ESERCIZIO PROGRAMMAZIONE

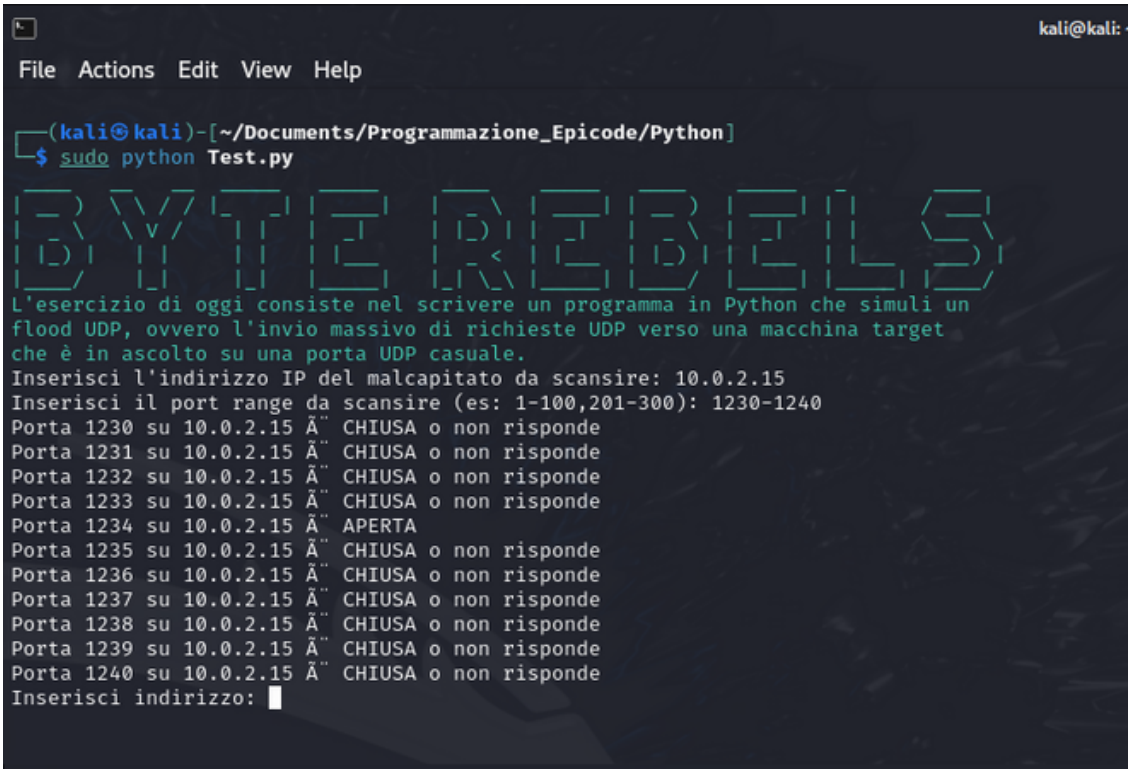
BYTE REBELS

UDP FLOOD

CANNAVACCIUOLO DAVIDE
DI MAIO PAOLO
FORLENZA SIMONE
RUSSO FEDERICO - LEADER
TIZZI FEDERICO
VAN ZWAM ARJEN

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**.

02



UDP FLOOD CLIENT .PY

Abbiamo progettato questo programma con due funzionalità principali, port scanner e UDP Packet Sender.
Port Scanner: Consente di eseguire una scansione delle porte su un determinato indirizzo IP per determinare quali porte sono aperte e quali sono chiuse o non rispondono. L'utente può specificare un range di porte da scansionare e l'indirizzo IP del target. Il programma utilizza thread per eseguire la scansione in modo più efficiente. Poi si passerà alla funzione vera e propria del progetto, il flood.
Il programma utilizza il modulo socket di Python per comunicare tramite socket, nonché il modulo threading per gestire i thread per le operazioni concorrenti.

A cosa serve?

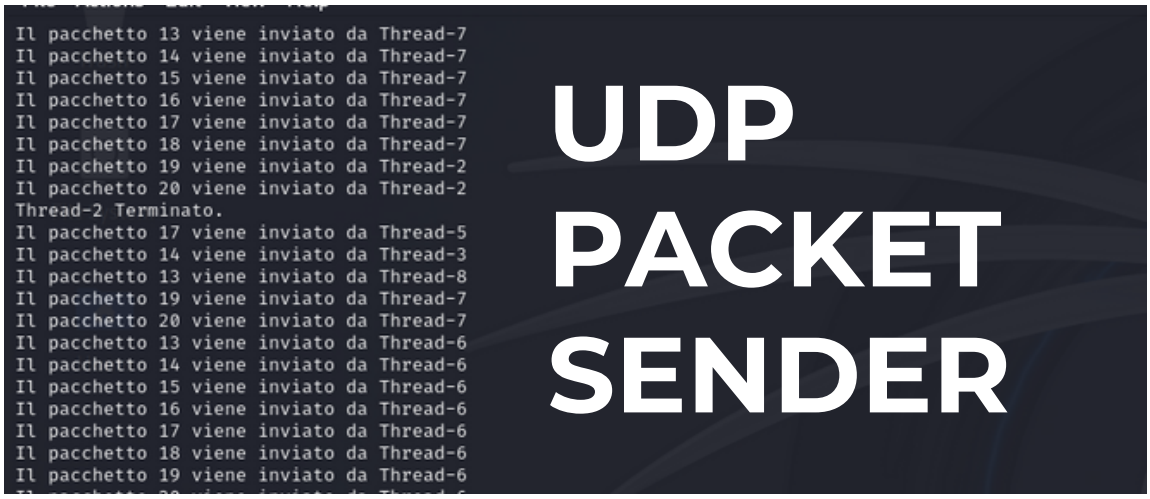
Consente di eseguire una scansione delle porte su un determinato indirizzo **IP** per determinare quali porte sono aperte e quali sono chiuse o non rispondono.
L'utente, inoltre, può specificare un range di porte da scansionare e l'indirizzo **IP** del target. Il programma utilizza *thread* per eseguire la scansione in modo più efficiente.

