

# Esercizio 10/10/2023

1 – Cos'è una Backdoor.

Una backdoor è letteralmente una “porta sul retro”, cioè una porta attraverso la quale un utente/informatico può accedere da remoto per andare a riprendere il controllo della macchina compromessa senza passare per le fasi di autenticazione. Per installarla bisogna essere amministratori o comunque “proprietari” del software.

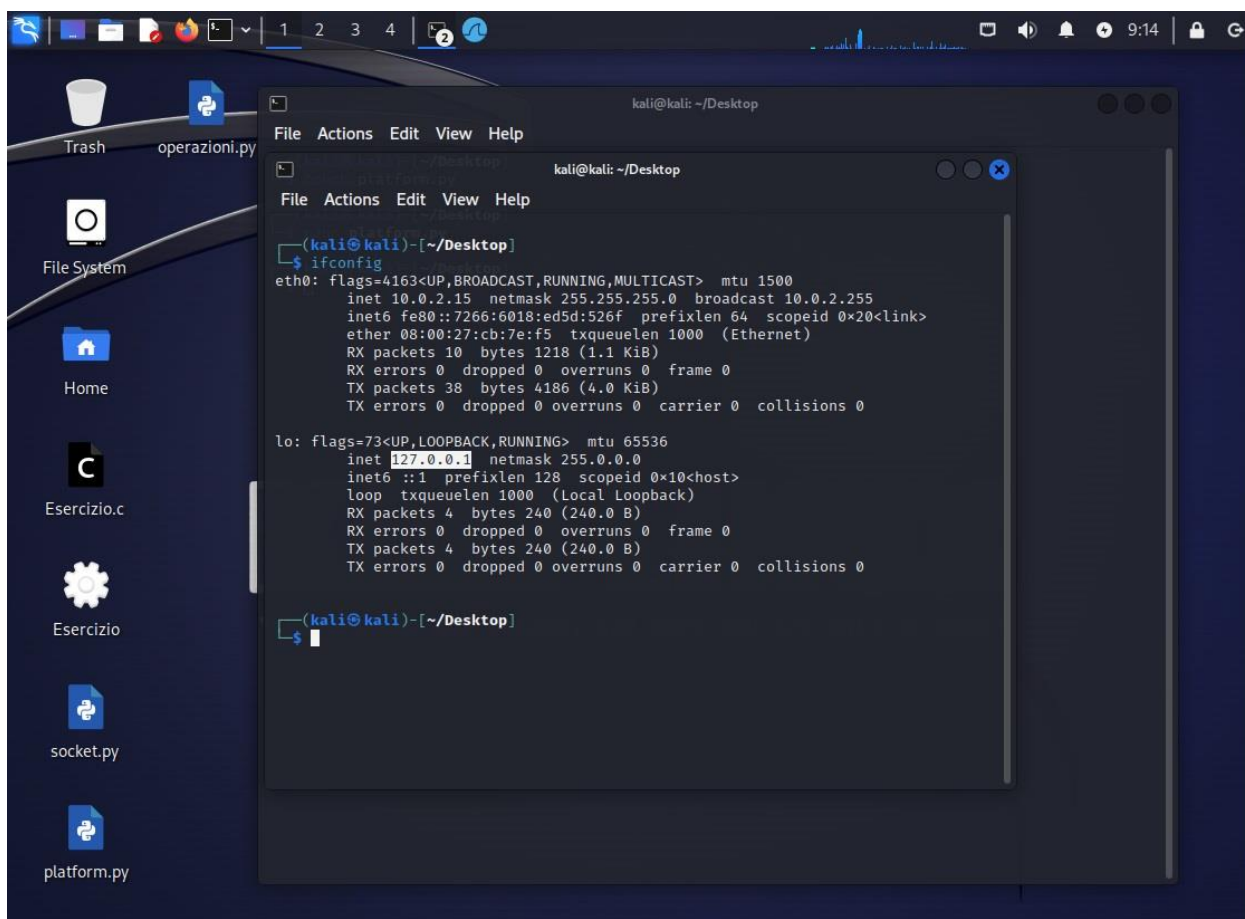
2 – Perché è pericolosa

E' pericolosa perché è soggetta ad attacchi da parte di black hat in quanto anche il black hat bypassa il sistema di autenticazione. Quando la backdoor è utilizzata da un black hat viene chiamata RAT (Remote Access Trojan), quindi una backdoor malevola

3 – Cosa fanno i codici e qual è la differenza

I codici utilizzati fanno riferimento uno a stabilire una connessione ad un server e ad una porta (socket), l'altro ci consente, una volta stabilita la connessione, di eseguire i comandi al suo interno, cioè ottenere informazioni di sistema o di elencare i contenuti di una directory (platform)

4 – Sono andata a cercare l'indirizzo IP di Kali attraverso il comando ifconfig (probabilmente errato, ho provato sia con inet 127.0.0.1 che è chiaramente un localhost, sia per sicurezza, con inet 10.0.2.15)

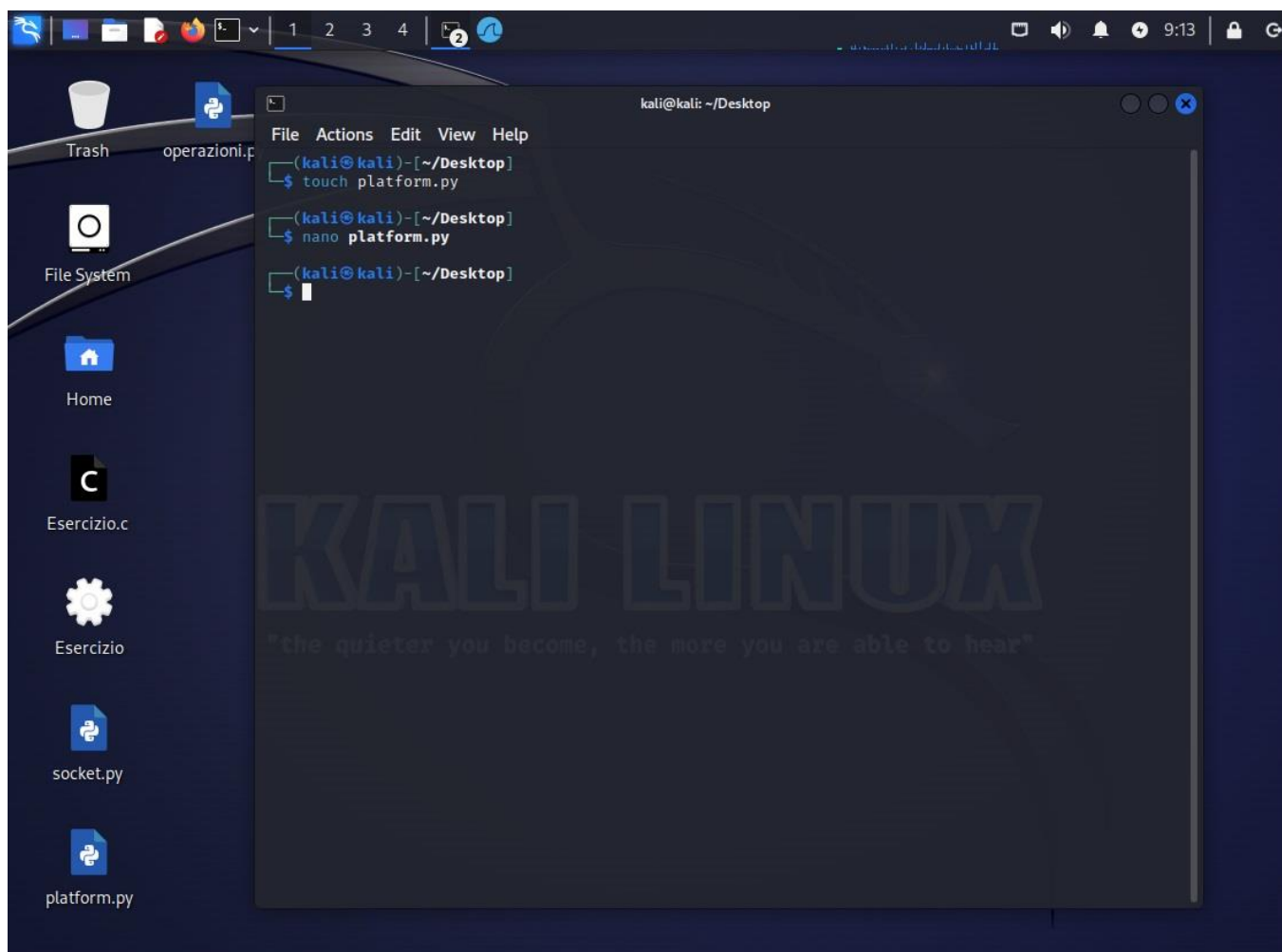


```
kali@kali: ~/Desktop
File Actions Edit View Help
File Actions Edit View Help
kali@kali: ~/Desktop
(kali@kali)-[~/Desktop]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::7266:6018:ed5d:526f prefixlen 64 scopeid 0<link>
    ether 08:00:27:cb:7e:f5 txqueuelen 1000 (Ethernet)
    RX packets 10 bytes 1218 (1.1 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 38 bytes 4186 (4.0 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

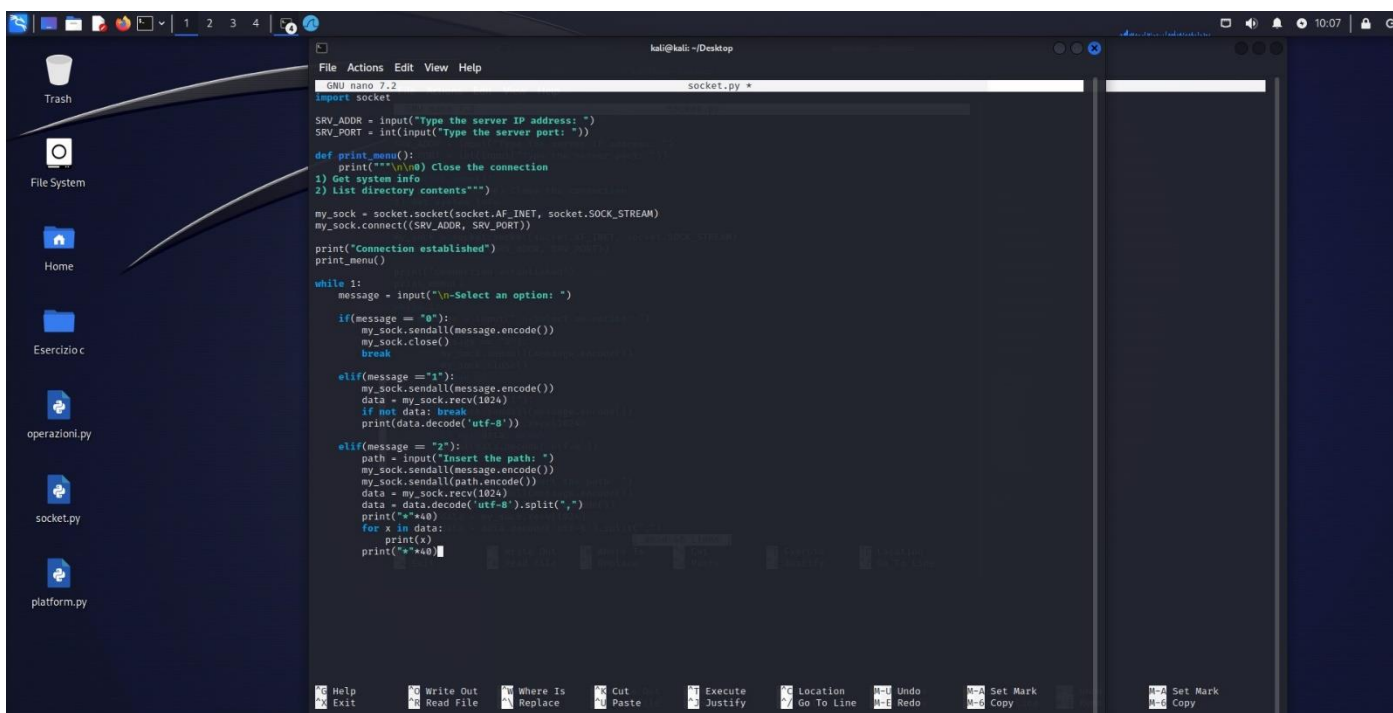
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 4 bytes 240 (240.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4 bytes 240 (240.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(kali@kali)-[~/Desktop]
$
```

