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

# \_\_\_\_\_\_\_ #

# /\_\_\_\_\_\_\_\ #

# \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_ #

# | \_\_\_ \ / \_\_\_ \ / \_ \_ \ / \_\_\_ \ #

# | | \ \ | / \ || || || |/ / / | #

# |\_| | || | | || || || || | (\_/ #

# \_\_\_\_/ / | \\_\_\_/ || || || |\ \\_\_\_\_\_ #

# |\_\_\_\_\_/ \\_\_\_\_\_/ |\_||\_||\_| \\_\_\_\_\_\_\ #

# #

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

# File: HELP-HAMMER #

# Author: MICHAEL SKIBA #

# Date: 04-01-2018 #

# Description: FIXES GUARDIAN #

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

# Version 0.0.1 - First try

# Version 0.0.2 - Cleaned up some crap

# Version 0.0.3 - Added mega-threading capability to the UDP spammer

# Version 1.0.0 - Ready for action, changed some instruction wording

# Version 1.0.1 - Fixed 'Firmware Update, known IP' bug

# Version 2.0.0 - Added calibration and factory reset

# Version 2.1.0 - Added 'Disable hotspot' option

#

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

# Make an EXE:

# pyinstaller --onefile --icon=gicon.ico --name=HelpHammer helphammer.py

import struct

import ctypes

import copy

import sys

import time

import itertools

import threading

import socket

try:

from winsound import Beep

except ImportError:

audio\_handle = file('/dev/audio', 'wb')

def Beep(chime, chlen):

half\_period = int(8000/chime/2)

beep = chr(1)\*half\_period+chr(0)\*half\_period

beep \*= int(chime\*chlen)

audio\_handle.write(beep)

fandone = False

"""

def logoprint():

print " \_\_\_\_\_\_\_\_ \_\_\_\_ \_\_\_ \_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_ \_\_\_\_\_ \_\_\_\_\_\_\_ "

print " / \_\_\_\_\_/| | \/ \_ \\\_\_\_\_\_\_ \\_\_\_\_\_\_ \ | | / \_ \ \ \\ "

print " / \ \_\_\_| | / /\_\ \| \_/| | \| |/ /\_\ \ / | \\ "

print " \ \\_\ \ | / | \ | \| / \ / | \/ | \\"

print " \\_\_\_\_\_\_ /\_\_\_\_\_\_/\\_\_\_\_|\_\_ /\_\_\_\_|\_ /\_\_\_\_\_\_\_ /\_\_\_\\_\_\_\_|\_\_ /\\_\_\_\_|\_\_ /"

print " \/ \/ \/ \/ \/ \/ "

print " "

print " HELP-HAMMER"

print " "

"""

def logoprint():

print " "

print " ,-."

print " \_\_\_,---.\_\_ /'|`\ \_\_,---,\_\_\_"

print " ,-' \` `-.\_\_\_\_,-' | `-.\_\_\_\_,-' // `-."

print " ,' | ~'\ /`~ | `."

print " / \_\_\_// `. ,' , , \\_\_\_ \\"

print " | ,-' `-.\_\_ \_ | , \_\_,-' `-. |"

print " | / /\\_ ` . | , \_/\ \ |"

print " \ | \ \`-.\_\_\_ \ | / \_\_\_,-'/ / | /"

print " \ \ | `.\_ `\\ | //' \_,' | / /"

print " `-.\ /' \_ `---'' , . ``---' \_ `\ /,-'"

print " `` / \ ,='/ \`=. / \ ''"

print " |\_\_ /|\\_,--.,-.--,--.\_/|\ \_\_|"

print " / `./ \\\`\ | | | /,//' \,' \\"

print " / / ||--+--|--+-/-| \ \\"

print " | | /'\\_\\_\ | /\_/\_/`\ | |"

print " \ \\_\_, \\_ `~' \_/ .\_\_/ /"

print " `-.\_,-' `-.\_\_\_\_\_\_\_,-' `-.\_,-'"

print " "

print " GUARDIAN HELL-HAMMER"

def nevergonna():

quarter = 500

doteighth = 375

eighth = 250

sixteenth = 125

f3 = 349

g3 = 392

a3 = 440

bflat3 = 466

c4 = 523

d4 = 587

eflat4 = 622

e4 = 659

f4 = 698

g4 = 784

a4 = 880

song=(

(f3, sixteenth), (g3, sixteenth), (bflat3, sixteenth), (g3, sixteenth), (d4, doteighth), (d4, doteighth),

