LABORATOR 3-- Socket Programming în Python.

1 Socket Programming

Programarea socketului este o modalitate de a conecta două noduri dintr-o rețea pentru a comunica între ele. Un socket (nod) ascultă pe un anumit port la un IP, în timp ce celălalt ajunge pentru a forma o conexiune. Serverul joacă rolul de ascultător în timp ce clientul are legătura la server. Ele sunt adevăratele coloane vertebrale din spatele navigării pe web. În termeni mai simpli, există un server și un client. Programarea socketului este începută prin importul bibliotecii de socket și realizarea unui socket simplu.

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Aici am făcut o instanță de socket și i-am transmis doi parametri. Primul parametru este AF_INET, iar al doilea este $SOCK\_STREAM$. AF_INET se referă la adresa-familiei ipv4. $SOCK\_STREAM$ înseamnă protocol TCP orientat spre conexiune. Acum ne putem conecta la un server folosind acest socket. Conectarea la un server:

```
$ ping www.google.com
```

Rețineți că, dacă apare vreo eroare în timpul creării unui socket, atunci avem o eroare și ne putem conecta la un server doar cunoscând IP-ul acestuia. Puteți găsi IP-ul serverului folosind aceasta:

```
import socket

ip = socket.gethostbyname('www.google.com')
print ip
```

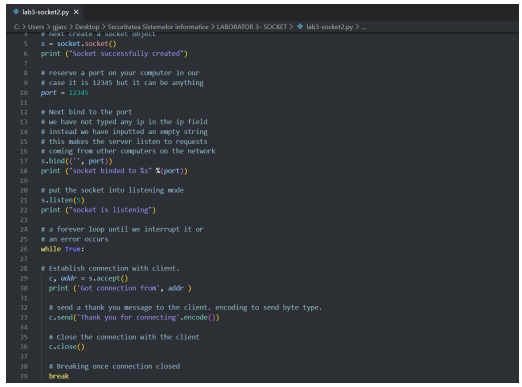
Un alt exemplu de script pentru conectarea la google:

```python
# An example script to connect to Google using socket
# programming in Python
import socket # for socket
import sys

try:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print ("Socket successfully created")
except socket.error as err:
    print ("socket creation failed with error %s" %(err))

# default port for socket
port = 80

try:
    host_ip = socket.gethostbyname('www.google.com')
except socket.gaierror:

    # this means could not resolve the host
    print ("there was an error resolving the host")
    sys.exit()

# connecting to the server
s.connect((host_ip, port))

print ("the socket has successfully connected to google")
```

la Output va afisa:

```
Socket successfully created
the socket has successfully connected to google
on port == 173.194.40.19
```

Să concluzionăm: În primul rând, am realizat un socket.
Apoi am rezolvat IP-ul google și, în sfârșit, ne-am conectat la google.
Acum trebuie să știm cum putem trimite niște date printr-un socket.
Pentru trimiterea datelor, biblioteca socket are o funcție sendall. Această funcție vă permite să trimiteți date către un server la care este conectat și serverul poate trimite și date către client folosind această funcție.
Un program simplu server-client:



Server :

Un server are o metodă bind() care îl leagă la un anumit IP și port, astfel încât să poată asculta cererile primite pe acel IP și port. Un server are o metodă listen() care pune serverul în modul de ascultare. Acest lucru permite serverului să asculte conexiunile primite. Și, în sfârșit, un server are o metodă accept() și close(). Metoda accept inițiază o conexiune cu clientul, iar metoda close închide conexiunea cu clientul.

