**PyRemoteDesk**

**Table of Contents:**

[**1. Introduction 2**](#_heading=h.gjdgxs)

[**2. Tool Details 2**](#_heading=h.30j0zll)

[**3. Installation of the Tool**](#_heading=h.1fob9te) **3**

[**4. Execution**](#_heading=h.3znysh7) **3**

[**5. Output**](#_heading=h.2et92p0)**4**

**6**[**. Conclusion**](#_heading=h.1t3h5sf) **4**

**7**[**. References**](#_heading=h.4d34og8) **4**

# Introduction

This tool (script) offers comprehensive solutions for remote desktop management and control, enabling users to interact with remote systems via live video streaming, input device manipulation, and remote command execution.

# Tool Details

**Dependencies**:

* cv2: OpenCV library for image processing.
* numpy: Numerical computing library for handling arrays.
* socket: Provides low-level networking interfaces.
* pickle: Serialization and deserialization of Python objects.
* struct: Provides functions to parse and format C-like data structures.
* threading: Multi-threading support for concurrent execution.
* lzma: Compression library for compressing data.
* os: Miscellaneous operating system interfaces.
* json: JSON encoding and decoding.
* pyautogui: Provides cross-platform control of mouse and keyboard.
* pynput: Monitor and control input devices.
* win32gui: Windows-specific API for GUI-related operations.
* PIL: Python Imaging Library for image processing.
* base64: Encoding and decoding binary data as ASCII.
* subprocess: Subprocess management for executing shell commands.
* mss: Cross-platform library to capture screenshots.
* screeninfo: Retrieves monitor/display information.

**Use Cases:**

* **Remote Desktop Monitoring and Control**: Allows users to view and interact with remote desktop environments through real-time video streaming.
* **Input Redirection**: Enables remote control of mouse and keyboard inputs, facilitating seamless interaction with remote systems.
* **Shell Command Execution**: Executes commands on the remote system, providing administrators with remote access and management capabilities.

# Installation of the Tool

**To install and configure the tools, follow these steps:**

**Python and Pip Installation:** Ensure Python and pip (Python package installer) are installed on your system.

**Install Dependencies:** Use pip to install required Python packages. Open a terminal or command prompt and execute:

pip install opencv-python numpy pyautogui pynput Pillow mss screeninfo

**Setup:** Depending on your use case (server or client), configure the scripts accordingly by setting IP addresses, ports, and permissions.

# Execution

**Both scripts serve distinct roles in remote desktop management:**

* **Code 1** (Server):
  + Acts as a server listening for incoming connections.
  + Captures remote desktop frames, manages mouse events, and executes shell commands on the remote system.
* **Code 2** (Client):
  + Connects to a remote server.
  + Streams the local desktop to the server, sends mouse and keyboard events, and executes shell commands on the remote system.

**Steps for using the scripts**

1. Edit the server.py.
2. Enter the Desired Port Number
3. Run it
4. Run the ngrok on your pc
5. Tunnel your ip(localhost) and port using the ngrok
6. After that make note of the ip and port number you get from the ngrok
7. After that enter this details in the client.py
8. Convert the client.py into exe by using the pyinstaller module.
9. After entering the ip and port send it to the victim(client)
10. Wait for the victim / client to run the exe
11. After that you will get the access of the victims screen and shell

# Output

Code 1 (Server):

* 1. Acts as a server listening for incoming connections.
  2. Captures remote desktop frames, manages mouse events, and executes shell commands on the remote system.

Code 2 (Client):

* 1. Connects to a remote server.
  2. Streams the local desktop to the server, sends mouse and keyboard events, and executes shell commands on the remote system.

# Conclusion

These Python scripts are essential tools for administrators and developers involved in remote system management. Offering capabilities such as live streaming, input control, and remote command execution, they enhance productivity and facilitate efficient management of distributed computing environments.

# References

Official documentation for Python packages used.