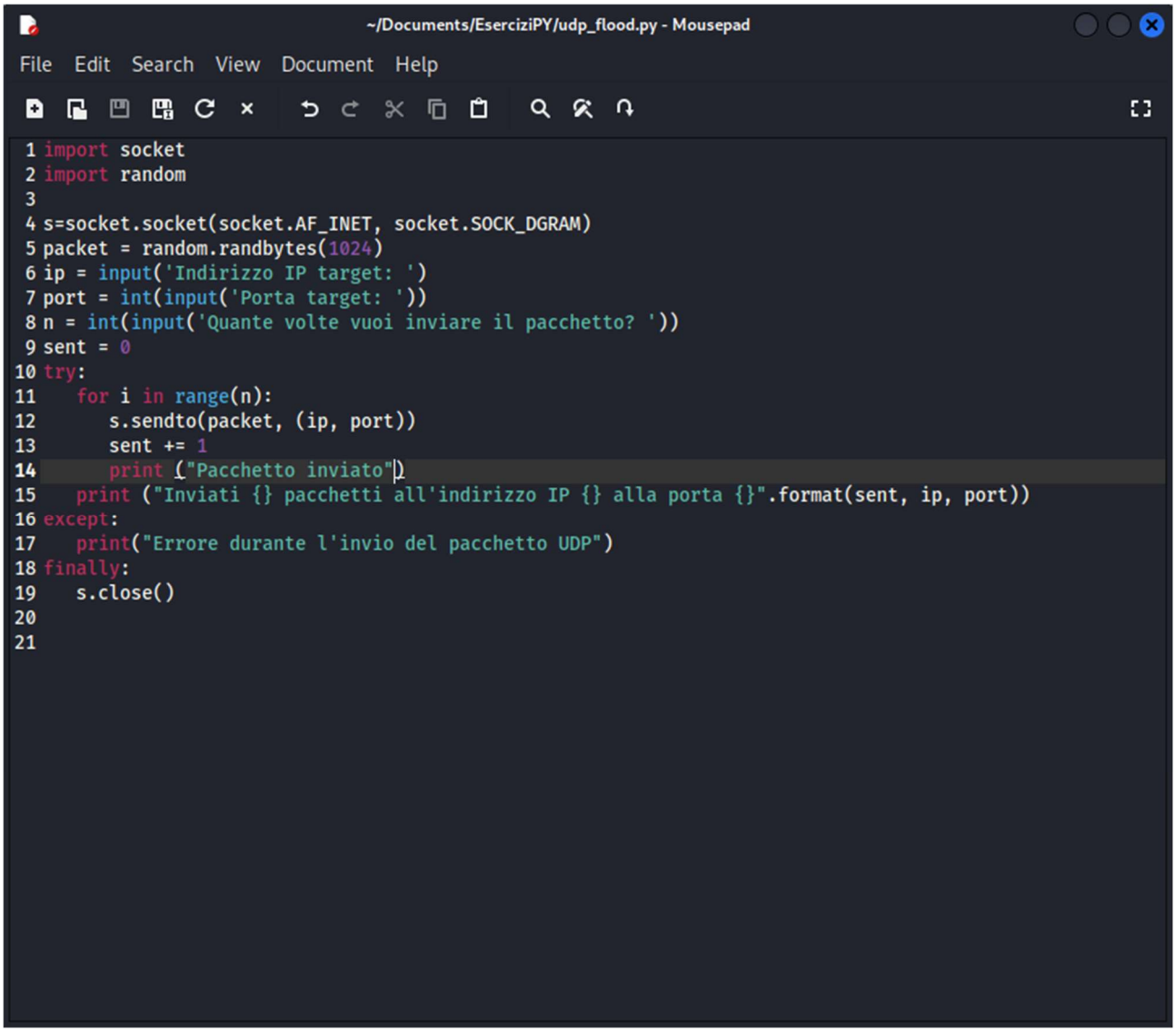


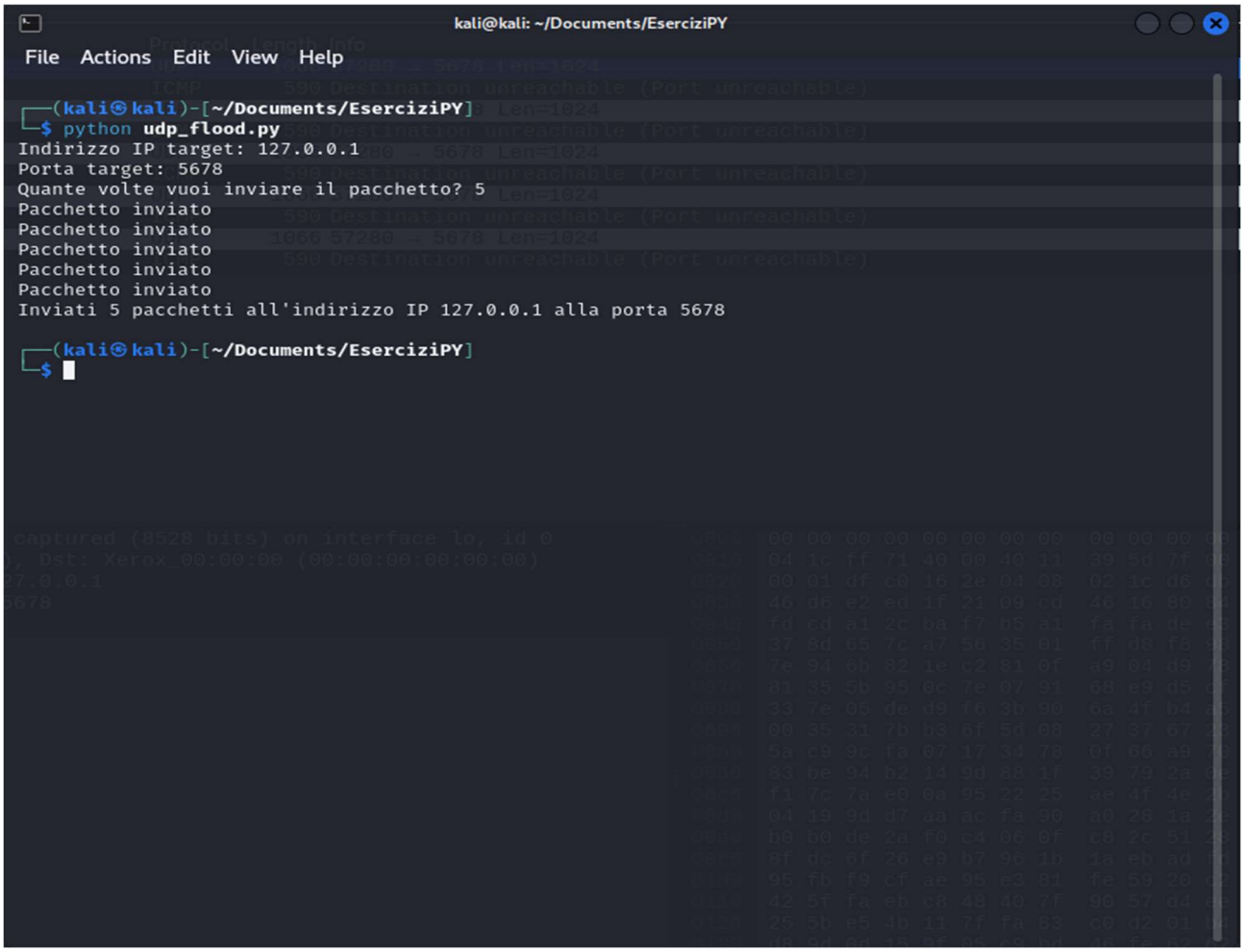
Questo è il codice per eseguire **l'udp flood**.

