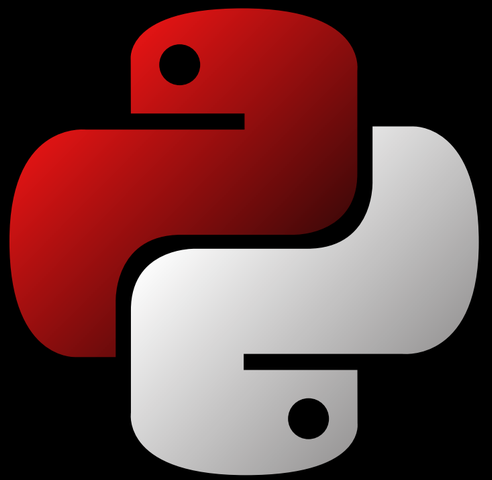
DEEPSEC ACADEMY

DeepSec Pentesting Web Python Course



Sockets

¿Que son los sockets?

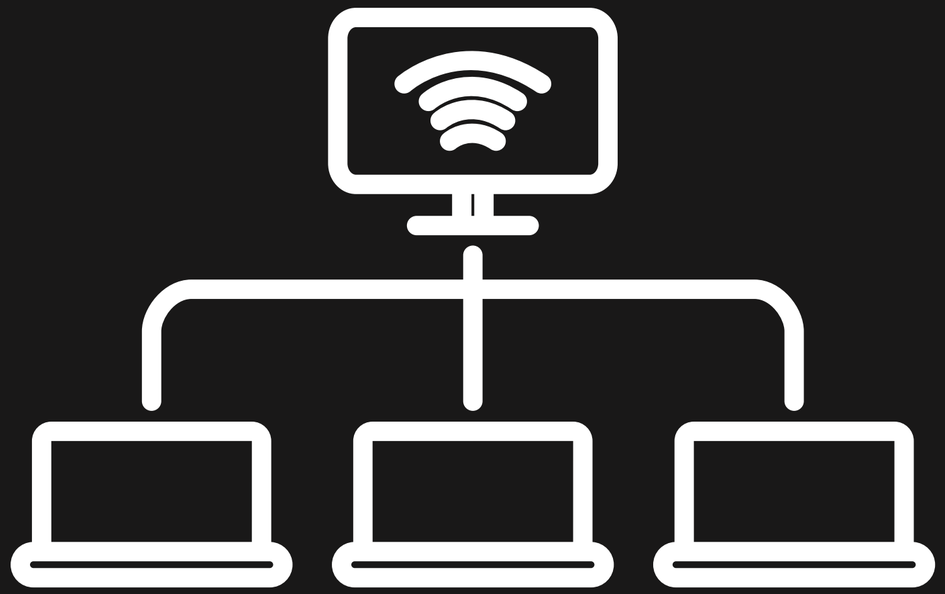
Socket es el intercambio de datos de una forma fiable y ordenada entre dos procesos (posiblemente situados en computadoras distintas)

DEEPSEC ACADEMY

Sockets

Programacion de sockets en Python

La programacion de sockets es una forma de conectar dos nodos en una red para comunicarse entre si



DEEPSEC ACADEMY

Sockets

Programacion de sockets en Python

Un socket (nodo) escucha en un puerto particular en una IP

IP: 192.168.86.31

Puerto en escucha: 7676

Sockets

Programacion de sockets en Python

Luego el otro socket se comunica con el otro para formar una conexion
En conclusion, el servidor forma el socket mientras el cliente se comunica con el servidor

IP: 192.168.86.31

Puerto en escucha: 7676

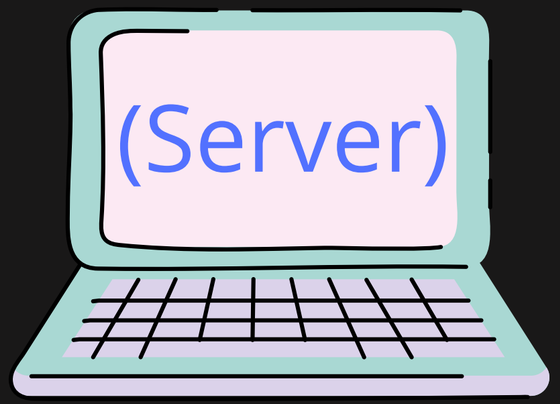
IP: 192.168.86.32

DEEPSEC ACADEMY

Sockets

Programacion de sockets en Python

Para la conexion se hara uso de la libreria **socket** y crearemos el socket



IP: 192.168.86.31

Puerto en escucha: 7676

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

Sockets

Programacion de sockets en Python

La variable **s** es una instancia de socket y en el constructor asignamos los parametros **socket.AF_INET** y **socket.SOCK_STREAM**

- **AF_INET**: Se refiere a la familia de direcciones ipv4
- **SOCK_STREAM**: significa protocolo TCP orientado a la conexion

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

DEEPSEC ACADEMY

Sockets

Programacion de sockets en Python

Con python podemos encontrar una IP

DNS: www.google.com
IP: 64.233.186.103

```
import socket
host_ip =
socket.gethostbyname('www.google.com')
```

