

Compito di oggi:

1) spiegare cos'è una backdoor e perchè è pericolosa.

2) Spiegare i codici qui sotto dicendo cosa fanno e qual è la differenza tra i due.

Opzionale (consigliato) testare praticamente il codice

Una "**backdoor**" è una *via di accesso segreta o nascosta* in un sistema informatico, software o dispositivo che consente a un utente autorizzato o non autorizzato di aggirare le normali procedure di autenticazione e ottenere l'accesso al sistema o alle sue funzionalità in modo non intenzionale. In altre parole, una backdoor è una porta secondaria che bypassa le normali misure di sicurezza.

Le backdoor possono essere deliberate o accidentali.

Le **backdoor deliberate** sono spesso create da sviluppatori di software o da individui malevoli per scopi nefasti. Ad esempio, un malware potrebbe installare una backdoor in un sistema per consentire agli hacker di accedere e controllare il sistema senza il consenso dell'utente legittimo.

Le **backdoor accidentali** possono verificarsi a causa di errori di programmazione o di configurazione. Questi errori possono creare vulnerabilità non intenzionali che possono essere sfruttate da terze parti per ottenere accesso non autorizzato.

Le backdoor possono comportare diversi pericoli, a seconda di come vengono utilizzate e implementate. Ecco alcuni dei principali rischi associati alle backdoor:

1. Accesso non autorizzato:

- La backdoor fornisce un accesso non autorizzato al sistema o al software in cui è implementata. Questo può permettere a un attaccante di compromettere la sicurezza del sistema, accedere a dati sensibili o eseguire azioni dannose.

2. Violenza della privacy:

- Se una backdoor è utilizzata malevolmente, può essere impiegata per monitorare le attività degli utenti senza il loro consenso. Ciò può portare a gravi violazioni della privacy, con conseguente accesso indebito a informazioni personali e riservate.

3. Attacchi mirati:

- Le backdoor possono essere utilizzate come parte di attacchi mirati, in cui gli attaccanti sfruttano questa via di accesso segreta per condurre operazioni di spionaggio, furto di informazioni aziendali o governative, o sabotaggio.

4. Danni ai dati e alle risorse:

- Gli utenti malevoli possono utilizzare backdoor per danneggiare o cancellare dati critici, interrompere le operazioni di sistema o causare danni irreparabili alle risorse digitali.

5. Diffusione di malware:

- Le backdoor sono spesso utilizzate come parte di malware più ampi. Una volta che una backdoor è installata su un sistema, può essere sfruttata per l'introduzione di ulteriori malware o per facilitare l'espansione di una minaccia informatica.

6. Minaccia persistente avanzata (APT):

- Le backdoor sono spesso coinvolte in attacchi APT, in cui gli attaccanti cercano di mantenere l'accesso non rilevato per lunghi periodi. Ciò può consentire loro di raccogliere informazioni sensibili nel tempo e di eseguire azioni dannose senza essere scoperti.

7. Rischi legali e reputazionali:

- L'installazione di backdoor può comportare conseguenze legali significative per gli sviluppatori di software o per coloro che implementano tali meccanismi senza il consenso degli utenti. Inoltre, può danneggiare la reputazione di un'organizzazione o di un individuo.

CODICI

Il primo programma è un semplice server che permette l'ingresso tramite una backdoor (porta 1234) alla macchina client che utilizzerà il codice 2 per vedere la tipologia del sistema operativo oppure il contenuto della directory che si immette.

