

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN – ĐHQG TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN SOCKET MẠNG MÁY TÍNH
ĐIỀU KHIỂN THIẾT BỊ TỪ XA
THÔNG QUA EMAIL

Giáo viên hướng dẫn: Đỗ Hoàng Cường - Huỳnh Thuy Bảo Trân

Họ và tên sinh viên:

Mã số sinh viên:

Lớp:

Hoàng Ngọc

23120146

23CTT2

Tô Thành Long

23120143

23CTT2

Nguyễn Thanh Phong

23120154

23CTT2

TP. Hồ Chí Minh, 2024

MỤC LỤC

| | | |
|-----------|--|-----------|
| 1. | Tổng quan..... | 1 |
| 1.1. | Thông tin nhóm | 1 |
| 1.2. | Giới thiệu đề tài | 1 |
| 1.2.1. | Tên đề tài | 1 |
| 1.2.2. | Sơ lược về các chức năng của đồ án..... | 1 |
| 1.3. | Công nghệ hỗ trợ | 2 |
| 1.4. | Bảng định nghĩa..... | 2 |
| 2. | Kế hoạch đồ án..... | 5 |
| 3. | Cách hoạt động của toàn bộ hệ thống..... | 9 |
| 4. | Kiến trúc hệ thống | 10 |
| 4.1. | Phía Server (Windows) | 10 |
| 4.1.1. | Start/Stop/List các Apps | 10 |
| 4.1.2. | Start/Stop/List các Services | 11 |
| 4.1.3. | Shutdown/Restart/Sleep | 13 |
| 4.1.4. | Tương tác tệp, thư mục..... | 14 |
| 4.1.5. | Chụp màn hình | 20 |
| 4.1.6. | Webcam..... | 22 |
| 4.1.7. | Keylogger | 25 |
| 4.2. | Phía client (Linux)..... | 27 |
| 4.2.1. | Gmail API..... | 28 |
| 4.2.2. | Giao tiếp với Server..... | 34 |
| 4.2.3. | Mô hình quản lý đa người dùng | 35 |
| 4.3. | Giao tiếp Socket | 37 |
| 4.3.1. | Cài thư viện | 37 |
| 4.3.2. | Tạo môi trường và cấu hình | 38 |
| 5. | Bảng phân công..... | 41 |
| 5.1. | Bảng phân công chức năng và công việc trong hệ thống | 41 |
| 5.2. | Bảng phân công viết báo cáo..... | 42 |
| 5.3. | Phân công hoạt động nhóm do nhóm trưởng tổ chức..... | 42 |

| | | |
|-----------|---|-----------|
| 6. | Trình bày các mẫu, bảng tra cứu..... | 44 |
| 6.1. | Mẫu Request của Client | 44 |
| 6.1.1. | Chức năng List Apps/ Services | 44 |
| 6.1.2. | Chức năng Start Apps/Services | 44 |
| 6.1.3. | Chức năng Stop Apps/Services | 44 |
| 6.1.4. | Chức năng Shutdown | 45 |
| 6.1.5. | Chức năng Sleep..... | 45 |
| 6.1.6. | Chức năng Restart | 45 |
| 6.1.7. | Chức năng Copy File..... | 45 |
| 6.1.8. | Chức năng Move File/Folder..... | 46 |
| 6.1.9. | Chức năng Delete File/Folder | 46 |
| 6.1.10. | Chức năng Create File/Folder | 46 |
| 6.1.11. | Chức năng List Files/Folders | 47 |
| 6.1.12. | Chức năng Save Files | 47 |
| 6.1.13. | Chức năng Get Files/Folder | 47 |
| 6.1.14. | Chức năng Screenshot | 47 |
| 6.1.15. | Chụp màn hình sau mỗi X giây. Chức năng có thể bật/tắt. | 47 |
| 6.1.16. | Chức năng Keylogger..... | 48 |
| 6.1.17. | Chức năng webcam | 48 |
| 6.2. | Mẫu Response của Server | 49 |
| 6.2.1. | Trả về ngay lập tức kết quả sau và có file đi kèm | 49 |
| 6.2.2. | Trả về ngay lập tức kết quả và không có file đi kèm | 49 |
| 6.2.3. | Trả về thông báo khi thực hiện bật/tắt chức năng có chu kì..... | 49 |
| 7. | Định hướng phát triển trong tương lai..... | 50 |
| 7.1. | Những vấn đề chưa giải quyết..... | 50 |
| 7.1.1. | Bảng mã Error_Code..... | 50 |
| 7.1.2. | Xử lý độ trễ giữa các gói tin | 50 |
| 7.1.3. | Xác thực người dùng | 51 |
| 7.2. | Giải pháp đề xuất và chức năng tương lai | 51 |
| 7.2.1. | Giải pháp cho các vấn đề..... | 51 |
| 7.2.2. | Chức năng trong tương lai..... | 51 |

