ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

SƯU LIỆU ĐỒ ÁN MÔN HỌC

MẠNG MÁY TÍNH

ĐỀ TÀI: Điều khiển máy tính thông qua email

Giảng viên lý thuyết: Thầy Đỗ Hoàng Cường
Lớp: 20TN
Thành viên thực hiện:

- 20120131 – Nguyễn Văn Lộc
- 20120536 – Võ Trọng Nghĩa
- 20120572 – Nguyễn Kiều Minh Tâm

THÀNH PHỐ HỒ CHÍ MINH, THÁNG 5-6 NĂM 2022

Mục lục

Danh sách hình vẽ

Danh sách bảng

1 Giới thiệu

Ứng dụng cho phép người dùng điều khiển các máy tính (tối đa 4 máy) trong cùng mạng LAN với server. Các chức năng được xây dựng trong đồ án bao gồm:

- Đăng ký email ứng với IP.
- Liệt kê danh sách các IP đang kết nối.
- Ngắt kết nối với một IP.
- Liệt kê danh sách các tiến trình (process), ứng dụng (application), tắt (kill) một tiến trình.
- Video màn hình hiện tại.
- Video webcam hiện tại.
- Bắt phím nhấn (keylog).
- Tắt máy, đăng xuất, khởi động lại máy tính.
- Danh sách các registry, cập nhật giá trị 1 entry trong registry.
- Liệt kê các thư mục/file trong 1 đường dẫn, copy file.

2 Cú pháp email

Các email điều khiển được gửi về địa chỉ **notabotbytheway@outlook.com**.
Các lệnh phải được viết ở **tiêu đề** của email.
Giá trị key là **ABDCFE123465**.
Cú pháp của các lệnh điều khiển được liệt kê bên dưới:

- **AUTH** <key> <IP>: "đăng ký" địa chỉ email để điều khiển <IP>. Tại một thời điểm, mỗi địa chỉ email đăng ký **duy nhất** với một địa chỉ IP và ngược lại.
- **LIST** <key>: liệt kê danh sách các IP đang kết nối.
- **DISC**: ngắt kết nối địa chỉ email này với IP hiện tại.
- **LIST_PROC**: liệt kê danh sách các tiến trình (process).
- **LIST_APP**: liệt kê danh sách các ứng dụng (application).
- **KILL** <ID/PID>: tắt (kill) tiến trình có mã là <ID/PID>.
- **SCREENSHOT** <time>: chụp màn hình liên tục trong thời gian <time> giây, mặc định là 0.5 giây.
- **WEB/REC** <time>: quay lại webcam của máy trong thời gian <time> giây, mặc định là 5 giây.

- **KEYLOG** <time>: bắt các phím nhấn trong thời gian <time> giây, mặc định là 10 giây.
- **SHUTDOWN**: tắt máy tính.
- **LOGOUT**: đăng xuất.
- **RESTART**: khởi động lại máy tính.
- **MAC**: lấy địa chỉ MAC.
- **REGISTRY LIST** <path>: danh sách các registry trong path.
- **REGISTRY UPDATE** <registry path> <value> <data type>: cập nhật giá trị của <registry path> thành <value> với kiểu <data type>. <value> **không** được có khoảng trắng. <data type> được lấy từ link.
- **DIR LIST** <path>: liệt kê các thư mục/file trong đường dẫn <path>.
- **DIR COPY** <source file> <destination path>: copy file có tên <source file> đến đường dẫn <destination path>.

3 Sưu liệu cho các hàm

3.1 Phân tích câu lệnh

Hàm phân tích câu lệnh được định nghĩa là:

```
def command_parser(message, sender_address):
```

Chức năng: phân tích câu lệnh được gửi đến, gọi hàm tương ứng với yêu cầu của câu lệnh.
Tham số:

- `message`: câu lệnh được gửi đến
- `sender_address`: địa chỉ email của người gửi.

Không có giá trị trả về.

Hàm `get_corresponding_ip`

```
def get_corresponding_ip(sender_address):
```

Chức năng: lấy ra địa chỉ IP tương ứng với `sender_address`.
Tham số:

- `sender_address`: địa chỉ email của người gửi.

Trả về địa chỉ IP tương ứng, trả về None nếu không tồn tại.

3.2 Module handle

Các hàm được định nghĩa trong module handle.py được sử dụng để xử lý các yêu cầu từ người dùng.

```
def authorize(email, ip):
```

Chức năng: authorize địa chỉ email với IP tương ứng.
Thám số:

- `email`: địa chỉ email.
- `ip`: địa chỉ IP.

Trả về True nếu thành công, False nếu không thành công.

```
def list_ip():
```

Chức năng: trả về danh sách các địa chỉ IP đang được kết nối.

```
def disconnect(email):
```

Chức năng: disconnect địa chỉ email hiện tại với IP tương ứng. Tham số:

- `email`: địa chỉ IP tương ứng.

Không có giá trị trả về.

```
def find_corresponding_email(ip_address):
```

Chức năng: tìm địa chỉ email tương ứng với IP được truyền vào. Tham số:

- `ip_address`: địa chỉ IP.

Trả về: địa chỉ email tương ứng đang điều khiển địa chỉ IP này.

```
def delete_ip_from_list(ip_address):
```

Chức năng: xóa địa chỉ IP ra khỏi danh sách IP kết nối. Tham số:

- `ip_address`: địa chỉ IP.