În primul rând, importăm socketul necesar. Apoi am făcut un obiect socket și am rezervat un port pe computerul nostru. După aceea, ne-am conectat serverul la portul specificat. Trecerea unui șir gol înseamnă că serverul poate asculta conexiunile primite și de la alte computere. Dacă am fi trecut de 127.0.0.1, atunci ar fi ascultat doar acele apeluri efectuate în computerul local. După aceea punem serverul în modul de ascultare.5 aici înseamnă că 5 conexiuni sunt menținute în așteptare dacă serverul este ocupat și dacă un al 6-lea socket încearcă să se conecteze atunci conexiunea este refuzată. În cele din urmă, facem o buclă și începem să acceptăm toate conexiunile de intrare și să închidem acele conexiuni după un mesaj de mulțumire către toate socketurile conectate. Client: Acum avem nevoie de ceva cu care un server poate interacționa. Am putea da telenet la server astfel doar pentru a ști că serverul nostru funcționează. Tastați aceste comenzi în terminal:

```
# start the server
$ python server.py

# keep the above terminal open
# now open another terminal and type:

$ telnet localhost 12345
```

Dacă ”telnet” nu este recunoscut, pe Windows, căutați funcțiile Windows și activați caracteristica ”client telnet”.

Ieșire:

```
# in the server.py terminal you will see
# this output:
Socket successfully created
socket binded to 12345
socket is listening
Got connection from ('127.0.0.1', 52617)

# In the telnet terminal you will get this:
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Thank you for connectingConnection closed by foreign host.
```

Aceasta arata că serverul nostru funcționează. Acum pentru partea clientului:

```
# Import socket module
import socket

# Create a socket object
s = socket.socket()

# Define the port on which you want to connect
port = 12345

# connect to the server on local computer
s.connect(('127.0.0.1', port))

# receive data from the server and decoding to get the string.
print (s.recv(1024).decode())
# close the connection
s.close()
```

În primul rând, facem un socket.
Apoi ne conectăm la localhost pe portul 12345 (portul pe care rulează serverul nostru) și, în sfârșit, primim date de la server și închidem conexiunea.
Acum salvăm acest fișier ca și client.py și rulăm de pe terminal după pornirea scriptului de server.

2 Steganografie cu Python

Steganografia imaginii este procesul de ascundere a datelor secrete într-o imagine. În această postare, vom ascunde o imagine în alta și o vom converti într-o altă imagine și apoi vom extrage înapoi ambele imagini din imaginea anterioară.

Ideea din spatele steganografiei bazate pe imagini este foarte simplă. Imaginile sunt compuse din date digitale (pixeli), care descrie ceea ce este în interiorul imaginii, de obicei culorile tuturor pixelilor. Din moment ce știm că fiecare imagine este formată din pixeli și fiecare pixel conține 3 valori (roșu, verde, albastru).

De exemplu, să presupunem că trebuie să ascundem img2 în img1, unde ambele img1 și img2 sunt date ca matrice de valori pixeli. Dimensiunea img2 trebuie

să fie mai mică decât dimensiunea img1. Folosim imagini color, prin urmare ambele vor avea 3 valori (roșu, verde, albastru). Valoarea fiecărui pixel variază de la 0 la 255, deci valoarea fiecărui pixel este de 1 octet sau 8 biți. Fie img[i][j][l] valoarea pixelului la locația (i, j) și a canalului l unde i variază de la 0 la lățime și j variază de la 0 la înălțime și l variază de la 0 la 2.

Notă: Calitatea noilor imagini este puțin mai mică decât a imaginilor vechi. Dorim să ascundem imaginea 2 în imaginea 1. Mai jos este implementarea in Python.

```python
def encrypt():

    # img1 and img2 are the
    # two input images
    img1 = cv2.imread('pic1.jpg')
    img2 = cv2.imread('pic2.jpg')

    for i in range(img2.shape[0]):
        for j in range(img2.shape[1]):
            for l in range(3):

                # v1 and v2 are 8-bit pixel values
                # of img1 and img2 respectively
                v1 = format(img1[i][j][l], '08b')
                v2 = format(img2[i][j][l], '08b')

                # Taking 4 MSBs of each image
                v3 = v1[:4] + v2[:4]

                img1[i][j][l]= int(v3, 2)

    cv2.imwrite('pic3in2.png', img1)
```

```python
# Decryption function
def decrypt():

    # Encrypted image
    img = cv2.imread('pic3in2.png')
    width = img.shape[0]
    height = img.shape[1]

    # img1 and img2 are two blank images
    img1 = np.zeros((width, height, 3), np.uint8)
    img2 = np.zeros((width, height, 3), np.uint8)

    for i in range(width):
        for j in range(height):
            for l in range(3):
                v1 = format(img[i][j][l], '08b')
                v2 = v1[:4] + chr(random.randint(0, 1)*48) * 4
                v3 = v1[4:] + chr(random.randint(0, 1)*48) * 4

                # Appending data to img1 and img2
                img1[i][j][l]= int(v2, 2)
```

```python
                img2[i][j][l]= int(v3, 2)

    # These are two images produced from
    # the encrypted image
    cv2.imwrite('pic2_re.png', img1)
    cv2.imwrite('pic3_re.png', img2)


# Driver's code
encrypt()
decrypt()
```

