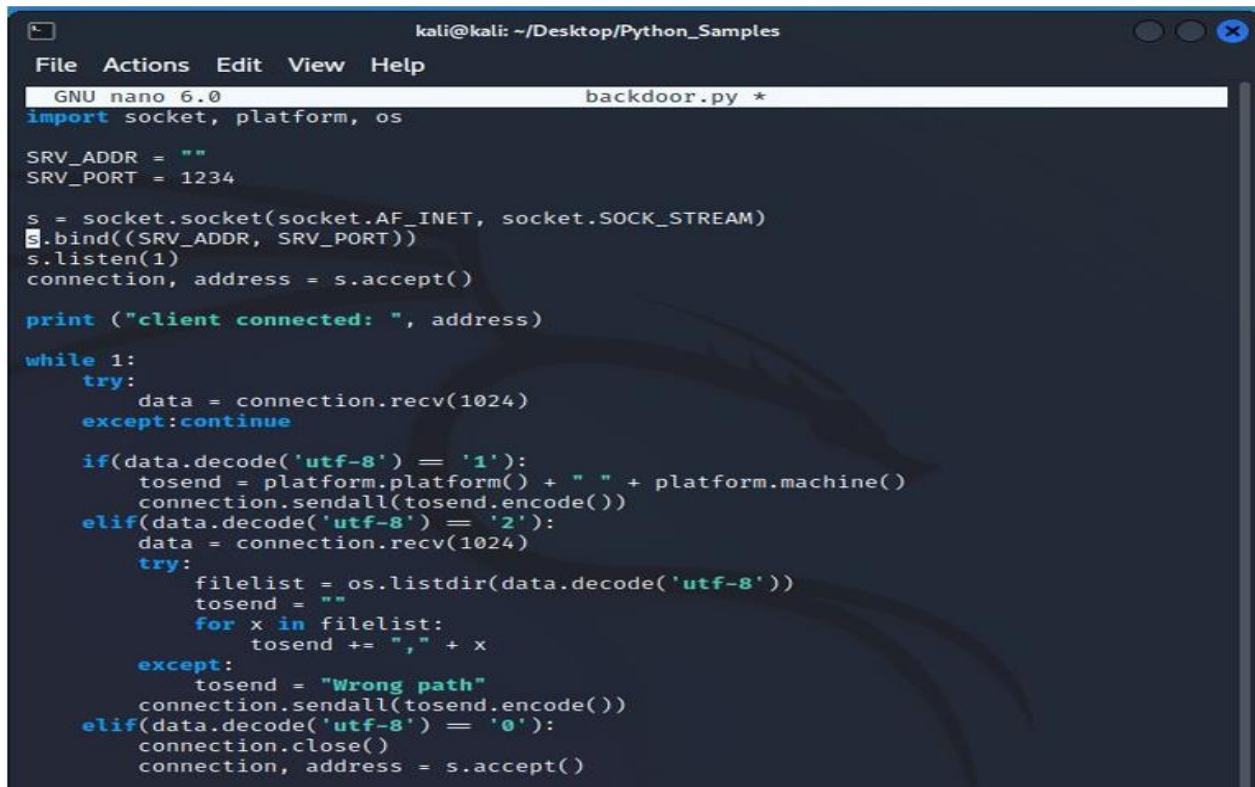


L'esercizio di oggi consiste nel commentare/spiegare questo codice che fa riferimento ad una backdoor. Inoltre, spiegare cos'è una backdoor.

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 listens for incoming connections on a specified IP and port (1234). It handles three types of commands: '1' to send system information, '2' to list files in a directory, and '0' to close the connection. The script uses the socket module for network communication and the platform module for system details.

```
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()
```

Una backdoor è una porta d'accesso sul retro, più specificatamente righe di codice create tramite l'utilizzo di linguaggi di programmazione che permettono a un utente remoto di accedere con i permessi di amministratore. Queste porte d'accesso "secondarie" possono essere implementate da sistemisti o gestori di reti, per eseguire manutenzioni delle infrastrutture di rete oppure per ripristinare le password dimenticate dagli utenti. Le backdoor possono essere inserite anche per scopi malevoli, tramite malware, da black hat che le utilizzano, tra questi malware c'è il RAT (Remote Access Control Trojan), che permette al malintenzionato di prendere il controllo totale del sistema informatico della vittima.