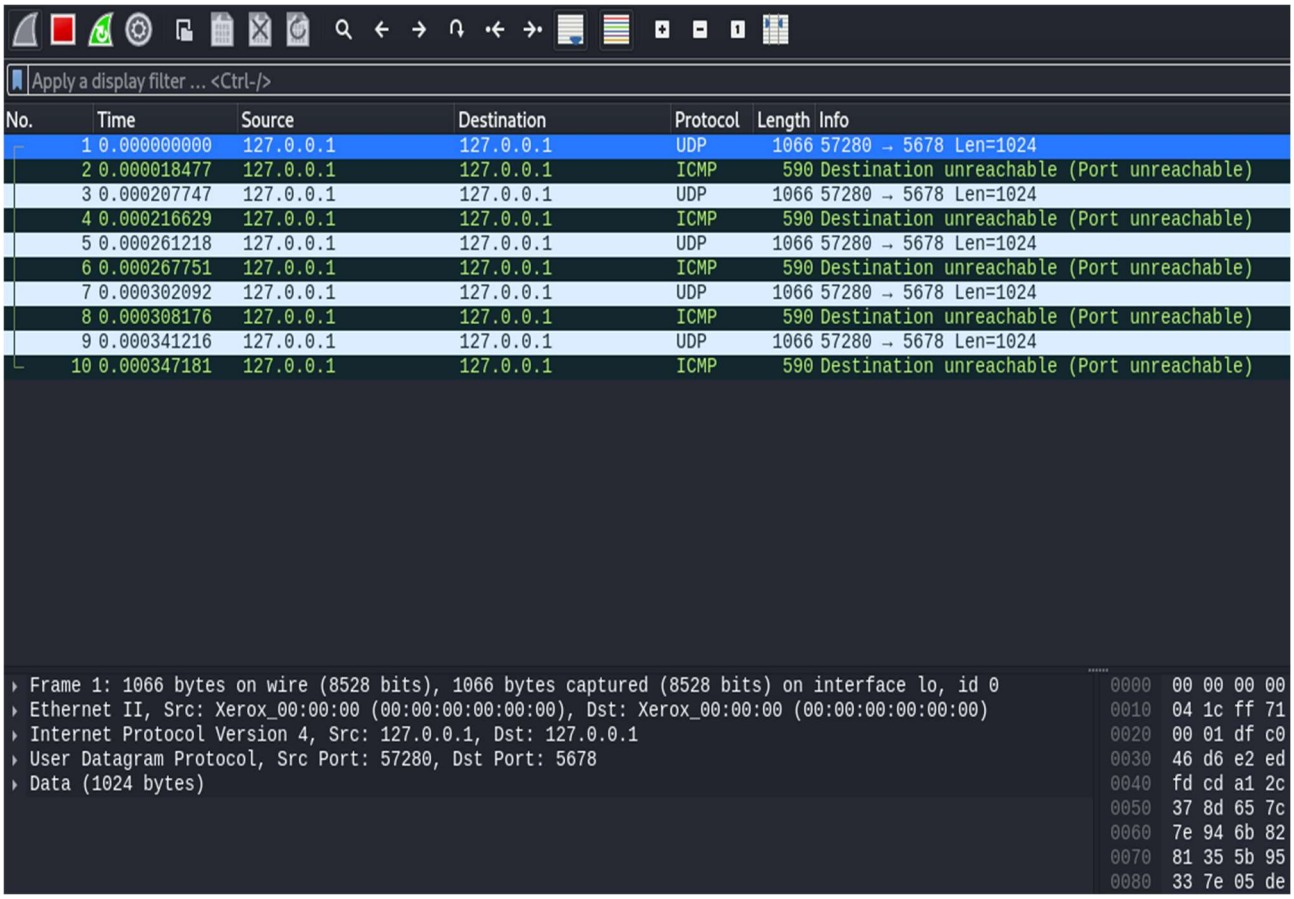
Si noti come **socket.socket** crea un nuovo socket (nel nostro caso chiamato s) inserendo dei parametri; il primo AF_INET specifica che vogliamo un socket che utilizzi IPv4, il secondo SOCK_DGRAM specifica che vogliamo una connessione *UDP*.

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]
$


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 0.000018477 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.