**Final Project – Communication Systems**

**Introduction to Cloud**

1. Register with a free tier account on AWS (please let me know if you have issues doing this).
2. Create 2 EC2 instance that are running your favorite version of Linux. EC2 instances are effectively virtual servers that are hosted in the AWS cloud.
3. One EC2 instance will be used as a client and the other will be used as a server.
4. Make sure you are able to ssh into both new EC2 instances.
5. Make sure you update your EC2 instance using yum or apt (update/upgrade).

**Web Server**

Create a python webserver. SSH into your EC2 server and set up your python webserver.

You can use the following as a good reference example:

<https://www.codexpedia.com/python/python-web-server-for-get-and-post-requests/>

Modify this example in the following ways:

* Port should be 80.
* self.send\_header content-type should be “application/json”.
* Modify do\_POST. Users will be using this to send a message to the server. It should support a POST request like: http://< EC2 server IP address > with the following data fields:

sender=<from\_user>

receiver=<to\_user>

message=<message>

The message will need to be stored. Use sqlite or mysql to store the messages.

* Modify do\_GET. Users will be using this to fetch messages from the server. It should support a GET request like: http://<EC2 server IP address>?user=<username>.

This GET request should “write” a JSON response containing all the messages that have been sent to this user. Access the sqlite or mysql database to fetch the messages and put it into JSON format. The JSON format can be as follows:

Example:

{“response”: {

“user”: “<user>”,

"messages": [

{“sender”: “<sender>”, "value": "<message1>"},

{“sender”: “<sender>”, "value": "<message2>"},

{“sender”: “<sender>”, "value": "<message3>"},

]

}}

Make sure your python web server is running. It is required for the other parts of this project.

**Web Client**

Create a python web client. SSH into your EC2 client and set up your python client

This reference may be useful:

<https://www.datacamp.com/community/tutorials/making-http-requests-in-python>

A user should be able to run the web client from command line (i.e. python webclient.py). It should take a user name as an argument. This user name will be used for the GET and POST messages.

When the python program is run, it should fetch all messages from the server you created using a GET request. It should parse the JSON and display the messages that have been sent to this user. The output format should be “(sender) message\n”.

A prompt should then be shown for the user, allowing the user to enter a command. There should be two commands that are supported: refresh, send, quit

* Format for refresh command: refresh

Refresh should display new messages only.

* Format for send command: send:<to\_user>:<msg>

Send should send the message in the POST request to your web server.

* Format for quit command: quit

Quit should exit your program

A command prompt should be displayed after executing each command.

Run this client from your EC2 client. You will need to use two terminals. SSH from each terminal into the same EC2 client. Run the web client python program using different user names for each instance to simulate a chat (i.e. user1 and user2). You will need a third ssh terminal to capture the network traffic (using tcpdump) as you send 2 messages from each client. If your program is working correctly, user1 and user2 should be able to chat using your program.

**Submit your .pcap file.**

**Perform “curl ipinfo.io” and record results.**

**Virtual Private Network (VPN)**

Create a VPN client and server. The VPN client will run on your EC2 client. The VPN server will run on your EC2 server. OpenVPN is the recommended VPN software to use. You can use the following reference to help build the necessary components: <https://github.com/Nyr/openvpn-install>. The VPN client and server will create an encrypted tunnel that safely protects your network traffic as it travels between the client and server.

Run your chat program again while the VPN client and server are running. Perform a network capture on the same interface (not the VPN’s tun interface). Notice that the traffic should be encrypted and should no longer be in plain text. This demonstrates how VPN’s can provide security through encryption for your network traffic.

**Submit your .pcap file.**

**Perform “curl ipinfo.io” and record results.**

**Turn off the VPN client and server after you are done so it does not impact the next section.**

**Anonymous Communications**

Create a Tor client and server. Tor is a network anonymizer. Please read up on it here (<https://tor.eff.org/>). In addition to encryption, it provides anonymity for the Tor client. We will be using Tor to make any communications between your EC2 client and server secure, private, and anonymous (nobody will know they are communicating with each other).

1. Tor Client

You can use the following to install and use tor from your EC2 client: <https://justhackerthings.com/post/using-tor-from-the-command-line/>

1. Tor (Hidden) Server

Create a Hidden Service using the following tutorial or an equivalent one that you find:

<https://www.linuxjournal.com/content/tor-hidden-services>

Make your EC2 server run as a Tor hidden service.

Run your chat program again while the Tor client and server are running. Perform a network capture on the same interface (not the Tor interface). Notice that the traffic should be encrypted and the destination IP should not be your EC2 server. This demonstrates how VPN’s can provide security through encryption for your network traffic.

**What IP address is your traffic going to?**

**Submit your .pcap file.**

**Perform “curl ipinfo.io” and record results.**

**IP Address Anonymity**

Observe how the results change each time you ran curl ipinfo.io.

What was the initial result of curl ipinfo.io?

What was the result of curl ipinfo.io when connected over VPN?

What was the result of curl ipinfo.io when connected over Tor?

When not using VPN or Tor, the result of ipinfo.io should be the IP of EC2 client. When connected over VPN, the IP should have been EC2 server. When connected over Tor, it should have been some random Tor IP address.

**Optional – Signal Client/Server**

Set up a private Signal Client and Server. Signal is a secure end-to-end messaging application that WhatsApp is built off of. It is an open-source project which has made the client and server available. You can use the following <https://github.com/signalapp/Signal-Server> to install and run your own private Signal server. Signal is a good example of how to build a secure and private messaging app.