LISA BONATO

ROSARIO ZAPPALA’

**TASK 02 - PORT SCANNING**

Una delle richieste dell’azienda Theta è quella di creare un programma Python per la valutazione dei servizi attivi sul web server.

Per rispondere alla richiesta dell’azienda e quindi per fare la scansione dei servizi, noi abbiamo scritto in linguaggio Python un vulnerability scanner che accetti in input un IP ed un range di porte da scansionare e che restituisca come output una lista delle porte con stato “aperto”.

**Creazione del codice**

Il codice è stato creato utilizzando il seguente processo:

1.Importazione del modulo socket

2.Richiesta all'utente dell'indirizzo IP del server e dell'intervallo di porte da scansionare

3.Conversione dell'intervallo di porte da stringa a numeri interi

4.Iniziazione della scansione delle porte

5.Stampa del risultato della scansione.

Abbiamo implementato il seguente codice (in cui abbiamo inserito i commenti necessari):

**import socket**

#IMPORTA IL MODULO SOCKET che fornisce una libreria di funzioni per la creazione e l'utilizzo di socket

**target = input('Inserire un indirizzo IP da scansionare: ‘)**

#Richiede all’utente di inserire l’indirizzo IP del server da scansionare

**portrange = input('Inserire un intervallo di porte da scansionare(es 5-200): ')**

#Richiede all'utente di inserire l'intervallo di porte da scansionare. L'intervallo di porte viene inserito come stringa, ma viene convertito in numeri interi prima di essere utilizzato

**lowport = int(portrange.split('-')[0])**

#Assegna il valore del primo numero dell'intervallo di porte alla variabile lowport

**highport = int(portrange.split(‘-')[1])**

#Assegna il valore del secondo numero dell'intervallo di porte alla variabile highport

**print('\nScannerizzando le porte dalla ' , lowport,' alla' , highport, ' del server con IP ' , target)**

#Stampa un messaggio a schermo che indica che la scansione delle porte sta per iniziare

**for port in range (lowport, highport+1):**

#Inizia un ciclo for che esegue la scansione di tutte le porte nell'intervallo specificato

**s = socket.socket (socket.AF\_INET, socket.SOCK\_STREAM)**

#CREA UN NUOVO SOCKET TCP

**status = s.connect\_ex ((target, port))**

#Tenta di connettersi al server sull'indirizzo IP e sulla porta specificati

**if (status == 0):**

#Verifica se la connessione è riuscita. Se è riuscita, il valore di status è 0

**print('\nPorta', port, '-APERTA')**

#Stampa a schermo un messaggio che indica che la porta è aperta

**s.close()**

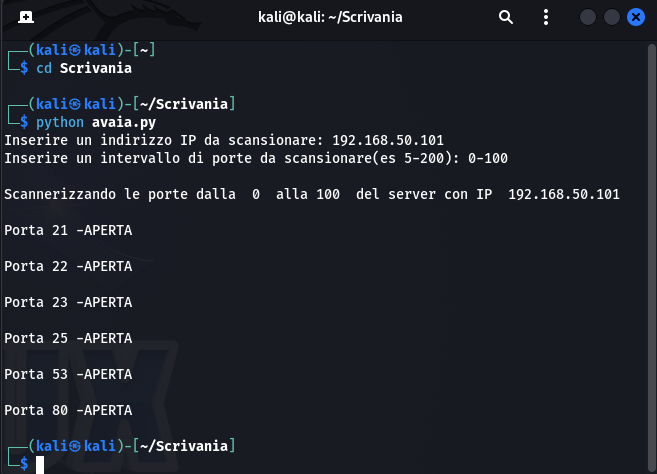
#CHIUDE IL SOCKET

**Funzionamento del codice**

Il codice è un programma in Python che esegue una scansione delle porte di un server. Il programma inizia importando il modulo socket, che fornisce una libreria di funzioni per la creazione e l'utilizzo di socket.

In seguito, il programma richiede all'utente di inserire l'indirizzo IP del server da scansionare e l'intervallo di porte da scansionare, in formato standard esempio 5-200. L'intervallo di porte viene inserito come stringa, ma viene convertito in numeri interi prima di essere utilizzato. Per lavorare sulla stringa, abbiamo utilizzato il metodo «split» e utilizzato il simbolo «-» come separatore.

Il ciclo «for» successivo ci serve per tentare la connessione TCP ad ogni porta nell’intervallo specificato. Utilizziamo il metodo «socket.socket» che ci restituisce un socket che chiamiamo «s». I parametri della socket che abbiamo scelto sono IPv4 e TCP.

La funzione successiva, «s.connect\_ex» tenta la connessione alla coppia IP:PORTA specificata. Il risultato della scansione viene stampato a schermo e ci restituisce solo lo stato «0»: la connessione è andata a buon fine, dunque possiamo dedurre che la porta è aperta. In caso contrario, la porta è chiusa.

**Limitazioni del codice**

Il codice ha alcune limitazioni:

* Esegue una scansione delle porte in modo sequenziale. Questo può essere inefficiente per intervalli di porte molto grandi.
* Non fornisce informazioni aggiuntive sulle porte aperte, come i servizi che vi sono in ascolto.

**Possibili miglioramenti**

* Utilizzare un algoritmo di scansione delle porte parallelo, per ridurre il tempo necessario per eseguire la scansione.
* Aggiungere funzioni per raccogliere informazioni aggiuntive sulle porte aperte.

**Conclusioni**

Abbiamo deciso di inserire solo la stampa finale delle porte aperte così da avere una visione chiara di ciò che ci serve visualizzare. Le porte aperte sono infatti un importante punto di accesso con cui un utente malintenzionato può entrare nel sistema e sfruttare le vulnerabilità presenti.

Il codice quindi, oltre ad essere utile per identificare potenziali vulnerabilità, è anche utilizzato per riuscire a proteggere il sistema da attacchi informatici.