Import socket platform os: importiamo il modulo (in C librerie), socket che fornisce le funzionalità necessarie per la comunicazione di rete;

SRV\_ADDR = " contiene l'indirizzo IP del server";

SRV\_PORT = " contiene il numero di porta su cui il server riceverà le connessioni in ingresso"(in questo caso 1234);

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM): Crea un socket utilizzando socket.socket(). Il primo parametro socket.AF\_INET specifica che si

tratta di un socket di tipo IPv4, e il secondo parametro `socket.SOCK_STREAM` specifica che si tratta di un socket TCP, (un socket è un dispositivo software astratto, che permette l'invio e la ricezione di dati in una rete, sintetizzato al massimo IP e porta);

`s.bind((SRV_ADDR, SRV_PORT))`: Associa il socket all'indirizzo IP e alla porta specificati con `bind()`, così il server sarà in ascolto su quell'indirizzo IP e porta per le connessioni in ingresso;

`s.listen(1)`: il comando `listen()` permetterà al socket di entrare in modalità di ascolto. Il parametro 1 indica che il server può accettare una sola connessione in entrata alla volta (è importante perché nelle aziende questi parametri permetteranno di difendersi da attacchi DDOS);

`print ("client connected: ", address)`: stamperà un messaggio, quindi visualizzeremo un messaggio che ci dice che il client è connesso, e l'IP del client;

`while 1`: entra in un ciclo while infinito per ricevere dati dal client;

`try`: nel caso tutto andasse nel modo prestabilito senza nessun tipo di problematiche il codice verrà eseguito;

`data = connection.recv(1024)`: con questo comando si possono stabilire il numero di byte ricevuti dal server (in questo caso un kilobyte, anche se le condizioni dipendono dalla velocità di connessione);

`except continue`: eviterà che il programma vada in crash, e manderà un messaggio informativo, un segnale in output questo permetterà di gestire meglio le eccezioni o imprevisti (in questo caso continua);

`if(data.decode('utf-8') == '1')`: Questa riga decodifica, alcuni dati binari ( **data**) utilizzando la codifica UTF-8 (utf-8 standard che utilizziamo per andare sul web, le lettere vengono associate al codice ASCII, formato da codice binario, che verrà letto e trasformato dalla CPU, in lettere) si confrontano i dati decodificati con la stringa 1 se i dati sono uguali alla stringa 1 verrà eseguito il blocco di codice nella stringa 1;

`tosend = platform.platform() + " " + platform.machine ()`:

`Connection.sendall(tosend.encode() )`: Se il dato decodificato è '1', questa riga crea una stringa ( **tosend**). `'platform.platform()'` restituisce informazioni sul sistema operativo e `'platform.machine()'` sul tipo di macchina;

**'Connection.sendall(tosend.encode() )'**: la stringa tosend viene quindi codificata utilizzando UTF-8 e inviata tramite la connessione di rete (connection è un socket);

`elif(data.decode.decode(' utf-8 ') == '2' )`: altrimenti si decodificheranno i dati con la stringa 2, identica cosa di prima se i dati sono uguali alla stringa 2 si eseguirà il codice di blocco;

`data = connection.recv(1024)`: con questo comando si possono stabilire il numero di byte ricevuti dal server (in questo caso un kylobite, anche se le condizioni dipendono dalla velocità di connessione);

`try`: nel caso tutto andasse nel modo prestabilito senza nessun tipo di problematiche il codice verrà eseguito;

`filelist = os.listdir(data.decode('utf-8' ) )`: decodifica i dati binari data tramite il percorso, e troverà l'elenco di nomi file in quella directory;

`Tosend = ""`: inizializza una variabile stringa;

`For x in filelist`: ciclo che ripete ciascun nome file 'x' nell'elenco della directory;

`Tosend+ = "," + x`: all'interno del ciclo aggiunge alla stringa ciascun nome file separato da ",";

`except`: viterà che il programma vada in crash, e manderà un messaggio informativo, un segnale in output questo permetterà di gestire meglio le eccezioni o imprevisti (in questo caso continua);

`Tosend = "Wrong Path"`: stringa che ci indica percorso sbagliato;

`Connection.sendall(tosend.encode() )`: tutti dati che verranno scambiati dalla stringa decodificata;

`elif(data.decode('utf-8') == '0' )`: non si ricevono più dati dal client;

`Connection.close()`: connessione chiusa;

`Connection, address = s.accept()`: socket, ip `s.accept=` connessione accettata().