Không có giá trị trả về.

```
def remove_this_connection(conn, ip_address):
```

Chức năng: loại bỏ kết nối tương ứng với IP. Tham số:

- `conn`: socket tương ứng.
- `ip_address`: địa chỉ IP.

Không có giá trị trả về.

```
def __init__():
```

Chức năng: khởi tạo những giá trị liên quan.

3.3 Process/Application

Server sẽ gửi lệnh **APP_PRO** đến client, sau đó sẽ gửi số tương ứng với lệnh.

- Gửi số 1 nếu ta cần đưa ra danh sách process/application. Sau đó gửi message là **APPLICATION** hoặc **PROCESS** tương ứng với yêu cầu của lệnh đến client. Client nhận thông điệp rồi gửi lại danh sách application/process dưới dạng một data frame.
- Gửi số 0 nếu ta cần kill một process/application. Sau đó ta sẽ gửi ID/PID của đối tượng cần kill. Client nhận thông điệp rồi gửi thông tin về quá trình kill lại cho server.

Các hàm cho quá trình này như sau:
Ở server:

```
def list_process(ip_address): # module handle
def _list(conn:socket.socket, s): # module app_process_server
def send_kill(conn:socket.socket, process_id): # module app_process_server
```

Thư viện sử dụng: pickle, struct, pandas.
Ở client:

```
def app_process(client): # module app_process_client
```

Thư viện sử dụng: pickle, psutil, struct, os, subprocess.
Server có các hàm phụ trách việc nhận dữ liệu:

```
def recvall(sock, size):
def receive(client):
```

Client còn có các hàm phục vụ việc gửi dữ liệu cho server và xử lý các yêu cầu về application/process trên máy.

```
def send_data(client, data):
def list_apps():
def list_processes():
def kill(pid):
```

3.4 Chụp màn hình

Server sẽ gửi lệnh **LIVESCREEN** đến client, sau đó sẽ gửi lệnh tương ứng với thời gian cần chụp màn hình, thời gian mặc định là 0.5 giây. Thời gian chụp không nên quá nhiều, vì giới hạn dung lượng tệp đính kèm trong email. Sau đó, server sẽ nhận ảnh gửi từ client, tạo thành một video.
Các hàm cho quá trình này như sau:
Ở server:

```
def capture_screen(ip_address, time=0.5): # module handle
```

Chức năng: gửi yêu cầu chụp màn hình đến cho client, nhận hình ảnh từ client gửi lại, tạo video.
Ở client:

```
def capture_screen(client):
```

Chức năng: nhận yêu cầu chụp màn hình từ server, chụp màn hình liên tục rồi gửi từng ảnh lại cho server.
Thư viện sử dụng: ImageGrab, io, time.
Ngoài ra, ở server còn có hàm sau dùng để tạo video từ những ảnh chụp màn hình nhận được từ client, sử dụng thư viện os, cv2.

```
def create_video(image_folder: str):
```

3.5 Webcam

Server sẽ gửi lệnh **WEBCAM** đến client, sau đó sẽ gửi lệnh tương ứng với thời gian cần chụp màn hình, thời gian mặc định là 5 giây. Client nhận thông điệp từ server, quay màn hình webcam trong khoảng thời gian đó, rồi gửi lại video cho server.
Các hàm cho quá trình này như sau:
Ở server:

```
def capture_webcam(ip_address, time=5):
```

Chức năng: gửi yêu cầu ghi lại webcam đến cho client, nhận video từ client gửi về.
Ở client:

```
def run(conn: socket.socket):
```

Chức năng: nhận yêu cầu từ server, quay màn hình webcam, gửi lại file video cho server.
Thư viện sử dụng: tempfile, cv2, time.
Ngoài ra, ở client còn có hàm sau dùng để gửi file về cho server:

```
def send_file(conn: socket.socket):
```

3.6 Keylogger

Server sẽ gửi lệnh **KEYLOG** cho client. Do thiết kế của các hàm có sẵn cho việc keylog ở client, server sẽ gửi lệnh **HOOK**, sau đó nghỉ một khoảng thời gian bằng đúng thời gian yêu cầu của lệnh, rồi lại gửi lệnh **HOOK** đến client. Sau đó, một lệnh **PRINT** được gửi đến client để client gửi dữ liệu về các phím nhấn đã bắt được trong thời gian đó lại cho server. Server nhận kết quả từ client.
Các hàm cho quá trình này như sau:
Ở server:

```
def keylog(ip_address, time=10):
```

Chức năng: gửi yêu cầu keylog đến client, nhận kết quả từ client gửi về.
Sử dụng hàm `sleep` của thư viện time.
Ở client:

```
def keylog(client):
```

Chức năng: nhận lệnh từ server, bắt các phím nhấn rồi gửi kết quả lại cho server.
Thư viện sử dụng: threading, keyboard, pynput.keyboard.
Ngoài ra, ở client còn có các hàm để bổ trợ cho hàm `keylog` trong việc bắt phím nhấn cũng như gửi kết quả lại cho server.

```
def keylogger(key):
def _print(client):
def listen():
```

Chức năng:
Tham số:

-

Trả về: