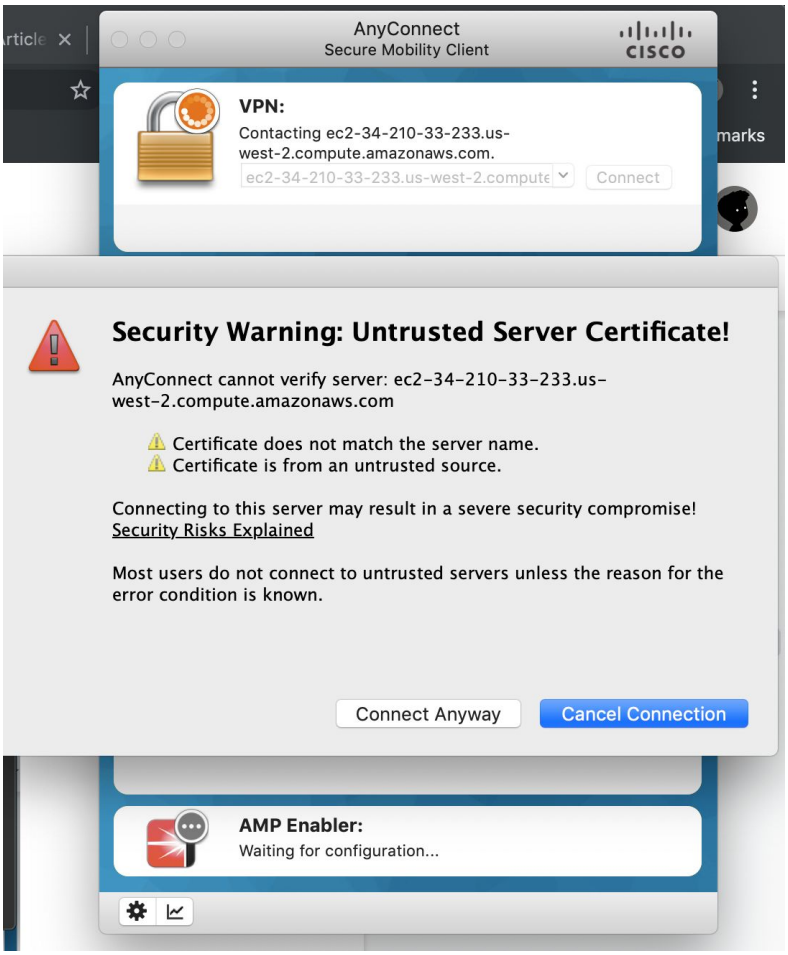

Figure-3

5. The client would then prompt for your account username and password; for my instructor Steve, the username is “stevev_vpn” and password is “thisisaweakpassforsteve”.
6. If everything goes well, you should be able to connect to the VPN server now, see Figure-4.