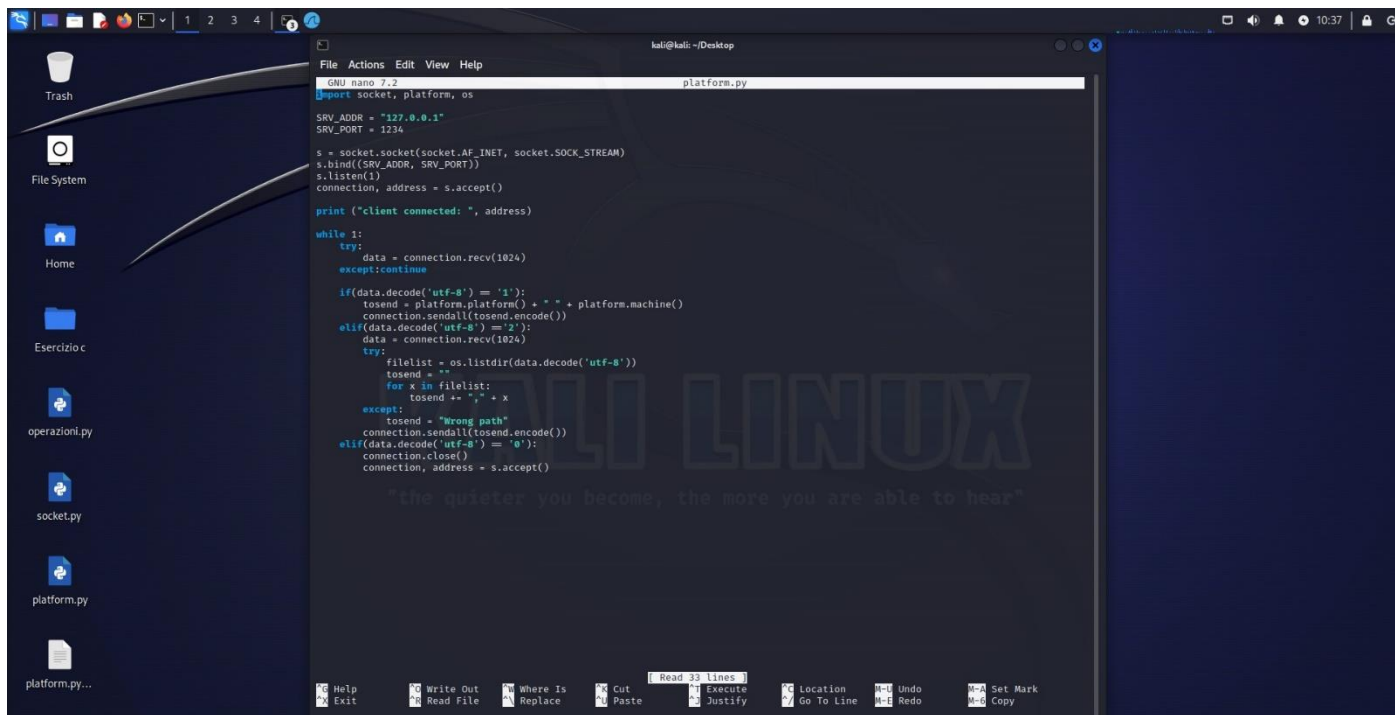
5 – Ho creato le due cartelle (socket e Platform)



6 – Sono entrata nel programma Socket attraverso il comando nano e l'ho modificato inserendo nella quart'ultima e nell'ultima riga un asterisco all'interno delle virgolette e aggiungendo prima del 40 un altro asterisco, quindi `print("***40")`



7 – Sono entrata nel programma Platform attraverso il comando nano e ho inserito l'indirizzo IP



