COMPUTER NETWORKS

IT-303

PROJECT REPORT

SUBMITTED BY :

DHRUV CHANDEL
2K18/IT/046

KARTIK BANSAL
2K18/IT/064

https://github.com/dhruvchandel/Reverse_shell

Project Report

1. Introduction

- In computing, a shell is a computer program which exposes an operating system's services to a human user or other program. In general, operating system shells use either a command-line interface or graphical user interface, depending on a computer's role and particular operation.
- A reverse shell is a type of shell in which the target machine (Client) communicates back to the attacking machine (Server). The attacking machine has a listener port on which it receives the connection, which by using, code or command execution is achieved.
- A firewall usually blocks incoming connections on open ports, but does not block outgoing traffic and since in reverse shell the connection is initiated by client itself; therefore reverse shell can easily bypass firewall and server/host can easily execute commands on client/target machine.

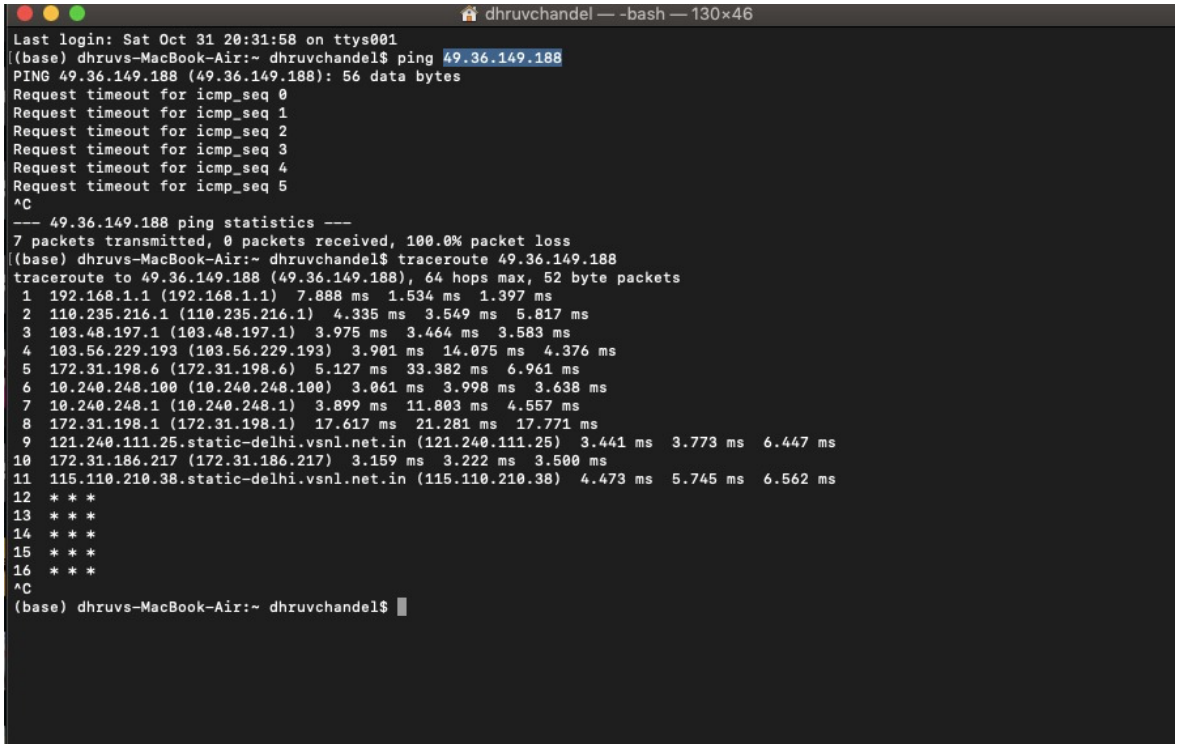
2. Idea

- Reverse shell is a powerful tool **as they are** the only way to perform remote maintenance on hosts behind a NAT, so they have legitimate administrative **uses**. However, they can also be **used** by cybercriminals to execute operating system commands on hosts protected from incoming connections by a firewall or other network security systems.
- So our Aim is to implement a multi-client reverse shell program using socket programming in python which has various features which are described in further sections.