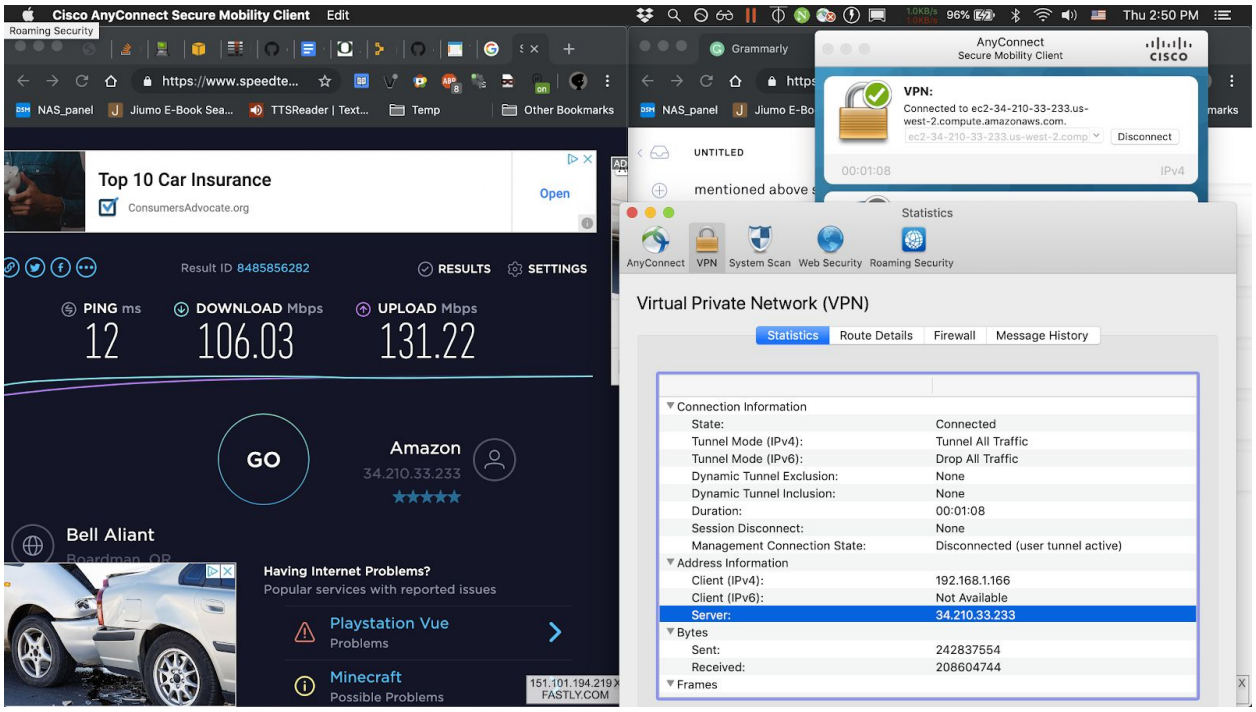
Cisco AnyConnect Secure Mobility Client
Edit
Roaming Security
Thu 2:50 PM
Top 10 Car Insurance
ConsumersAdvocate.org
Open
Result ID 8485856282
RESULTS
SETTINGS
PING ms
12
DOWNLOAD Mbps
106.03
UPLOAD Mbps
131.22
GO
Amazon
34.210.33.233
Bell Aliant
Having Internet Problems?
Popular services with reported issues
Playstation Vue
Problems
Minecraft
Possible Problems
151.101.194.219
FASTLY.COM
AnyConnect
Secure Mobility Client
VPN:
Connected to ec2-34-210-33-233.us-west-2.compute.amazonaws.com.
Disconnect
00:01:08
IPv4
UNTITLED
Statistics
AnyConnect
VPN
System Scan
Web Security
Roaming Security
Virtual Private Network (VPN)
Statistics
Route Details
Firewall
Message History
Connection Information
State: Connected
Tunnel Mode (IPv4): Tunnel All Traffic
Tunnel Mode (IPv6): Drop All Traffic
Dynamic Tunnel Exclusion: None
Dynamic Tunnel Inclusion: None
Duration: 00:01:08
Session Disconnect: None
Management Connection State: Disconnected (user tunnel active)
Address Information
Client (IPv4): 192.168.1.166
Client (IPv6): Not Available
Server: 34.210.33.233
Bytes
Sent: 242837554
Received: 208604744
Frames

Figure-4

Administrator's Documentation

Server System Requirement

The project was tested and fully functional on EC2 running on Ubuntu 18.04. There is no guarantee that this software is able or unable to run on any other Operating System.

Prerequisite

- The server should have Git installed and is capable of pulling the source code from Github.
- The server should have Puppet with version 5.4.0 or higher installed to deploy the project.
- Make sure the server's 443-port is not occupied.

Installation

1. Change the current directory to "/etc/puppet" by using the command "cd /etc/puppet".
2. Remove all the file in this directory by using the command "rm -rf *".
3. Pull this project's source code into this directory by using the command "git clone https://github.com/cis399-2019-team/team9-puppet ./".
4. Let Puppet do its job by using the command "puppet apply manifests/site.pp".