The screenshot shows a Kali Linux desktop environment. A terminal window is open, displaying the nano text editor editing a file named `platform.py`. The code in the editor is a Python script for a socket server. It imports the `socket` module, defines `SRV_ADDR = "127.0.0.1"` and `SRV_PORT = 1234`, and then uses `socket.socket()` to create a server socket. It binds the socket to the address and port, starts listening, and enters a `while 1:` loop to accept connections. Inside the loop, it receives data, decodes it, and checks for specific commands like `'1'` or `'2'`. If `'1'` is received, it sends back the machine's platform information. If `'2'` is received, it lists the contents of the current directory. The desktop background features the Kali Linux logo and the text "the quieter you become, the more you are able to hear".

```
GNU nano 2.2 platform.py
import socket, platform, os

SRV_ADDR = "127.0.0.1"
SRV_PORT = 1234

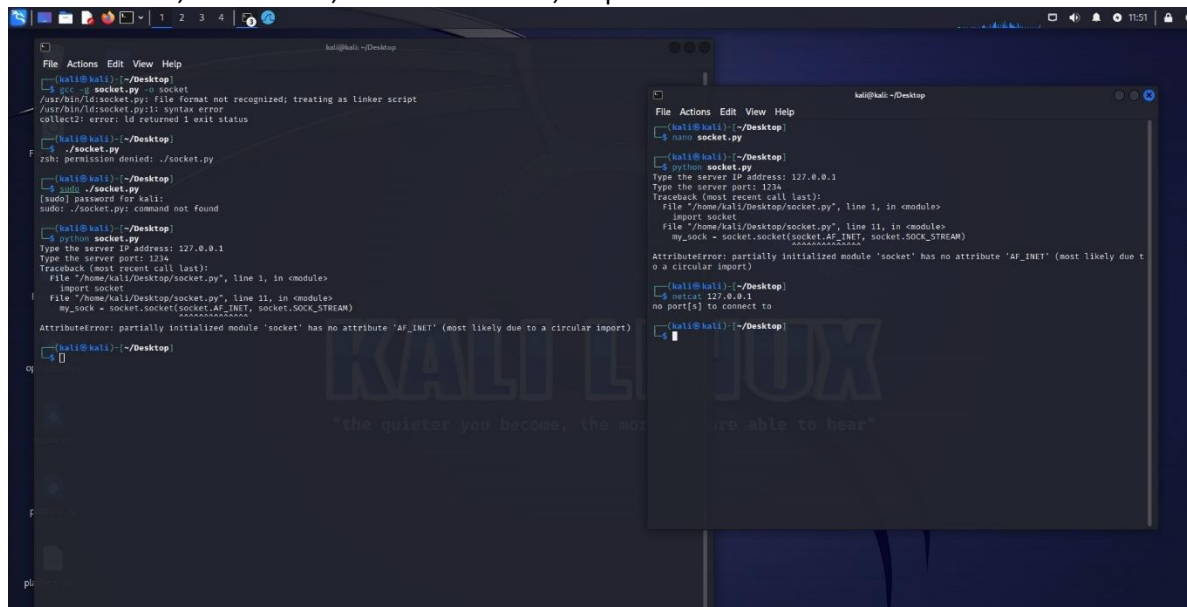
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((SRV_ADDR, SRV_PORT))
s.listen(1)
connection, address = s.accept()

print ("client connected: ", address)

while 1:
    try:
        data = connection.recv(1024)
        except:continue

    if(data.decode('utf-8') == '1'):
        tosend = platform.platform() + " " + platform.machine()
        connection.sendall(tosend.encode())
    elif(data.decode('utf-8') == '2'):
        data = connection.recv(1024)
        try:
            filelist = os.listdir(data.decode('utf-8'))
            tosend = ""
            for x in filelist:
                tosend += " " + x
            except:
                tosend = "Wrong path"
            connection.sendall(tosend.encode())
    elif(data.decode('utf-8') == '0'):
        connection.close()
        connection, address = s.accept()
```

8 – Non riesco, ovviamente, a testare il codice, ho provato anche con il comando sudo



The screenshot shows a Kali Linux desktop with a terminal window displaying several error messages. The user has attempted to run the `socket.py` script using `python socket.py`, `sudo ./socket.py`, and `python socket.py`. The errors include "File format not recognized; treating as linker script", "syntax error: collect2: error: ld returned 1 exit status", "permission denied: ./socket.py", and "command not found". A detailed traceback is shown, indicating an `AttributeError: partially initialized module 'socket' has no attribute 'AF_INET' (most likely due to a circular import)`. The desktop background is the same Kali Linux logo and text as in the previous screenshot.