Utilizzo

All'utente viene chiesto di inserire l'indirizzo IP del target da scansire. · Successivamente, l'utente inserisce il range di porte da scansionare, specificando la porta minima e la porta massima desiderata separate da un trattino. Ad esempio: 1-100.
Il programma esegue la scansione delle porte e stampa i risultati in ordine numerico, comunicando un feedback.

Scopo

Trovare le porte aperte della vittima



04

A cosa serve?

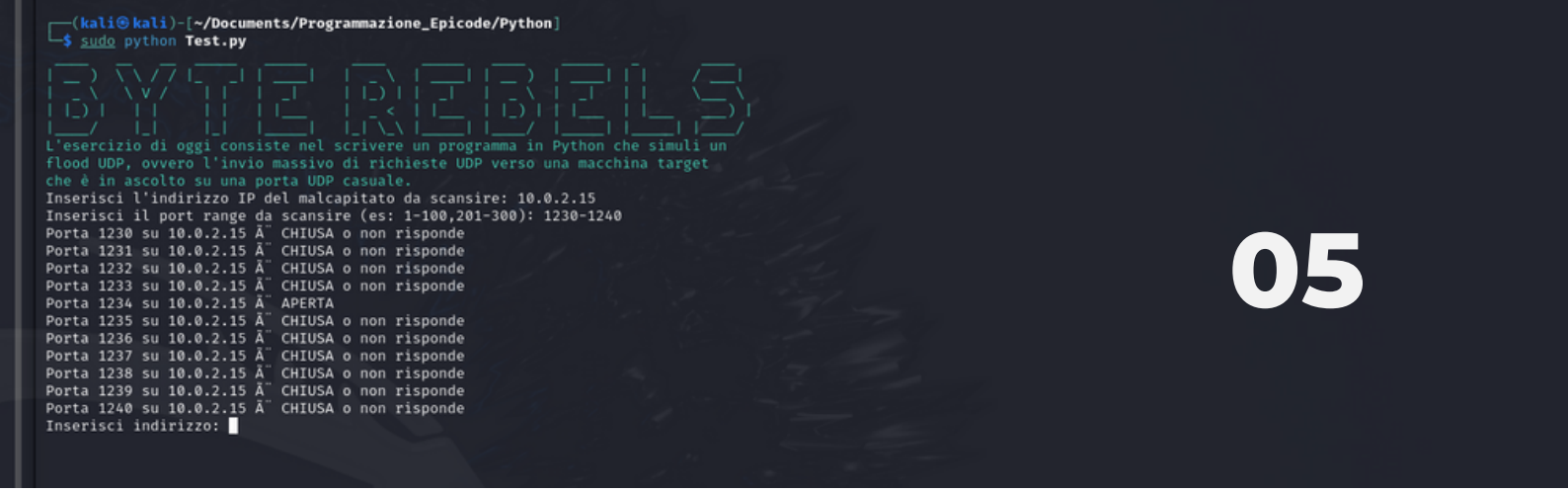
Permette di inviare un numero *specificato* di **pacchetti UDP** ad un determinato indirizzo IP e porta associata. L'utente può specificare l'indirizzo IP di destinazione (*vittima*), la porta ed il numero di pacchetti da inviare. Anche questa funzionalità utilizza *thread* per inviare i pacchetti in parallelo.

Utilizzo

Dopo aver individuato la porta tramite il port scanner all'utilizzatore viene chiesto di inserire l'indirizzo IP della vittima per l'invio dei **pacchetti UDP.**
Quindi, l'utente inserisce il numero di porta e il numero di pacchetti UDP da inviare.
Il programma avvia un certo numero di *threads* per inviare i pacchetti UDP in **parallelo**.

Scopo

Intasare di pacchetti fittizzi l'**IP** target



05

ANALISI: Client - Flood.py

1. Il programma inizia chiedendo l'indirizzo IP Target
2. Inseriremo dunque il range di porte che si vogliono scansire
3. Inserire l'indirizzo della vittima
4. Inserire la porta acquisita precedentemente
5. Inserimento della quantità di pacchetti UDP da mandare

ANALISI: Server.py

Abbiamo semplicemente creato un mini server si mette in ascolto UDP nella sua porta(abbiamo ipotizzato 1234).

06

PILLOLE DI SCRIPT

IMPORT

Abbiamo importato le varie librerie per il corretto svolgimento.

DEF FUNZIONI

Abbiamo dunque definito le funzioni send_udp_packets, la def port_scan_range, def port_scanner e ovviamente la main

IMPORTANTE

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

SOCK_**DGRAM** per indicare il protocollo **UDP** e non TCP

data = bytearray(random.getrandbits(8) ***for*** *_ in range(1024))*

Quindi: 8 è 1Byte, **for** _ in range(1024): per arrivare fino a 1024 byte

L'utilizzo dei SOCKET in più punti è stato volutamente introdotto per renderlo più rapido ed efficiente.