| | | |
|-----------|---|-----------|
| 8. | Thư viện sử dụng | 53 |
| 8.1. | Thư viện OpenCV (Open Source Computer Vision Library) | 53 |
| 8.2. | Thư viện cURL (Client URL Library) | 53 |
| 8.3. | Thư viện json (JavaScript Object Notation)..... | 53 |
| 9. | Tài liệu tham khảo..... | 55 |

PHỤ LỤC HÌNH ẢNH

| | |
|---|----|
| Hình 1: Liệt kê danh sách các ứng dụng, tiến trình | 10 |
| Hình 2: Liệt kê danh sách các dịch vụ | 12 |
| Hình 3: Sau khi tắt dịch vụ có tên SearchIndexer..... | 13 |
| Hình 4: Các thư mục được tạo trong đường dẫn..... | 15 |
| Hình 5: Các thư mục được di chuyển đến đường dẫn | 15 |
| Hình 6: Các tệp và thư mục đã được liệt kê..... | 18 |
| Hình 7: Tệp ảnh được yêu cầu lưu vào Server..... | 19 |
| Hình 8: Tệp ảnh sau khi được lưu..... | 19 |
| Hình 9: Tệp nén sau khi được tải về ở máy người dùng..... | 20 |
| Hình 10: Email đính kèm ảnh chụp màn hình | 21 |
| Hình 11: Các ảnh chụp màn hình theo chu kỳ | 22 |
| Hình 12: Ảnh chụp bằng webcam trả về file .zip | 23 |
| Hình 13: Ảnh chụp bằng webcam theo chu kỳ và lưu vào ổ D | 24 |
| Hình 14: Ảnh quay video bằng webcam và lưu vào ổ D | 25 |
| Hình 15: Gõ phím trên một ứng dụng bất kỳ trên thiết bị server..... | 27 |
| Hình 16: Trả về các file lưu phím ấn sau một khoảng chu kỳ cho client | 27 |
| Hình 17: Cài đặt thư viện winsock32 | 38 |

1. Tổng quan

1.1. Thông tin nhóm

| MSSV | Họ tên | Email | Vai trò |
|----------|--------------------|--|-------------|
| 23120146 | Hoàng Ngọc | 23120146@student.hcmus.edu.vn | Nhóm Trưởng |
| 23120154 | Nguyễn Thanh Phong | 23120154@student.hcmus.edu.vn | Thành viên |
| 23120143 | Tô Thành Long | 23120143@student.hcmus.edu.vn | Thành viên |

1.2. Giới thiệu đề tài

1.2.1. Tên đề tài

Điều khiển thiết bị từ xa thông qua email. Lý do chọn đề tài: **“Điều khiển thiết bị từ xa thông qua email (socket)”** được lựa chọn nhờ tính ứng dụng cao của email, cho phép xây dựng giải pháp điều khiển hiệu quả mà không tốn kém. Theo dõi các hoạt động trên máy chủ từ xa phù hợp cho việc giám sát và quản lý. Tăng tính tự động hóa bằng cách giảm sự can thiệp thủ công vào các tác vụ quản trị. Nhanh chóng phản hồi sự cố hoặc xử lý các tình huống khẩn cấp tăng tính bảo mật và dễ bảo trì.