(c4, eighth + quarter), (f3, sixteenth), (g3, sixteenth), (bflat3, sixteenth), (g3, sixteenth), (c4, doteighth), (c4, doteighth),

(bflat3, quarter), (a3, sixteenth), (g3, eighth), (f3, sixteenth), (g3, sixteenth), (bflat3, sixteenth), (g3, sixteenth), (bflat3, quarter),

(c4, eighth), (a3, eighth + sixteenth), (g3, sixteenth), (f3, quarter), (f3, eighth), (c4, quarter), (bflat3, quarter + quarter))

for chime, chlen in song:

Beep(chime, chlen)

def clearit(item):

sys.stdout.write(item)

sys.stdout.flush()

sys.stdout.write('\r')

sys.stdout.flush()

def countdown(t):

while t:

mins, secs = divmod(t, 60)

timeformat = '{:02d}:{:02d}'.format(mins, secs)

sys.stdout.write(timeformat)

sys.stdout.flush()

sys.stdout.write('\r')

sys.stdout.flush()

time.sleep(1)

t -= 1

def fanimate():

for c in itertools.cycle(['|', '/', '-', '\\']):

if fandone:

sys.stdout.write('\r')

break

sys.stdout.write('\rloading ' + c)

sys.stdout.flush()

time.sleep(0.1)

def printbreak():

time.sleep(0.6)

for i in range(40):

print "-",

time.sleep(0.005)

else:

print

def validip(s):

a = s.split('.')

if len(a) != 4:

return False

for x in a:

if not x.isdigit():

return False

i = int(x)

if i < 0 or i > 255:

return False

return True

def senditudp(message, ip):

port = 9999

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

sock.sendto(message, (ip, port))

def megaup(beta, style, ip):

com = ""

fw = ""

if (style == 'esp'):

com = 'update\_esp32'

if (beta == True):

fw = '/G/esp\_beta.bin'

else:

fw = '/G/esp\_prod.bin'

else:

com = 'update\_lora'

if (beta == True):

fw = '/G/st\_beta.bin'

else:

fw = '/G/st\_prod.bin'

message = '{"command":"' + com + '", "address":"104.236.234.184", "port":"80", "filename":"' + fw + '", "type":0}'

senditudp(message, ip)

def wificonn():

print ("""

Please connect your PC to the Guardian hotspot

It will be displayed as Guardian-XXXX, where

XXXX is the device PIN

""")

raw\_input("Press Enter when you're connected")

print ("""

Please enter the Wi-Fi network name

(Case sensitive!!!)

""")

ssid = raw\_input(" >>> ")

printbreak()

print ("""

Please enter the Wi-Fi password

(Case Sensitive!!!)

""")

password = raw\_input(" >>> ")

printbreak()

print "Please wait..."

ip = "192.168.4.1"

UDP\_PORT = 9999

wifimessage = '{"command":"set\_WIFI\_station","type":0,"SSID":"' + ssid + '","' + 'PASS":"' + password + '", "connect":0}'

senditudp(wifimessage, ip)

time.sleep(6)

apmessage = '{"command":"set\_WIFI\_ap", "option":0 ,"type":0}'

senditudp(apmessage, ip)

time.sleep(3)

print "DONE!!"

time.sleep(3)

printbreak()

menuprime()

def apoff():

print "One moment, finding Valve Controller..."

guardian = str(hammertime())

print "Found it! Sending commands, please wait..."

apmessage = '{"command":"set\_WIFI\_ap", "option":0 ,"type":0}'

senditudp(apmessage, guardian)

time.sleep(3)

print "DONE!!"

time.sleep(2)

printbreak()

menuprime()

def hammertime():

def get\_macaddress(host):

""" Returns the MAC address of a network host, requires >= WIN2K. """

# Check for api availability

try:

SendARP = ctypes.windll.Iphlpapi.SendARP

except:

raise NotImplementedError('Usage only on Windows 2000 and above')

# Doesn't work with loopbacks, but let's try and help.

if host == '127.0.0.1' or host.lower() == 'localhost':

host = socket.gethostname()