3. DESIGN

❖ Connection Initiation Request

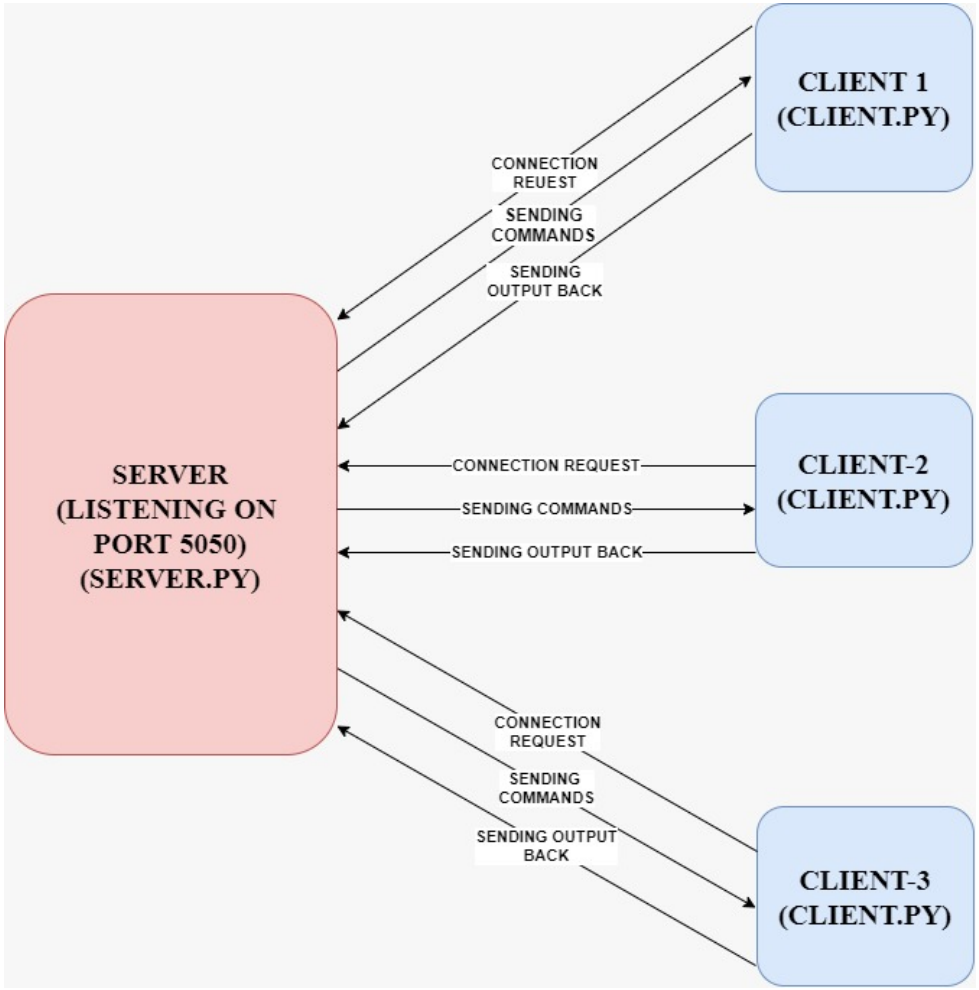
- ➢ LAN Connection: When 'server' and 'client' are present on same LAN then, connection can easily be established using private IP of the server only.
- ➢ Connection Outside LAN: when 'server' and 'client' are present on different networks and want to connect using public Internet (through ISP), then public IP of server needs to be accessible by the client.
  - ▪ But usually due to firewalls deployed by most ISP's on their internal Networks public IP's are not accessible, and also most ISP's don't allow port-forwarding on their routers.
  - ▪ So to tackle above stated problem, we can use a third party VPN server (which allows port forwarding) to act as an intermediate to pass information from server to client and vice-versa and successfully (using VPN tunnelling). *Eg: OpenVPN*



❖ Sending Commands From Server to Client

- ➢ Interactive shell -> We created an interactive shell (KD) that provides various commands and functionalities for multi-client support which are mentioned in further section.

https://github.com/dhruvchandel/Reverse_shell



https://github.com/dhruvchandel/Reverse_shell

❖ Receiving infrastructure (client to server)
  - We used a send and receive infrastructure while transferring of files and commands in which we send an acknowledgement for each message received at either end and thus ensured that entire message sent from one end is received at other end successfully.