1.2.2. Sơ lược về các chức năng của đồ án

Hệ thống điều khiển thiết bị từ xa qua mail cung cấp các tính năng tiện ích nhằm quản lý và tương tác với máy chủ hoặc thiết bị một cách hiệu quả. Dưới đây là mô tả sơ lược về từng chức năng:

- **List apps/processes:** Hiển thị danh sách tất cả các ứng dụng hoặc tiến trình đang hoạt động trên máy chủ, bao gồm tên, nhà phát triển, trạng thái hoạt động, số tiến trình con, bộ nhớ đang sử dụng và đường dẫn. Tính năng nâng cao, có thể sắp xếp danh sách theo một trường tùy ý.
- **Start app:** Mở một ứng dụng cụ thể dựa trên tên hoặc đường dẫn được chỉ định.
- **Stop app:** Ngừng chạy một ứng dụng hiện tại để tiết kiệm tài nguyên hoặc đảm bảo an ninh.
- **List services:** Hiển thị danh sách các dịch vụ đang được vận hành trên máy chủ, tương tự như List apps/processes.
- **Start service:** Kích hoạt một dịch vụ cụ thể, thường liên quan đến các tác vụ nền như mạng hoặc cơ sở dữ liệu.
- **Stop service:** Tạm dừng một dịch vụ đang chạy để giải phóng tài nguyên hoặc khắc phục lỗi.
- **Shutdown server:** Tắt máy chủ một cách an toàn và có kiểm soát.
- **Restart server:** Khởi động lại máy chủ để cập nhật cấu hình hoặc xử lý sự cố.
- **Sleep server (chức năng thêm):** Đặt máy chủ vào chế độ ngủ.

- **Copy files/folders:** Thực hiện sao chép tệp/thư mục từ một vị trí này sang vị trí khác nhằm mục đích lưu trữ hoặc đồng bộ.
- **Move files/folders (chức năng thêm):** Di chuyển tệp/thư mục từ một vị trí này sang vị trí khác nhằm mục đích lưu trữ hoặc đồng bộ.
- **Delete files/folders:** Xóa các tệp/thư mục không cần thiết để giải phóng không gian lưu trữ.
- **List files/folders (chức năng thêm):** Liệt kê các tệp và thư mục trong một vị trí cụ thể, bao gồm tên tệp/thư mục, thời gian chỉnh sửa lần cuối, kích thước, thao tác khả dụng và loại.
- **Create files/folders (chức năng thêm):** Tạo tệp/thư mục ở một vị trí tùy ý trên máy.
- **Save files (chức năng thêm):** Lưu tệp đính kèm từ Gmail về máy server.
- **Get files (chức năng thêm):** Tạo email có đính kèm các tệp yêu cầu lấy từ máy server.
- **Screenshot:** Chụp ảnh màn hình hiện tại trên máy chủ để giám sát hoặc kiểm tra hoạt động. Ngoài ra, server có thể chụp màn hình theo chu kỳ sử dụng đa luồng.
- **Keylogger:** Ghi lại toàn bộ thao tác bấm phím để theo dõi hoạt động của người dùng. Tính năng nâng cao, người dùng có thể chỉ định ứng dụng nào được ghi phím hoặc phím nào sẽ bị khoá.
- **Webcam:** Bật camera để thu hình ảnh hoặc video từ xa. Ngoài ra, server có thể chụp hình hoặc video theo chu kỳ sử dụng đa luồng.

1.3. Công nghệ hỗ trợ

| Tên công cụ | Chức năng |
|---------------------------------|--|
| Google Document | Tổ chức báo cáo, lên ý tưởng, tóm tắt nội dung cần thực hiện của từng giai đoạn đồ án. |
| Google Meet | Họp nhóm, thảo luận, đưa ra các phương án xử lý trực tuyến |
| Github | Quản lý mã nguồn dự án |
| Trello | Lập kế hoạch thực hiện dự án, quản lý tiến độ. |
| Visual Studio Code, Visual Code | Viết mã nguồn, chạy thử nghiệm và kiểm tra ứng dụng chính của đồ án. |