# gethostbyname blocks, so use it wisely.

try:

inetaddr = ctypes.windll.wsock32.inet\_addr(host)

if inetaddr in (0, -1):

raise Exception

except:

hostip = socket.gethostbyname(host)

inetaddr = ctypes.windll.wsock32.inet\_addr(hostip)

buffer = ctypes.c\_buffer(6)

addlen = ctypes.c\_ulong(ctypes.sizeof(buffer))

if SendARP(inetaddr, 0, ctypes.byref(buffer), ctypes.byref(addlen)) != 0:

raise WindowsError('Retreival of mac address(%s) - failed' % host)

# Convert binary data into a string.

macaddr = ''

for intval in struct.unpack('BBBBBB', buffer):

if intval > 15:

replacestr = '0x'

else:

replacestr = 'x'

macaddr = ''.join([macaddr, hex(intval).replace(replacestr, '')])

return macaddr.upper()

results = []

splitsmac = []

splitsip = []

def arpanet(iptry, j):

try:

macaddy = get\_macaddress(iptry)

result = str(iptry) +'='+ str(macaddy)

results.append(result)

splitsmac.append(str(macaddy))

splitsip.append(str(iptry))

time.sleep(1)

except WindowsError:

pass

UDP\_IP = "255.255.0.0"

UDP\_PORT = 9999

goodip = str(socket.gethostbyname(socket.gethostname()))

subnettemp = goodip

subnetsplit = subnettemp.split('.')

subnet = subnetsplit[0]+'.'+subnetsplit[1]+'.'+subnetsplit[2]+'.'

ipaddy = subnet

threads = []

for j in range(1,255):

newguy = ipaddy + str(j)

try:

g = threading.Thread(target = arpanet, args = (newguy, j))

threads.append(g)

g.start()

if j == 254:

for threadz in threads:

threadz.join()

except WindowsError:

pass

guardip = ""

for i in splitsmac:

if i.startswith("30AE"):

guardip = str(splitsip[splitsmac.index(i)])

return guardip

def firmwareupdate():

beta = ""

style = ""

ans = ""

print ("""

Beta?!

""")

ans = raw\_input(" Y/N: ")

if (ans=="Y" or ans=="y"):

beta = True

elif (ans=="N" or ans=="n"):

beta = False

guardian = str(hammertime())

printbreak()

print "Sending ESP32 update, please wait"

megaup(beta, "esp", guardian)

countdown(35)

printbreak()

print "Sending ST update, please wait"

megaup(beta, "st", guardian)

countdown(35)

clearit(" ")

printbreak()

def calibrate():

guardian = str(hammertime())

close = '{"command":"motor\_calibration", "action":"close","type":0}'

open = '{"command":"motor\_calibration", "action":"open","type":0}'

print "Stand by, calibrating..."

senditudp(open, guardian)

time.sleep(15)

senditudp(close, guardian)

time.sleep(15)

senditudp(open, guardian)

time.sleep(15)

print "DONE!"

printbreak()

def factdef():

guardian = str(hammertime())

message = '{"command":"DEBUG\_factory\_reset","type":0}'

senditudp(message, guardian)

time.sleep(3)

print 'DONE!'

time.sleep(2)

printbreak()

def menuprime():

ans=True

while ans:

print ("""

Welcome!

Read the choices carefully since the menu options have changed:

1. Connect Guardian to Wi-Fi

2. Disable hotspot

3. Firmware update

4. Calibration

5. Factory reset

6. Exit

""")

ans=raw\_input(" What will it be, buddy?: ")

if ans=="1":

print("\n Ok hold on")

printbreak()

wificonn()

elif ans=="2":

print("\n Coming right up!")

printbreak()

apoff()

elif ans=="3":

print("\n Sure, one sec")

printbreak()

firmwareupdate()

elif ans == "4":

print("\n Ok hold on")

printbreak()

calibrate()

elif ans == "5":

print("\n Sure, one sec")

printbreak()

factdef()

elif ans=="6":

printbreak()

print("\n BYE BYE!!")

time.sleep(2)

exit(0)

else:# ans !="":

print("\n Not a valid choice, try again!!")

printbreak()

if \_\_name\_\_=="\_\_main\_\_":

fandone = False

logoprint()

t = threading.Thread(target=fanimate)

t.start()

nevergonna()

fandone = True

sys.stdout.write('\r')

sys.stdout.flush()

print " "

printbreak()

menuprime()