Verification of the Installation

If there is no error message printing out when running the last command from the installation section, the VPN service is now up and running. You could verify this by using the command "systemctl -all list-sockets" and see if there is at least one "ocserv.service" listening to the 443-port.

VPN User Management

The file “vpn_users.pp”, in the directory /etc/puppet/code/modules/ocserv/manifests/, defines all the VPN user. I configured the OpenConnect Server to handle password authentication through Pluggable Authentication Modules, or PAM, which allows you to use Ubuntu system accounts to login from VPN clients. The following introduces how to add and delete a user to the system by manipulating the “vpn_users.pp” file.

Addition

1. Ensure the package “openssl” is installed to hash the plaintext password.
2. Use command “openssl passwd -1” and input a desired password for this new user in order to generate a hash for this password.
3. Append a user definition inside the ocserv::vpn_users class.
4. The user definition should be declared similar to the following template:

```
user { '<New user’s name>':#do not forget the quotation marks
    ensure => present,
    managehome => false,
    comment => 'Puppet managed VPN user',
    password => '<Hash for the desired password>’,#do not forget the
quotation marks
    groups => ['vpn-users'],
  }
```

5. After saving the file, wait for 5 minutes to let Puppet automatically apply the change, or do it manually by using the command “puppet apply /etc/puppet/manifests/site.pp”

   You may find the documentation of the User type from Puppet’s website, https://puppet.com/docs/puppet/5.3/types/user.html, useful if you want more control over the VPN user.

Deletion

1. Find the targeted user definition block in the file.
2. In the block, change the line “ensure => present,” to “ensure => absent,”.
3. After saving the file, wait for 5 minutes to let Puppet automatically apply the change, or do it manually by using the command “puppet apply /etc/puppet/manifests/site.pp”

Optional Certificate Configuration

This project shipped with a preconfigured private and self-signed certificate, thus users need to disable the functionality in Cisco AnyConnect Client that blocks connections from the untrusted server. You might want to make your own private key and self-signed certificate on your server, or even better if using a certificate from a trusted authority. Make sure they are in PEM format though. After that, you would need to put your private key and certificate into /etc/puppet/code/modules/ocserv/files/, and rename the private key to “server-key.pem” and the certificate to “server-cert.pem”. As usual, wait for Puppet to apply changes automatically or apply changes manually.

Programmer's Documentation

This documentation would demonstrate the structure of the project and shows possible ways to improve the code for anyone with interests. The following would be the outline of the project:

1. The firewall including the module firewall, my_fw, and stdlib.
2. The ocserv module.
3. The resolvconf module.

I am going to elaborate these three parts one by one.

The firewall including the module firewall, my_fw, and stdlib

The firewall and stdlib module are from the official puppet forge website; they work more like a library which would be called in other modules. Basically, I use it to interact with the iptables on Linux.

The my_fw module essentially purges the iptables and establish basic rules such as allowing incoming ICMP traffic, HTTP(s) traffic, and drop all other incoming traffic.

The ocserv module

This module configs and installs everything needed to fire up the OpenConnect VPN server. It firstly inserts a Masquerade rule, or MASQ, to the iptable to allow connected VPN users invisibly access the Internet via it as the MASQ gateway. Then it installs the ocserv package and injects preconfigured *.conf files, key and certificate into the system.

The resolvconf module

Technically, the modules mentioned above should suffice. However, since Ubuntu 18.04 uses Systemd-resolve to replace Dnsmasq, the OS by default listens to 127.0.0.53 for DNS

query. After applying the firewall rules mentioned in the ocserv module, specifically the MASQ rule, the query to 127.0.0.53 is dropped even if I have inserted rules to accept any traffic with a destination of 127.0.0.53/32. I have spent hours searching for a solution, but unfortunately it seems like there is currently no better solution rather than adding public DNS servers to the /etc/resolv.conf. Therefore, I use the resolvconf module to install resolvconf package in order to append two public DNS servers address to /etc/resolv.conf.