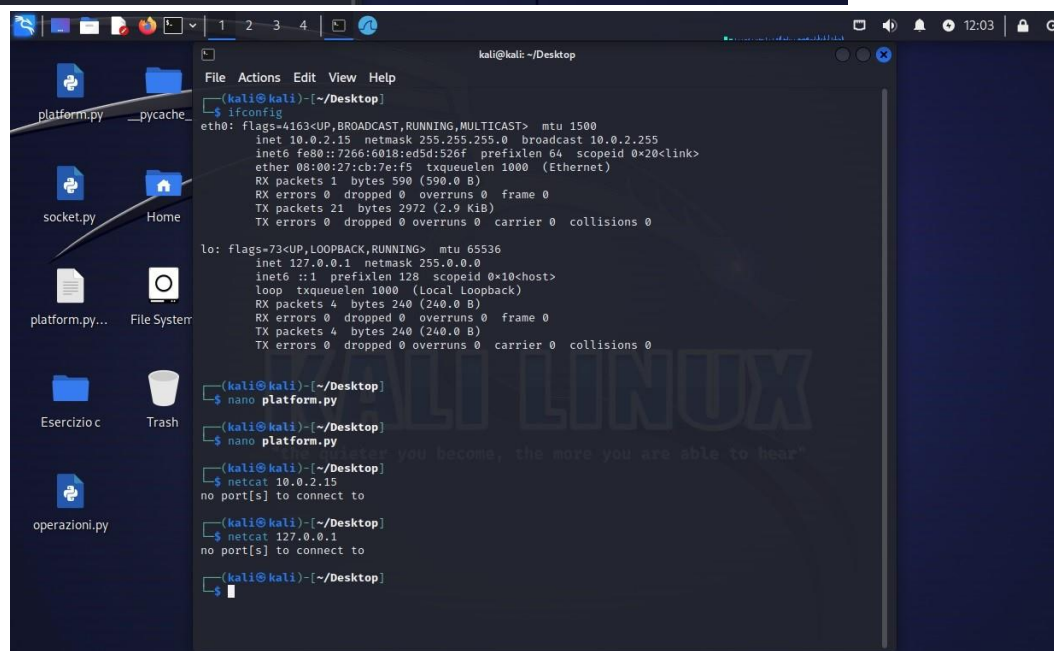
```
(kali@kali):~/Desktop
$ python socket.py
/usr/bin/ld:socket.py: file format not recognized; treating as linker script
/usr/bin/ld:socket.py:1: syntax error
collect2: error: ld returned 1 exit status

(kali@kali):~/Desktop
$ ./socket.py
zsh: permission denied: ./socket.py

(kali@kali):~/Desktop
$ sudo ./socket.py
[sudo] password for kali:
sudo: ./socket.py: command not found

(kali@kali):~/Desktop
$ python socket.py
Type the server IP address: 127.0.0.1
Type the server port: 1234
Traceback (most recent call last):
  File "/home/kali/Desktop/socket.py", line 11, in <module>
    import socket
  File "/home/kali/Desktop/socket.py", line 11, in <module>
    my_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
AttributeError: partially initialized module 'socket' has no attribute 'AF_INET' (most likely due to a circular import)

(kali@kali):~/Desktop
$
```



The screenshot shows a Kali Linux desktop with a terminal window displaying network configuration and netcat attempts. The user has run `ifconfig`, showing details for the `eth0` and `lo` interfaces. The `eth0` interface is configured with IP `10.0.2.15` and netmask `255.255.255.0`. The `lo` interface is the loopback interface with IP `127.0.0.1` and netmask `255.0.0.0`. The user has then attempted to run `netcat 10.0.2.15` and `netcat 127.0.0.1`, both of which resulted in "no port[s] to connect to". The desktop background is the same Kali Linux logo and text as in the previous screenshots.

```
(kali@kali):~/Desktop
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::7266:6018:ed5d:526f prefixlen 64 scopeid 0<20<link>
    ether 08:00:27:cb:7e:f5 txqueuelen 1000 (Ethernet)
    RX packets 1 bytes 590 (590.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 21 bytes 2972 (2.9 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0<10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 4 bytes 240 (240.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4 bytes 240 (240.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(kali@kali):~/Desktop
$ nano platform.py
(kali@kali):~/Desktop
$ nano platform.py
(kali@kali):~/Desktop
$ netcat 10.0.2.15
no port[s] to connect to

(kali@kali):~/Desktop
$ netcat 127.0.0.1
no port[s] to connect to

(kali@kali):~/Desktop
$
```