1.4. Bảng định nghĩa

| Từ khoá | Định nghĩa |
|---------|---|
| Socket | Giao diện phần mềm cho phép giao tiếp giữa hai ứng dụng thông qua mạng. |

| | |
|--|--|
| JSON | (JavaScript Object Notation) Định dạng dữ liệu nhẹ, dễ đọc/ghi, dùng để trao đổi dữ liệu giữa các hệ thống. |
| Tệp zip | Một định dạng tệp nén dữ liệu phổ biến được sử dụng để lưu trữ một hoặc nhiều tệp hoặc thư mục trong một tệp duy nhất, đồng thời giảm kích thước của chúng để tiết kiệm không gian lưu trữ và tăng tốc độ truyền tải. |
| Powershell | Công cụ dòng lệnh và ngôn ngữ kịch bản của Microsoft, dùng để tự động hóa và quản lý hệ thống. |
| URL-safe base64 | Phiên bản mã hóa Base64 an toàn cho URL, thay thế ký tự "+" bằng "-", và "/" bằng "_". |
| bitmap | Định dạng đồ họa đại diện hình ảnh bằng các điểm ảnh (pixels). |
| RGB | Mô hình màu dựa trên ba thành phần Red, Green, Blue, thường dùng trong đồ họa và hiển thị màn hình. |
| API | Giao diện lập trình ứng dụng cho phép các phần mềm hoặc hệ thống giao tiếp với nhau. API định nghĩa các quy tắc và phương thức mà qua đó một ứng dụng có thể truy cập các chức năng hoặc dữ liệu của một ứng dụng khác mà không cần biết cách thức hoạt động bên trong của nó. |
| HTTP | (HyperText Transfer Protocol) Giao thức truyền tải siêu văn bản, dùng để truyền dữ liệu trên web. |
| curl | Công cụ dòng lệnh để truyền và nhận dữ liệu qua URL, hỗ trợ nhiều giao thức như HTTP, FTP. |
| MIME (Multipurpose Internet Mail Extensions) | Một tiêu chuẩn Internet mở rộng định dạng của email, cho phép gửi và nhận các nội dung không chỉ giới hạn ở văn bản thuần túy, mà còn bao gồm hình ảnh, âm thanh, video, và các tệp đính kèm. |
| Giao thức TCP | (Transmission Control Protocol) Giao thức đảm bảo truyền dữ liệu đáng tin cậy trên mạng. |
| Giao thức DHCP | (Dynamic Host Configuration Protocol) Giao thức tự động cấp địa chỉ IP và thông tin cấu hình mạng cho thiết bị. |
| Địa chỉ IP | Địa chỉ định danh duy nhất cho mỗi thiết bị trên mạng, có thể ở dạng IPv4 hoặc IPv6. |
| MAC (Media Access Control) | (Media Access Control) Địa chỉ phần cứng của card mạng, định danh thiết bị ở tầng liên kết dữ liệu. |
| Nmap | Công cụ mạng mã nguồn mở, dùng để quét mạng, kiểm tra bảo mật và phát hiện các thiết bị đang hoạt động. |

| | |
|--------------------|--|
| Hệ điều hành Linux | Hệ điều hành mã nguồn mở, phổ biến trong lập trình, quản trị hệ thống và phát triển ứng dụng. |
| Multithreading | Kỹ thuật cho phép một ứng dụng chạy đồng thời nhiều luồng (threads), tăng hiệu quả xử lý. |
| Oauth 2.0 | Một giao thức ủy quyền chuẩn mở, cho phép các ứng dụng bên thứ ba truy cập tài nguyên của người dùng trên một dịch vụ mà không cần phải chia sẻ thông tin đăng nhập của người dùng (như tên người dùng và mật khẩu). |
| Frame | Trong xử lý hình ảnh kỹ thuật số, "frame" có thể được hiểu là một hình ảnh được xử lý tại một thời điểm cụ thể. |
| Windows Hook | Một cơ chế trong hệ điều hành Windows cho phép ứng dụng theo dõi, can thiệp và xử lý các sự kiện hoặc thông điệp (messages) được truyền giữa hệ điều hành và các ứng dụng khác. |