4. FEATURES

❖ **Interactive Shell Commands**
  - Provides interactive shell for easy management and handling of multiple clients from single server.
  - Commands :
    - List
    - Help
    - Select <client_id>
    - exit

❖ **Webcam Feed**
  - It returns a webcam feed of the client machine (Pre-set at dur=6s and fps=20)
  - Command = webcam

❖ **Screenshot**
  - Displays .png format screenshot of the client machine
  - Command = ss

❖ **File Transfer**
  - File Transfer to and fro between client and server.
  - Command (to retrieve file from client machine at the given file path) = getfile filepath
  - Command (to send file from server to client at the given file path) = sendfile filepath

❖ **Live Screen Feed**

- It returns a screen feed of the client machine (Pre-set at dur=6s and fps=20)
- Command = rec

❖ **System Information**

- Returns Host name, OS version, chip information etc.
- Command = sysinfo

❖ **Shell Commands**

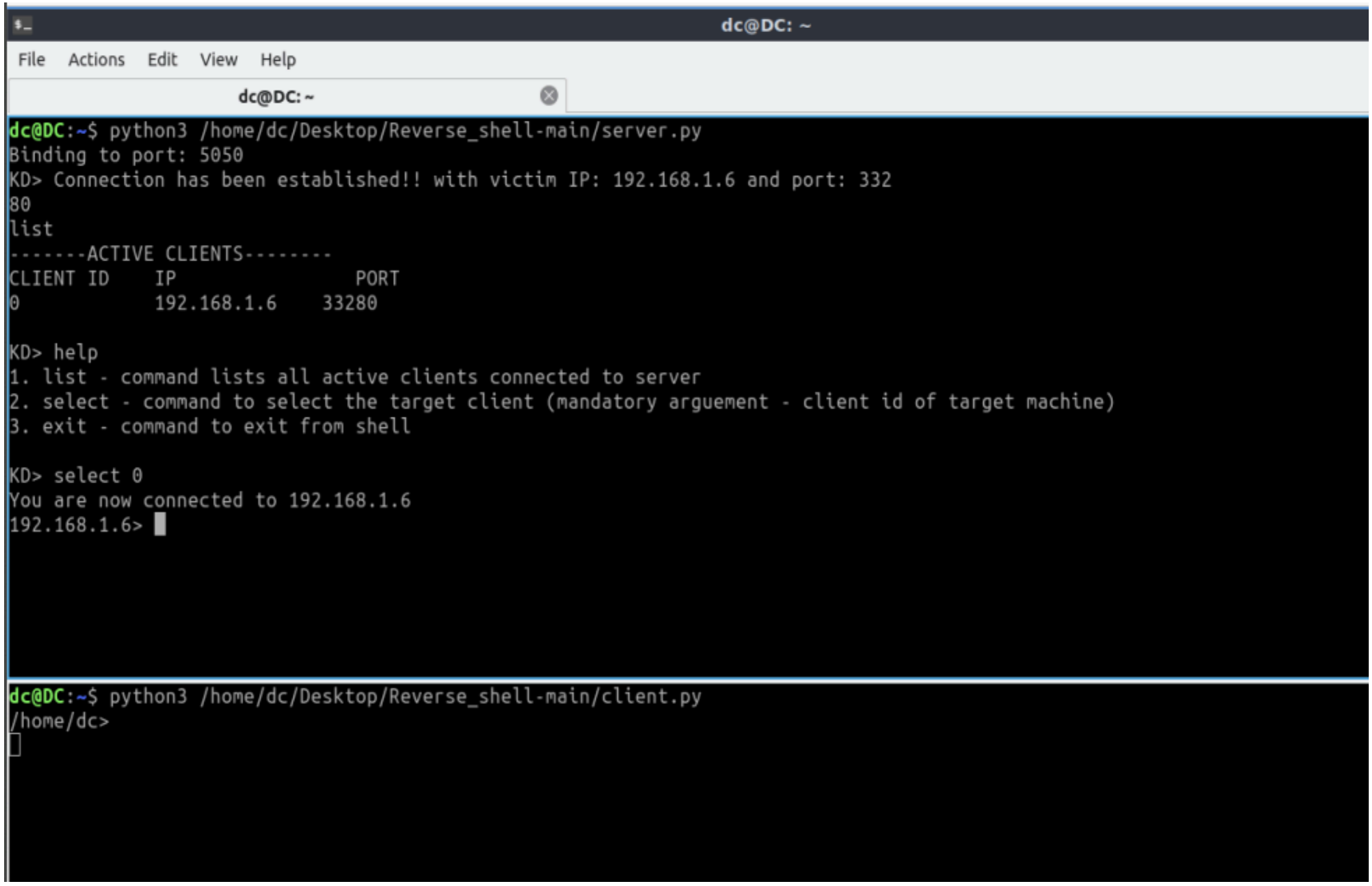
- Execute terminal/cmd commands on the client machine.

