**Esercizio Programmazione per Hacker**

**Attacco DoS**

Cosa è un attacco DoS: Un **attacco DoS** (Denial of Service, **Negazione del Servizio**) è una forma di attacco informatico in cui un aggressore cerca di rendere un sistema, un servizio o una rete **non disponibile** per gli utenti legittimi. L’obiettivo principale di un attacco DoS è interrompere o rallentare il normale funzionamento di un server, una rete o una risorsa online, sovraccaricando il sistema di richieste in modo che non possa rispondere correttamente alle richieste legittime.

**Obiettivo 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.

1. **Importazione dei Moduli**

import socket

import random

import time

• **socket**: Questo modulo consente di creare connessioni di rete e di inviare pacchetti. È essenziale per qualsiasi tipo di comunicazione di rete in Python.

• **random**: Utilizzato per generare dati casuali. In questo programma, serve a riempire i pacchetti di byte casuali, simulando del traffico di rete generico.

• **time**: Serve per introdurre un ritardo (sleep) tra l’invio di un pacchetto e l’altro, in modo che l’invio sia più graduale e meno aggressivo.

**2. Definizione della Funzione udp\_flood ()**

Tutta la logica del programma si trova all’interno di questa funzione.

def udp\_flood():

    # Chiedi all'utente di inserire l'IP e la porta target

    ip\_target = input("Inserisci l'IP della macchina target: ")

    porta\_target = int(input("Inserisci la porta UDP della macchina target (ad es. 8080): "))

    numero\_pacchetti = int(input("Inserisci il numero di pacchetti da 1 KB da inviare: "))

**3. Raccolta degli Input dall’Utente**

• **ip\_target**: Chiede all’utente di inserire l’indirizzo IP della macchina target (la macchina verso cui verranno inviati i pacchetti).

• **porta\_target**: Chiede all’utente di inserire la porta UDP della macchina target. La porta è come una “porta di ingresso” per la comunicazione di rete sulla macchina target. In questo caso, potrebbe essere 8080 o qualsiasi altra porta su cui il target sta ascoltando.

• **numero\_pacchetti**: Chiede all’utente di specificare il numero di pacchetti da inviare. Questo permette di regolare l’intensità dell’invio.

**4. Creazione del Socket UDP**

    sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

• Qui viene creato un **socket** di tipo UDP.

• **socket.AF\_INET** specifica che stiamo utilizzando un indirizzo IPv4 (quindi un indirizzo come 192.168.1.149).

• **socket.SOCK\_DGRAM** indica che stiamo usando il protocollo **UDP** (User Datagram Protocol), che è un protocollo “senza connessione”. A differenza del protocollo TCP, UDP non richiede una connessione stabile, quindi i pacchetti vengono inviati direttamente senza attendere una conferma di ricezione.

**5. Costruzione del Pacchetto di 1 KB**

    pacchetto = bytes(random.getrandbits(8) for \_ in range(1024))  # 1024 byte = 1 KB

• Questo crea un pacchetto di **1 KB** (1024 byte) di dati casuali.

• **random.getrandbits(8)** genera un byte casuale (8 bit) alla volta. Il ciclo **for \_ in range(1024)** lo ripete 1024 volte per creare un array di byte casuali.

• **bytes(...)** converte l’array di byte in un formato che può essere inviato attraverso la rete. Il pacchetto generato sarà quindi costituito da dati casuali, simulando traffico generico.

**6. Invio dei Pacchetti con Ritardo**

    for i in range(numero\_pacchetti):

        sock.sendto(pacchetto, (ip\_target, porta\_target))

        print(f"Pacchetto {i+1} inviato a {ip\_target}:{porta\_target}")

        time.sleep(0.01)  # Ritardo di 10 millisecondi tra i pacchetti

• **Ciclo for**: Il ciclo invia il numero di pacchetti specificato dall’utente.

• **sock.sendto(pacchetto, (ip\_target, porta\_target))**: Questo comando invia il pacchetto di 1 KB all’indirizzo e alla porta specificati. **(ip\_target, porta\_target)** è una tupla che rappresenta il target dell’invio.

• **print(...)**: Stampa un messaggio di conferma per ogni pacchetto inviato, utile per vedere i progressi dell’invio.

• **time.sleep(0.01)**: Introduce un ritardo di **10 millisecondi** (0.01 secondi) tra un pacchetto e il successivo. Questo rallenta un po’ il ritmo dell’invio, evitando che il target sia sovraccaricato troppo velocemente.

**7. Chiusura del Socket**

    sock.close()

    print("Invio completato.")

• **sock.close()** chiude il socket una volta completato l’invio di tutti i pacchetti, liberando le risorse.

• **print("Invio completato.")** stampa un messaggio per indicare che l’operazione è terminata.

**8. Esecuzione del Programma**

udp\_flood()

• Questa riga richiama la funzione udp\_flood(), avviando il programma. Quando l’utente esegue lo script, questa è la parte che dà inizio al processo di raccolta input e invio pacchetti.