2. Kế hoạch đồ án

| Cột mốc | Công việc dự kiến | Mức độ hoàn thành |
|----------------------------|---|--|
| Ngày 19/10/2024 | Viết Document trình bày các công nghệ, các kiến thức, các bước để thực hiện các chức năng có liên quan đến nội dung được phân chia. | Hoàn thành xong bộ phác thảo hệ thống Client-Server sẽ xây dựng với đầy đủ các chức năng. Mẫu hướng dẫn, report cho mô hình Client-Server. |
| Ngày 20/10/2024 | Xây dựng các chức năng tương tác tệp/thư. | Hoàn thành việc xây dựng các hàm cơ bản, chưa cài đặt thêm chức năng nâng cao. |
| Ngày 21/10/2024 | Xây dựng hệ thống lấy email định kỳ và hệ thống socket để giao tiếp với máy chủ server. | Hoàn thành việc xây dựng hệ thống lấy email định kỳ nhờ các token và hệ thống socket để giao tiếp với máy chủ server. |
| Ngày 22/10/2024 | Thống nhất mẫu gửi request cho việc giao tiếp. | Hoàn thành xong mẫu request gửi cho server (dạng json), hoàn thiện socket + nhận/gửi về cho admin. |
| Ngày 24/10/2024 | Cài đặt một số chức năng cơ bản. | Cơ bản cài đặt được chức năng chụp màn hình screenshot và List/Stop/Start Apps (Services)+ shut down và restart thiết bị. |
| Ngày 30/10/2024 | Kiểm thử đa luồng cho chức năng chụp màn hình định kỳ. | Hoàn thiện dần mã nguồn cho chức năng screenshot và các hàm tương tác tệp/thư mục nâng cao. |
| Ngày 31/10/2024 | Tiếp tục cải thiện các chức năng cơ bản vẫn còn thô, chưa đầy đủ. | Hoàn thành xong chức năng chụp màn hình screenshot. |

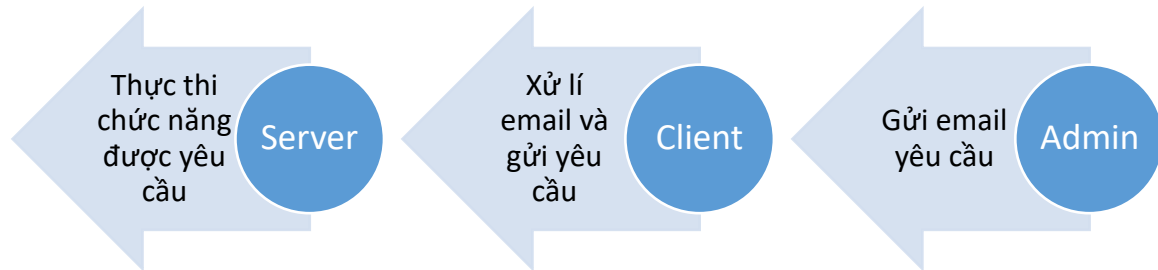
| | | |
|-----------------------------|--|---|
| Ngày 4/11/2024 | Tiếp tục cải thiện các chức năng cơ bản vẫn còn thô, chưa đầy đủ. | Hoàn thành xong chức năng List/Stop/Start Apps (Services) cơ bản + shut down và restart thiết bị. |
| Ngày 6/11/2024 | Tiếp tục cải thiện các chức năng cơ bản vẫn còn thô, chưa đầy đủ. Đồng thời thêm chức năng mới. | Cơ bản cài đặt được và hoàn thành xong chức năng bắt phím keylogger và khoá phím. |
| Ngày 11/11/2024 | Tiếp tục cải thiện các chức năng cơ bản vẫn còn thô, chưa đầy đủ. Đồng thời thêm chức năng mới. | Thêm tính năng mới: Lưu trữ và lấy tệp từ máy chủ Tối ưu hóa chức năng nhận từng phần của tệp. Bổ sung cải tiến chức năng List/Stop/Start Apps (Services). |
| Ngày 16/11/2024 | Xử lý nâng cao các trường hợp ngoại lệ phát sinh như có nhiều yêu cầu gửi đến server cùng một lúc. | Chuyển đổi sang mô hình thử nghiệm đa luồng cho client nhằm hỗ trợ tương tác đa người dùng. |
| Ôn thi giữa học kỳ I | | |
| Ngày 1/12/2024 | Nghiên cứu và hoàn thiện cơ bản bước đầu để xử lý nâng cao các trường hợp ngoại lệ phát sinh. | Hoàn thành triển khai socket hỗ trợ đa người dùng. |
| Ngày 3/12/2024 | Thêm chức năng mới để dựa vào địa chỉ vật lý MAC dò IP trong mạng rồi thiết lập kết nối socket. | Hoàn thành việc tạo chức năng phát hiện địa chỉ IP của thiết bị người dùng dựa trên địa chỉ MAC. |