3 Steganografie bazată pe imagini folosind Python

1) Criptare

Abordare:

Importați imaginea din modulul PIL folosind cuvântul cheie import.
Importați modulul stepic folosind cuvântul cheie import.
Deschideți imaginea folosind funcția open() a modulului Image, pasând numele/calea fișierului imagine ca argument în care va fi stocat mesajul secret.
Dați un mesaj secret aleatoriu și stocați-l într-o variabilă.
Convertiți mesajul secret dat de mai sus în format UTF-8 folosind funcția encode() și stocați-l în aceeași variabilă.
Treceți imaginea dată și mesajul secret în funcția encode() a modulului stepic care oferă o nouă imagine în care mesajul secret este ascuns.
Salvați imaginea codificată/criptată de mai sus cu un nume și o extensie aleatorie folosind funcția save(). Tipăriți un text aleatoriu pentru confirmare.
Ieșirea din program.
Mai jos este implementarea:

```
# Import Image from PIL module using the import keyword
from PIL import Image
# Import stepic module using the import keyword
import stepic
# Open the image using the open() function of the Image module
# by passing the image filename/path as an argument to it
# in which the secret message will be stored.
gvn_image = Image.open("sampleimage.png")
# Give some random secret message and store it in a variable
secret_msg = "Welcome to Python-programs"
# Convert the above given secret message into UTF-8 format using the encode() function
# and store it in the same variable
secret_msg = secret_msg.encode()
# Pass the given image and secret message into the encode() function of the stepic module
# which gives a new image within which the secret message is hidden.
encodedimage = stepic.encode(gvn_image, secret_msg)
# Save the above encoded/encrypted image with some random name and extension using the
# save() function
encodedimage.save("EncryptedImage.png")
# Print some random text for acknowledgment
print("Encryption has done successfully")
```

2) Decriptare

Abordare:

Importați imaginea din modulul PIL folosind cuvântul cheie import.
Importați modulul stepic folosind cuvântul cheie import.
Deschideți imaginea din care doriți să extrageți mesajul secret (imaginea criptată) folosind funcția open() pasând numele fișierului/calea ca argument.
Stocați-l într-o variabilă.
Treceți imaginea criptată de mai sus la funcția decode() a modulului stepic pentru a oferi mesajul secret criptat în format șir.
Stocați-l într-o altă variabilă.
Tipăriți un text aleatoriu pentru confirmare.

Imprimați mesajul decriptat/decodificat.
Ieșirea din program.

```
# Import Image from PIL module using the import keyword
from PIL import Image
# Import stepic module using the import keyword
import stepic
# Open the image from which you want to extract the secret message(encrypted image) using the
# open() function by passing filename/path as an argument to it.
# Store it in a variable
encryptd_image = Image.open("EncryptedImage.png")
# Pass the above encrypted image to the decode() function of the stepic module
# to give the encrypted secret message in the string format.
# Store it in another variable
decryptedmessage = stepic.decode(encryptd_image)
# Print some random text for acknowledgment
print("Decryption has done successfully")
# Print the decrypted/decoded message
print("The decrypted message = ", decryptedmessage)
```

Se va vedea la decodificare mesajul ascuns de noi in prealabil la codificare.

4 Bibliografie

Python Hacking Essentials Paperback 2015 by Earnest Wish,
https://www.amazon.com/Python-Hacking-Essentials-Earnest-Wish/dp/1511797568
https://www.geeksforgeeks.org/socket-programming-python/