**Riepilogo del Programma**

1. Chiede all’utente di inserire l’IP target, la porta UDP e il numero di pacchetti da inviare.

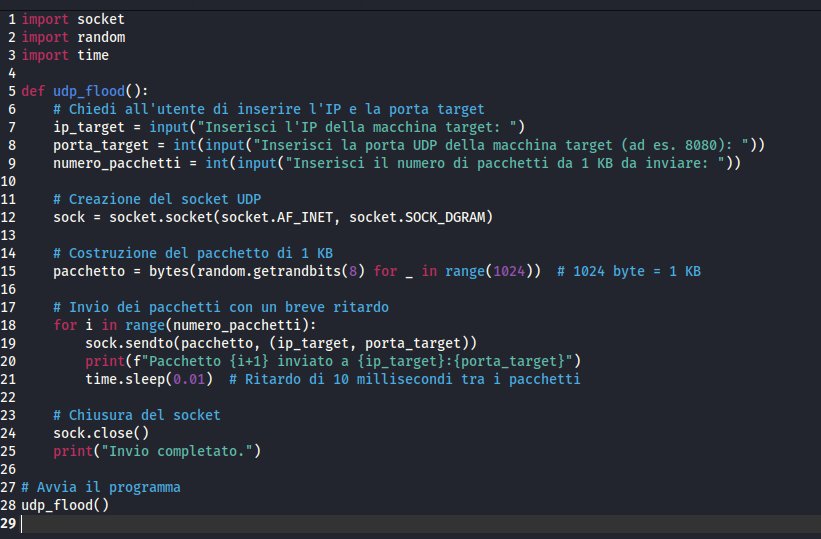
2. Crea un socket UDP e costruisce un pacchetto di 1 KB con dati casuali.

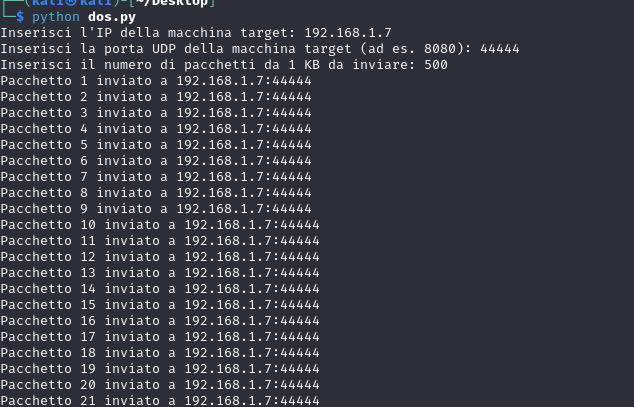
3. Invia il numero specificato di pacchetti al target con un breve ritardo tra un pacchetto e l’altro.

4. Chiude il socket e segnala la fine dell’invio.

**Note Finali**

Questo programma è un esempio semplice di come inviare pacchetti UDP in modo programmato, utile per capire le basi dei protocolli di rete. Come sempre, usa questo programma solo in ambienti di test controllati o su dispositivi di cui hai il controllo, poiché inviare pacchetti UDP senza autorizzazione può essere considerato un attacco DoS (Denial of Service) in un contesto reale.





**Cosa Vedi su Wireshark**

1. **Interfaccia Principale di Wireshark**

• Wireshark mostrerà una lista di pacchetti che rappresentano le comunicazioni tra il tuo computer e l’IP target.

• Ogni riga rappresenta un singolo pacchetto catturato.

• La colonna **No.** indica il numero progressivo dei pacchetti.

• **Time** mostra il timestamp relativo alla cattura di ciascun pacchetto.

• **Source** e **Destination** indicano gli indirizzi IP del mittente e del destinatario del pacchetto.

• **Protocol** mostrerà “UDP” per questi pacchetti, poiché il programma usa il protocollo UDP.

• **Length** è la dimensione del pacchetto, che sarà di circa 1024 byte come definito nello script.

• **Info** può mostrare ulteriori dettagli, come la porta di origine e destinazione.



**Conclusione**

Questo programma Python implementa un semplice attacco di **UDP flood**, in cui vengono inviati pacchetti UDP casuali a una macchina target attraverso una specifica porta. Sebbene il codice sia utile per comprendere il funzionamento delle connessioni di rete e dei pacchetti UDP, è fondamentale sottolineare che l’uso di questo script in ambienti non autorizzati può essere considerato illegale e dannoso, poiché può sovraccaricare una rete o un server, causando interruzioni del servizio (DoS).

L’uso di strumenti come Wireshark consente di osservare in tempo reale il traffico di rete generato, evidenziando come i pacchetti UDP vengano inviati al target con la dimensione predefinita di 1 KB e con un ritardo tra i pacchetti, simulando un flusso continuo di traffico. Questo programma, pur essendo un esempio educativo, dovrebbe essere eseguito esclusivamente in contesti controllati per evitare di danneggiare altre infrastrutture o violare leggi e regolamenti.

Infine, la comprensione del funzionamento di questi attacchi aiuta a migliorare la consapevolezza della sicurezza informatica e la protezione delle reti, con l’obiettivo di prevenire o mitigare minacce simili.