Codice server

```
File Edit Search View Document Help

provasocket1.py

1 import socket, platform, os #moduli importati per sfruttarne le funzionalità
2
3 SVR_ADDR = "" #richiesta all'utente di inserire IP del client = variabile IP
4 SVR_PORT = 1234 #richiesta all'utente di inserire porta per la connessione = variabile porta
5
6 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #la funzione socket.socket restituisce crea un nuovo socket, 's', utilizzando IPV4 e connessione TCP
7 s.bind((SVR_ADDR, SVR_PORT)) #collegamento del socket 's' in ascolto con metodo bind(), associando socket alle variabili IP e porta
8 s.listen(1) #metodo listen configura il socket 's' in ascolto sulla coppia IP:porta. (1) = backlog, indica numero massimo di connessioni in coda
9 connection, address = s.accept() #metodo accept per accettare e prendere la connessione con con il client.
10
11 while 1: #ciclo while, che è sempre vero, per far partire lo scambio dati server - client. In pratica gestisce le richieste del cliente.
12     try: #il costrutto try-except:continue serve in caso di errore nella comunicazione dei dati per permettere di far continuare il programma.
13         data= connection.recv(1024)
14     except:continue
15
16     if(data.decode('utf-8') == '1'): #se i dati inviati dal cliente=1, il server creato dal programma invia dati sotto forma di stringa circa: l
17         tosend = platform.platform() + " " + platform.machine() #la piattaforma su cui programma viene eseguito (sist. op.) e architettura(tipo di macchina)
18         connection.sendall(tosend.encode()) #invia messaggio a tutti i partecipanti alla comunicazione circa il fatto che la connessione è riuscita=mex arrivato.
19     elif(data.decode('utf-8') == '2'): #se i dati inviati dal client=2, cerca lista file nella directory specificata dai dati ricevuti e invia lista al client.
20         (data = connection.recv(1024))
21         try:
22             filelist = os.listdir(data.decode('utf-8'))
23             tosend = ""
24             for x in filelist:
25                 tosend += " " + x
26         except:
27             tosend = "wrong path"
```

```
File Edit Search View Document Help

provasocket1.py

connection, address = s.accept() #metodo accept per accettare e prendere la connessione con con il client.

while 1: #ciclo while, che è sempre vero, per far partire lo scambio dati server - client. In pratica gestisce le richieste del cliente.
    try: #il costrutto try-except:continue serve in caso di errore nella comunicazione dei dati per permettere di far continuare il programma.
        data= connection.recv(1024)
    except:continue

    if(data.decode('utf-8') == '1'): #se i dati inviati dal cliente=1, il server creato dal programma invia dati sotto forma di stringa circa: l
        tosend = platform.platform() + " " + platform.machine() #la piattaforma su cui programma viene eseguito (sist. op.) e architettura(tipo di macchina)
        connection.sendall(tosend.encode()) #invia messaggio a tutti i partecipanti alla comunicazione circa il fatto che la connessione è riuscita=mex arrivato.
    elif(data.decode('utf-8') == '2'): #se i dati inviati dal client=2, cerca lista file nella directory specificata dai dati ricevuti e invia lista al client.
        (data = connection.recv(1024))
        try:
            filelist = os.listdir(data.decode('utf-8'))
            tosend = ""
            for x in filelist:
                tosend += " " + x
        except:
            tosend = "wrong path"
        connection.sendall(tosend.encode())
    elif(data.decode('utf-8') == '0'): #Se i dati sono '0', chiude la connessione corrente e aspetta una nuova connessione.
        connection.close()
        #I dati inviati dal client vengono ricevuti (fino a 1024 byte)
        connection, address = s.accept()
```

Codice client

```
~/Desktop/python/provasocket2.py - Mousepad
File Edit Search View Document Help
provasocket1.py provasocket2.py x

1 import socket ##modulo importati per sfruttarne le funzionalità
2
3 SRV_ADDR =input("Type the server IP address: ") #dichiarazione delle variabili del server richieste all'utente: Ip e porta del server
4 SRV_PORT =int(input("Type the server port: "))
5
6 def print_menu(): #funzione che presenta menu all'utente per poter interagire con il server, chiede che tipo di dati vuole chiedere.
7     print("\n\n0) Close the connection
8 1) Get system info
9 2) List directory contents")
10
11 my_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #creazione del socket che si mette in ascolto tramite IP:porta
12 my_sock.connect((SRV_ADDR, SRV_PORT)) #il socket si connette con il server attraverso la coppia IP:porta
13
14 print("Connection established") #viene stampato a video il fatto che la connessione è stabilita.
15 print_menu() #viene stampato a video il menù delle opzioni.
16
17 while 1: #con il ciclo while il programma entra in un ciclo infinito in cui l'utente può inserire le opzioni del menu.
18     message = input("\n-Select an option: ")
19
20     if(message == 0): #questo blocco serve a gestire le scelte dell'utente.
21         my_sock.sendall(message.encode()) #Se utente sceglie=0, invia il messaggio al server per chiudere la connessione e termina il programma.
22         my_sock.close()
23         break
24
25     elif(message == "1"): #Se l'utente sceglie=1, invia mex al server e riceve risposta contenente informazioni di sistema, poi stampa la risposta.
26         my_sock.sendall(message.encode())
27         data = my_sock.recv(1024)
```

```
~/Desktop/python/provasocket2.py - Mousepad
File Edit Search View Document Help
provasocket1.py provasocket2.py x

17 while 1: #con il ciclo while il programma entra in un ciclo infinito in cui l'utente può inserire le opzioni del menu.
18     message = input("\n-Select an option: ")
19
20     if(message == 0): #questo blocco serve a gestire le scelte dell'utente.
21         my_sock.sendall(message.encode()) #Se utente sceglie=0, invia il messaggio al server per chiudere la connessione e termina il programma.
22         my_sock.close()
23         break
24
25     elif(message == "1"): #Se l'utente sceglie=1, invia mex al server e riceve risposta contenente informazioni di sistema, poi stampa la risposta.
26         my_sock.sendall(message.encode())
27         data = my_sock.recv(1024)
28         if not data: break
29         print(data.decode('utf-8'))
30
31     elif(message == "2"): #se utente sceglie=2,
32
33         path = input("Insert the path: ") #il codice richiede un percorso utente, lo invia al server, riceve e visualizza la lista dei file restituiti dal server.
34         my_sock.sendall(message.encode())
35         my_sock.sendall(path.encode())
36         data = my_sock.recv(1024)
37         data = data.decode('utf-8').split(",")
38         print("***40")
39         for x in data:
40             print(x)
41         print("***40")
42
43
```