Sockets

Programa Cliente-Servidor

Un servidor tiene un metodo bind() que lo vincula a una IP y puerto especifico para que pueda escuchar las solicitudes entrantes en esa IP y puerto especificado

```
import socket
s = socket.socket()
port = 7676
ip_address = "192.168.86.31"
s.bind((ip_address, port))
```

Sockets

Programa Cliente-Servidor

Un servidor tiene un metodo listen(**x**) acepta un tamaño de cola atraves de la acumulacion de parametros, basicamente indica el numero maximo de conexiones **x**

```
import socket
s = socket.socket()
port = 7676
ip_address = "192.168.86.31"
s.bind((ip_address, port))
max_clients = 5
s.listen(max_clients)
```

Sockets

Programa Cliente-Servidor

Deberemos de encerrar en un bucle infinito el establecimiento de la conexión así nos evitamos de problemas

```
import socket
s = socket.socket()
port = 7676
ip_address = "192.168.86.31"
s.bind((ip_address, port))
max_clients = 5
s.listen(max_clients)
while True:
        conection()
        break
```

Sockets

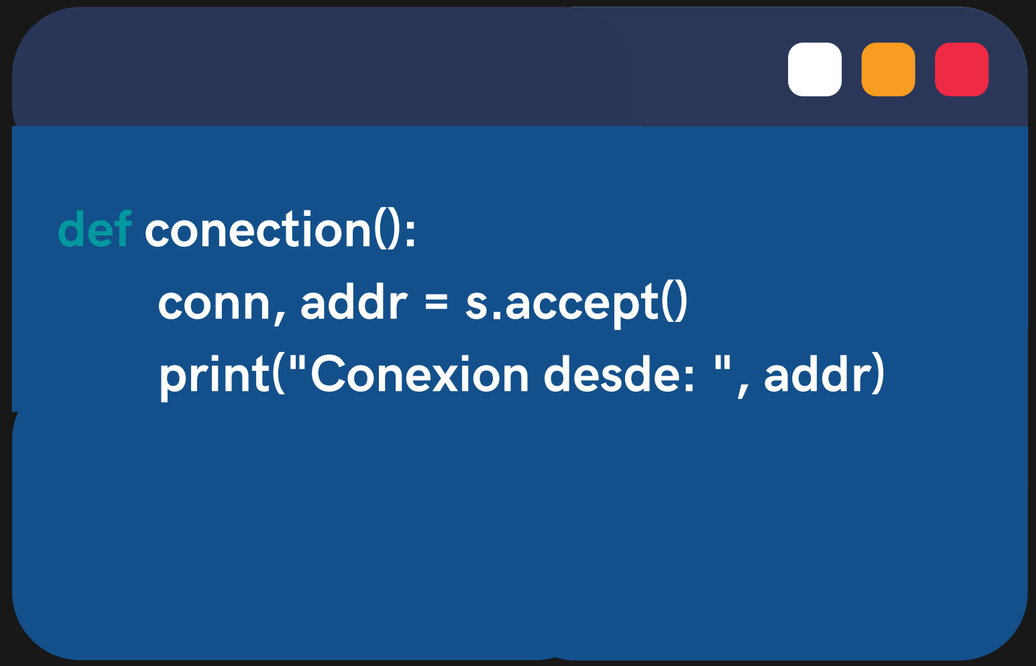
Programa Cliente-Servidor

El metodo accept() se encarga de bloquear la ejecucion y espera a una conexion entrante, devuelve un objeto socket que representa la conexion y una tupla que contiene:

(host, port)

Para conexiones en ipv6 nos devolvera:

(host, port, flowinfo, scopeid)



```
def conection():
    conn, addr = s.accept()
    print("Conexion desde: ", addr)
```

Sockets

Programa Cliente-Servidor

El metodo send() se usa para enviar datos de un socket a otro(Nota: los datos deben estar codificados en bytes).
El metodo **cadena**.encode() se usa para pasar a bytes la **cadena**.
El metodo close() se encarga de cerrar la conexion del socket **conn**.

```
def conection():
        conn, addr = s.accept()
        print("Conexion desde: ", addr)
        conn.send("Gracias por
                conectarte".encode())
        conn.close()
```

Sockets

Programa Cliente-Servidor

Para el programa cliente lo unico que cambia es que adicionamos un metodo recv(**x**) que se encarga de recibir una cantidad **x** de bytes y el metodo decode() se encarga de pasar de bytes a string

```
import socket
s = socket.socket()

port = 7676
server_ip = "192.168.86.31"
s.connect((server_ip, port))
print(s.recv(1024).decode())
s.close()
```

Sockets

Programa Cliente-Servidor

Para el programa cliente lo unico que cambia es que adicionamos un metodo recv(**x**) que se encarga de recibir una cantidad **x** de bytes y el metodo decode() se encarga de pasar de bytes a string

```
import socket
s = socket.socket()

port = 7676
server_ip = "192.168.86.31"
s.connect((server_ip, port))
print(s.recv(1024).decode())
s.close()
```

Sockets

Programa Cliente-Servidor (Cliente con la libreria PWN)

El metodo remote() nos ayuda a conectarnos a un determinado servidor y el metodo recvline() nos ayuda a mostrar lo que nos envie el servidor a nuestro cliente

```
from pwn import *
conn = remote('192.168.86.33', 21)
print(conn.recvline())
```

¡Muchas gracias!