Sam Barnes, Ian Schwartz, Jack Stoetzel

COSC 370 Project

5/13/19

Source Code and Sample Output

**To run this code:**

You will need Python 2.7, and then run it using: python simVPN.py

(due to how the environment is set up, it will be run using: python2 simVPN.py)

**Code:**

#simVPN.py

#this file contains functions to encypher/decypher and make a random ip

#this file also simulates a VPN

import random

#####################################################################

#makes a random IP address by using one of the 8 numbers at the beginning and then 3 other random numbers

def randomIP():

ip = ""

first = ["94", "101", "112", "115", "170", "189", "200", "234"]

rand = random.random() \* 8

ip += first[int(rand)] + '.'

for i in range(3):

x = random.random() \* 254

ip += str(int(x))

if(i != 2):

ip += '.'

return ip

#####################################################################

#simple caesar cipher to encode the message, key is 15

def encyph(str):

message = ""

for i in str:

#change lowercase

if i >= 'a' and i <= 'z':

m = chr(ord(i)+15)

#check if need to wrap around

if m > 'z':

diff = ord('z') - ord(i)

m = chr(ord('a')+(14 - diff))

#print i + " " + m

message += m

#change uppercase

elif i >= 'A' and i <= 'Z':

m = chr(ord(i)+15)

#check if need to wrap around

if m > 'Z':

diff = ord('Z') - ord(i)

m = chr(ord('A')+(14 - diff))

#print i + " " + m

message += m

else:

message += i

return message

#####################################################################

#decyphers the simple caesar cipher, key is 15

def decyph(str):

message = ""

for i in str:

#change lowercase

if i >= 'a' and i <= 'z':

m = chr(ord(i)-15)

#check if need to wrap around

if m < 'a':

diff = ord('a') - ord(i)

m = chr(ord('z')-(14 + diff))

message += m

#change uppercase

elif i >= 'A' and i <= 'Z':

m = chr(ord(i)-15)

#check if need to wrap around

if m < 'A':

diff = ord('A') - ord(i)

m = chr(ord('Z')-(14 + diff))

message += m

else:

message += i

return message

#####################################################################

#this helps display correctly where the message should go

#if current node (curr) is less than the destination node (dest),

# then the message needs to go to a higher node so add 1

#if current node (curr) is greater than the destination node (dest),

# then the message needs to go to a lower node so sub 1

#if current node (curr) equals the destination node (dest),

# then return the current node since it is not sending anywhere

def sendTo(curr, dest):

if curr < dest:

newNumber = curr + 1

return newNumber

if curr > dest:

newNumber = curr - 1

return newNumber

return curr

#####################################################################

#this helps display correctly where the message came from

#if current node (curr) is less than the destination node (dest),

# then the message came from a lower node so sub 1

#if current node (curr) is greater than the destination node (dest),

# then the message came from a higher node so add 1

#if current node (curr) equals the destination node (dest),

# then return the current node since it is not sending anywhere

def sentFrom(curr, dest):

if curr < dest:

newNumber = curr - 1

return newNumber

if curr > dest:

newNumber = curr + 1

return newNumber

return curr

#####################################################################

#print what was received when nothing needs to be sent

def printOutput(nodeNumber):

#print "\nNode " + str(nodeNumber) + ":"

#print "\tReceived from " + nodes[sentFrom(nodeNumber, dest)] + " (Node " + str(sentFrom(nodeNumber, dest)) + ")"

print "\tMessage: " + message

#####################################################################

#print statement to show message at beginning

def printStart(nodeNumber):

print "\nNode " + str(nodeNumber) + ":"

print "\tReceived from " + nodes[nodeNumber] + " (Node " + str(nodeNumber) + ")"

print "\tMessage: " + message

#####################################################################

#print statement to send packet somewhere

def printSend(nodeNumber):

print "\nNode " + str(nodeNumber) + ":"

print "\tReceived from " + nodes[sentFrom(nodeNumber, dest)] + " (Node " + str(sentFrom(nodeNumber, dest)) + ")"

print "\tMessage: " + message

print "\tSending to " + nodes[sendTo(nodeNumber, dest)] + " (Node " + str(sendTo(nodeNumber, dest)) + ")"

#####################################################################

#print statement after encyphering and need to send out

def printStartS(nodeNumber):

#print "\nNode " + str(nodeNumber) + ":"

#print "\tReceived from " + nodes[nodeNumber] + " (Node " + str(nodeNumber) + ")"

print "\tIP changed to " + nodes[nodeNumber] + " (Node " + str(nodeNumber) + ")"

print "\tMessage: " + message

print "\tSending to " + nodes[sendTo(nodeNumber, dest)] + " (Node " + str(sendTo(nodeNumber, dest)) + ")"

#####################################################################

#print statement at the end to get from correct node

def printEnd(nodeNumber):

print "\nNode " + str(nodeNumber) + ":"

if start < dest:

print "\tReceived from " + nodes[sentFrom(nodeNumber, dest)-1] + " (Node " + str(sentFrom(nodeNumber, dest)-1) + ")"

elif start > dest:

print "\tReceived from " + nodes[sentFrom(nodeNumber, dest)+1] + " (Node " + str(sentFrom(nodeNumber, dest)+1) + ")"

else:

print "\tReceived from " + nodes[sentFrom(nodeNumber, dest)] + " (Node " + str(sentFrom(nodeNumber, dest)) + ")"

print "\tMessage: " + message

#####################################################################

#main

start = input("Where to send from: ")

dest = input("Where to send to: ")

message = raw\_input("Enter the message: ")

currentNode = start

#node0 = randomIP()

#node1 = randomIP()

#node2 = randomIP()

node0 = "192.168.1.2"

node1 = "192.168.1.4"

node2 = "172.168.2.2"

print "Node 0 = " + node0

print "Node 1 = " + node1

print "Node 2 = " + node2

nodes = [node0, node1, node2]

#print nodes

#exit()

#while still the current node is not destination

while currentNode != dest:

#check if have not sent message yet by checking if currentNode is the start node

if currentNode == start:

printStart(currentNode)

print "\n\tEncyphering...\n"

#encypher message

message = encyph(message)

nodes[currentNode] = randomIP()

printStartS(currentNode)

#send to next node

currentNode = sendTo(currentNode, dest)

else:

#send to next node

printSend(currentNode)

currentNode = sendTo(currentNode, dest)

#decypher code when it reaches destination

if currentNode == dest:

printEnd(currentNode)

print "\n\tDecyphering...\n"

message = decyph(message)

printOutput(currentNode)

print "\n"

exit()

#to test if encyph and decyph function works

print "\nEncyph lower = " + encyph("the quick brown fox jumps over a lazy dog")

print "Encyph upper = " + encyph("THE QUICK BROWN FOX JUMPS OVER A LAZY DOG")

print "Encyph mixed = " + encyph("The qUICK BroWN FoX JuMpS OveR A lAZy DoG")

print "\nDecyph lower = " + decyph("iwt fjxrz qgdlc udm yjbeh dktg p apon sdv")

print "Decyph upper = " + decyph("IWT FJXRZ QGDLC UDM YJBEH DKTG P APON SDV")

print "Decyph mixed = " + decyph("Iwt fJXRZ QgdLC UdM YjBeH DktG P aPOn SdV")

**Output:**