| | | |
|------------------------|--|--|
| Ngày 6/12/2024 | Hoàn thiện giải pháp xử lý nâng cao các trường hợp ngoại lệ phát sinh như có nhiều yêu cầu gửi đến server cùng một lúc. Đồng thời cải thiện chức năng chưa hoàn thành. | Cơ bản hoàn thành chức năng đa luồng (multithreading) và chức năng bật/ tắt webcam. |
| Ngày 8/12/2024 | Tiếp tục cải thiện các chức năng cơ bản gần hoàn thành. | Cập nhật cải thiện hàm bật/ tắt webcam và hoàn thiện phản hồi cho client. |
| Ngày 9/12/2024 | Tiếp tục cải thiện các chức năng cơ bản gần hoàn thành. Đồng thời thêm chức năng mới. | Hoàn thành chức năng List/Stop/Start Apps (Services) cơ bản + shut down và restart thiết bị, đồng thời xây dựng chức năng gửi tệp CSV, cuối cùng sửa cấu trúc phản hồi gửi tới client. |
| Ngày 10/12/2024 | Bắt đầu kết hợp mã nguồn riêng lẻ (merge code) để dần hoàn thiện hệ thống socket. | Tích hợp thành công chức năng keylogger vào máy chủ. |
| Ngày 11/12/2024 | Tiếp tục kết hợp mã nguồn riêng lẻ (merge code) để dần hoàn thiện hệ thống socket. | Kết hợp chức năng chụp ảnh screenshot và quay video từ webcam; cải thiện hiệu suất mã nguồn của máy chủ. Kết hợp các chức năng shutdown, restart, và đưa máy chủ vào chế độ sleep. |
| Ngày 12/12/2024 | Tiếp tục kết hợp mã nguồn riêng lẻ (merge code) để dần hoàn thiện hệ thống socket. | Hoàn thiện việc tích hợp các tương tác với máy: shutdown/restart/sleep; tích hợp các tương tác với app/ service: list/ start/ stop nhiều app/ service. Cập nhật chức năng khởi động ứng dụng, tạo tệp CSV chứa danh sách các ứng dụng .exe. |

| | | |
|----------------------------|---|---|
| Ngày 20/12/2024 | Kiểm tra rà soát lại các lỗi nhỏ trong hệ thống và chỉnh sửa. | Cập nhật lại mã nguồn, report, các mẫu gửi hoàn thiện. |
| Ngày 28/12/2024 | Hoàn thành hệ thống. Biên dịch mã nguồn thành tệp .exe để cài đặt trên mọi máy. | Hoàn thành hệ thống điều khiển thiết bị từ xa thông qua email (socket). |

3. Cách hoạt động của toàn bộ hệ thống

Đối với quá trình gửi yêu cầu:



Đối với quá trình nhận phản hồi:



- Bước 1: Người dùng gửi lệnh `cmd = startas`, tham số `type = app` với danh sách tên ứng dụng cần khởi động, ví dụ `winword, notepad, chrome,...`
- Bước 2: Hệ thống tìm kiếm các ứng dụng trong danh sách và gửi tín hiệu khởi động
 - Tìm kiếm đường dẫn thực thi của từng ứng dụng thông qua cấu trúc `processPaths`.
 - Gửi lệnh khởi động ứng dụng qua PowerShell [8].
- Bước 3: Trạng thái khởi động (thành công hoặc thất bại) được ghi nhận và trả về.

4.1.1.3. Chức năng Stop

Công dụng: Dừng một hoặc nhiều ứng dụng được chỉ định.

Cách hoạt động:

- Bước 1: Người dùng gửi lệnh `cmd = stopas`, tham số `type = app` với danh sách tên ứng dụng cần dừng, ví dụ `winword, notepad, chrome,...`
- Bước 2: Gửi tín hiệu dừng tới từng ứng dụng thông qua PowerShell [8].
- Bước 3: Trạng thái dừng (thành công hoặc thất bại) được ghi nhận và trả về.