.

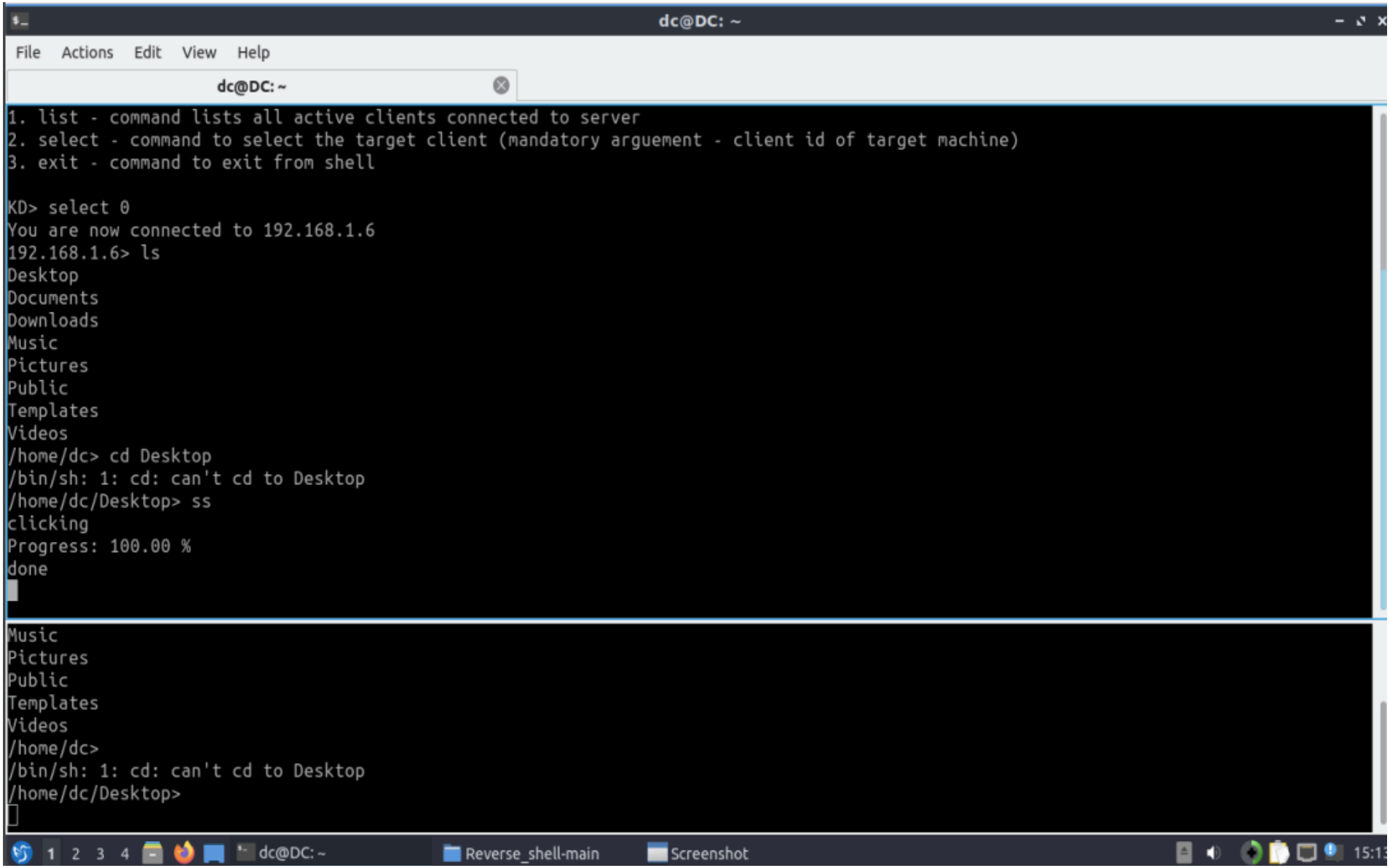
Table I: Implementation Environment, an example

PROGRAMMING LANGUAGE(S)	Python 3
OPERATING SYSTEM	Cross-Platform (Windows/Mac/Linux)
LIBRARY PACKAGES OR APIS USED (IF ANY)	Subprocess, threading, os, pyautogui, opencv, pyscreenshot, socket, sys, queue, platform
INTERFACE DESIGN (GUI / WEB / OTHER)	Interactive Shell
SERVERS USED (IF ANY)	OpenVpn (Non LAN connections)

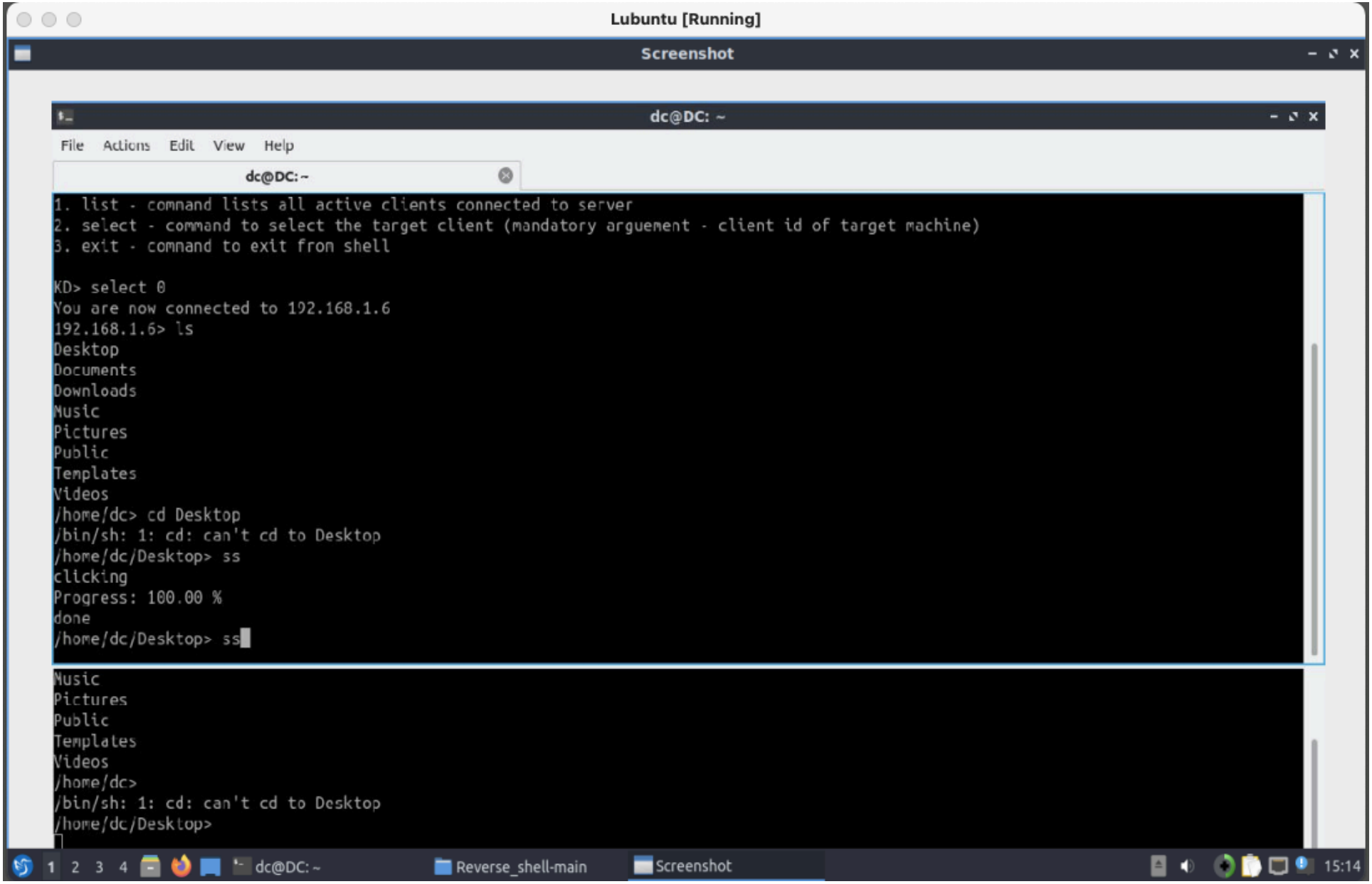
5. RESULTS



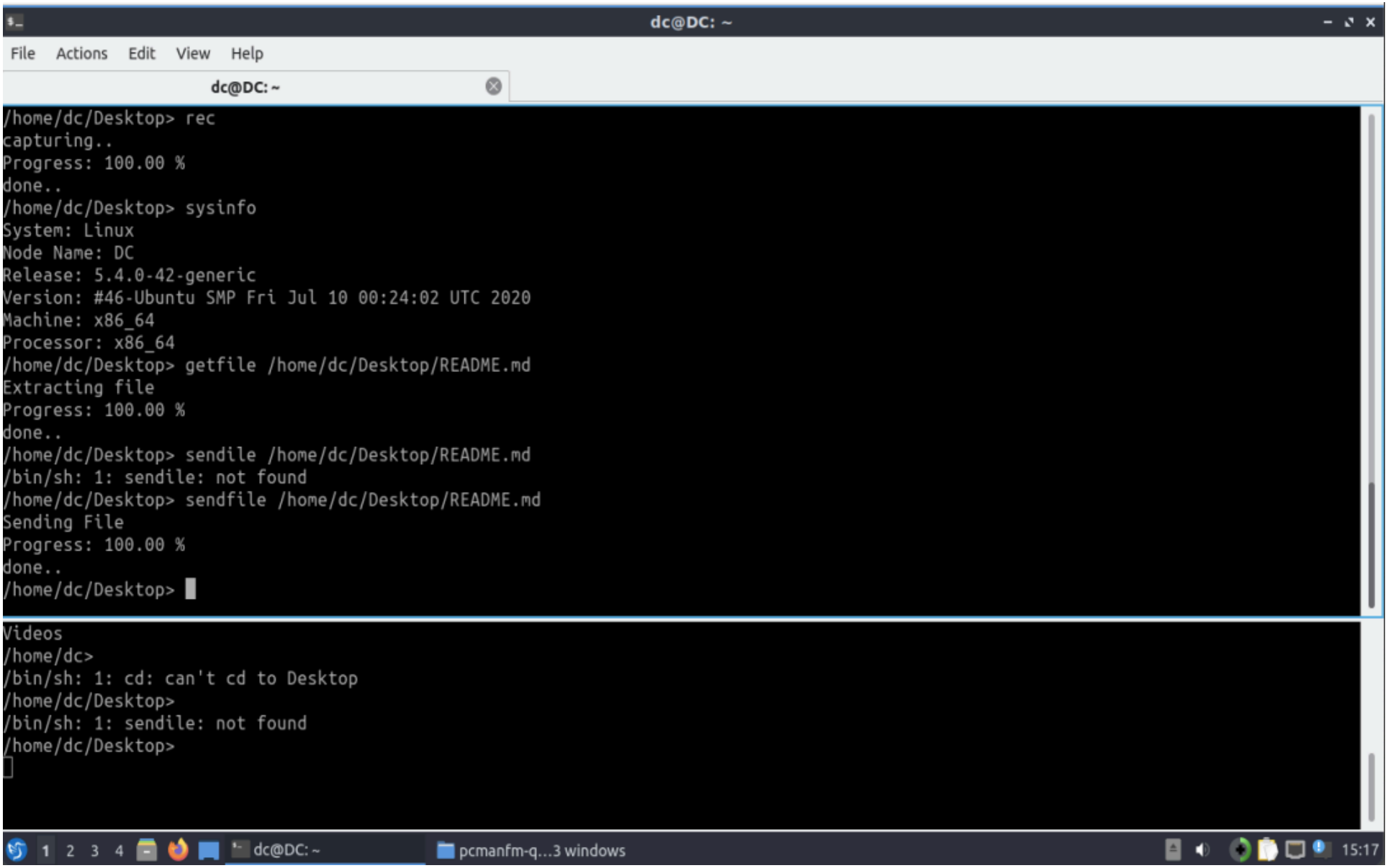
https://github.com/dhruvchandel/Reverse_shell



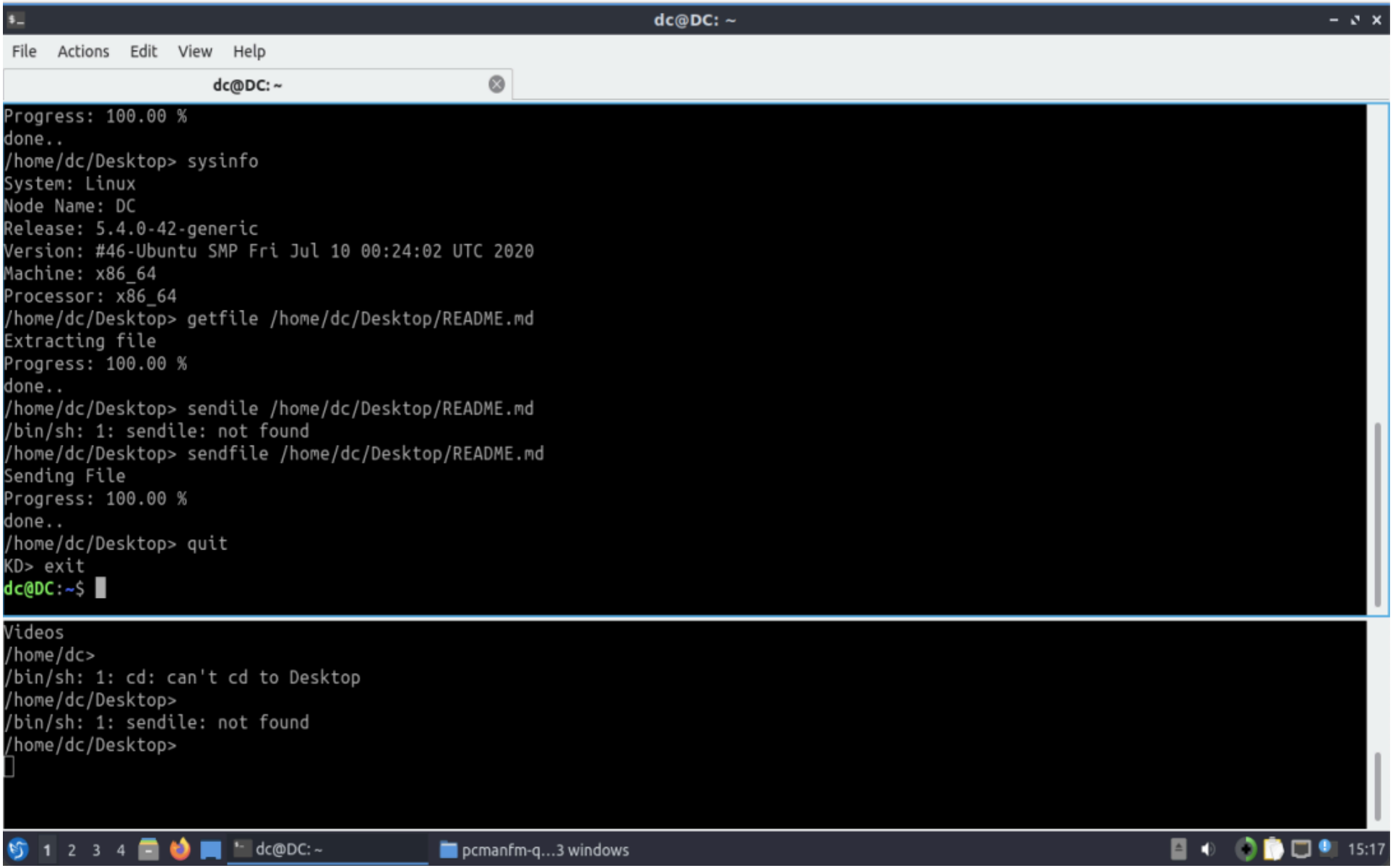
https://github.com/dhruvchandel/Reverse_shell



https://github.com/dhruvchandel/Reverse_shell



https://github.com/dhruvchandel/Reverse_shell



6. FUTURE WORK

- Provide a more interactive and easy to use GUI for easy access to the functionalities.

https://github.com/dhruvchandel/Reverse_shell

- Provide high speed servers to provide live webcam/screen feed instead of recorded ones.
- Provide keyboard/mouse input from host to client machine.

7. REFERENCES

- https://en.wikipedia.org/wiki/Reverse_connection#:~:text=In%20a%20normal%20forward%20connection,firewall%20and%20router%20security%20restrictions.