4.1.2. Start/Stop/List các Services

4.1.2.1. Chức năng List

Công dụng: Liệt kê danh sách các dịch vụ đang hoạt động, đã dừng, hoặc tắt cả.

Cách hoạt động:

- Bước 1: Người dùng gửi lệnh `cmd = listas` với tham số `type = service`.
- Bước 2: Hệ thống truy vấn danh sách dịch vụ từ hệ thống và trả về kết quả. Quá trình xử lý thông qua một số hàm chính:
 - Gọi hàm `getServices(status)` với trạng thái cụ thể (`running`, `stopped`, hoặc `all`).
 - Dữ liệu trả về được phân tích thông qua `parseServiceOutput()`.
 - Nếu có yêu cầu sắp xếp, sử dụng hàm `sortServices()`. (**Chức năng cải tiến**)
- Bước 3: Danh sách dịch vụ dưới dạng file `.csv` hoặc giao diện người dùng.

| 1 | ProcessName | Status | StartMode | Memory | Path | | | | | | | | |
|-----|--------------------------|---------|-----------|--------|---|--|--|--|--|--|--|--|--|
| 186 | esifsvc | Running | Auto | 6.37 | C:\Windows\system32\Intel\DPTF\esif_uf.exe | | | | | | | | |
| 187 | RtkBtManServ | Running | Auto | 6.74 | C:\Windows\RtkBtManServ.exe | | | | | | | | |
| 188 | DeviceAssociationService | Running | Auto | 6.77 | C:\Windows\system32\svchost.exe -k LocalSystemNetworkRestricted -p | | | | | | | | |
| 189 | LicenseManager | Running | Manual | 6.82 | C:\Windows\System32\svchost.exe -k LocalService -p | | | | | | | | |
| 190 | DispBrokerDesktopSvc | Running | Auto | 6.82 | C:\Windows\system32\svchost.exe -k LocalService -p | | | | | | | | |
| 191 | cplspcon | Running | Auto | 6.86 | C:\Windows\System32\DriverStore\FileRepository\iigd_dch.inf_amd64_51f685305808e | | | | | | | | |
| 192 | cphs | Running | Manual | 6.89 | C:\Windows\System32\DriverStore\FileRepository\iigd_dch.inf_amd64_51f685305808e | | | | | | | | |
| 193 | PolicyAgent | Running | Manual | 7.03 | C:\Windows\system32\svchost.exe -k NetworkServiceNetworkRestricted -p | | | | | | | | |
| 194 | fdPHost | Running | Manual | 7.09 | C:\Windows\system32\svchost.exe -k LocalService -p | | | | | | | | |
| 195 | W32Time | Running | Auto | 7.27 | C:\Windows\system32\svchost.exe -k LocalService | | | | | | | | |
| 196 | FoxitReaderUpdateService | Running | Auto | 7.37 | C:\Program Files (x86)\Foxit Software\Foxit Reader\FoxitReaderUpdateService.exe | | | | | | | | |
| 197 | BTAGService | Running | Manual | 7.38 | C:\Windows\system32\svchost.exe -k LocalSystemNetworkRestricted | | | | | | | | |
| 198 | AudioEndpointBuilder | Running | Auto | 7.4 | C:\Windows\System32\svchost.exe -k LocalSystemNetworkRestricted -p | | | | | | | | |
| 199 | SgrmBroker | Running | Auto | 7.47 | C:\Windows\system32\SgrmBroker.exe | | | | | | | | |
| 200 | gpsvc | Running | Auto | 7.47 | C:\Windows\system32\svchost.exe -k netsvcs -p | | | | | | | | |

Hình 2. Liệt kê danh sách các dịch vụ

4.1.2.2. Chức năng Start

Công dụng: Khởi động một hoặc nhiều dịch vụ.

Cách hoạt động:

- **Bước 1:** Người dùng gửi lệnh `cmd = startas` và tham số `type = service` với danh sách tên dịch vụ cần khởi động.
- **Bước 2:** Tìm kiếm dịch vụ trong hệ thống và gửi lệnh khởi động thông qua PowerShell [8].
- **Bước 3:** Trạng thái khởi động (thành công hoặc thất bại) được ghi nhận và trả về.

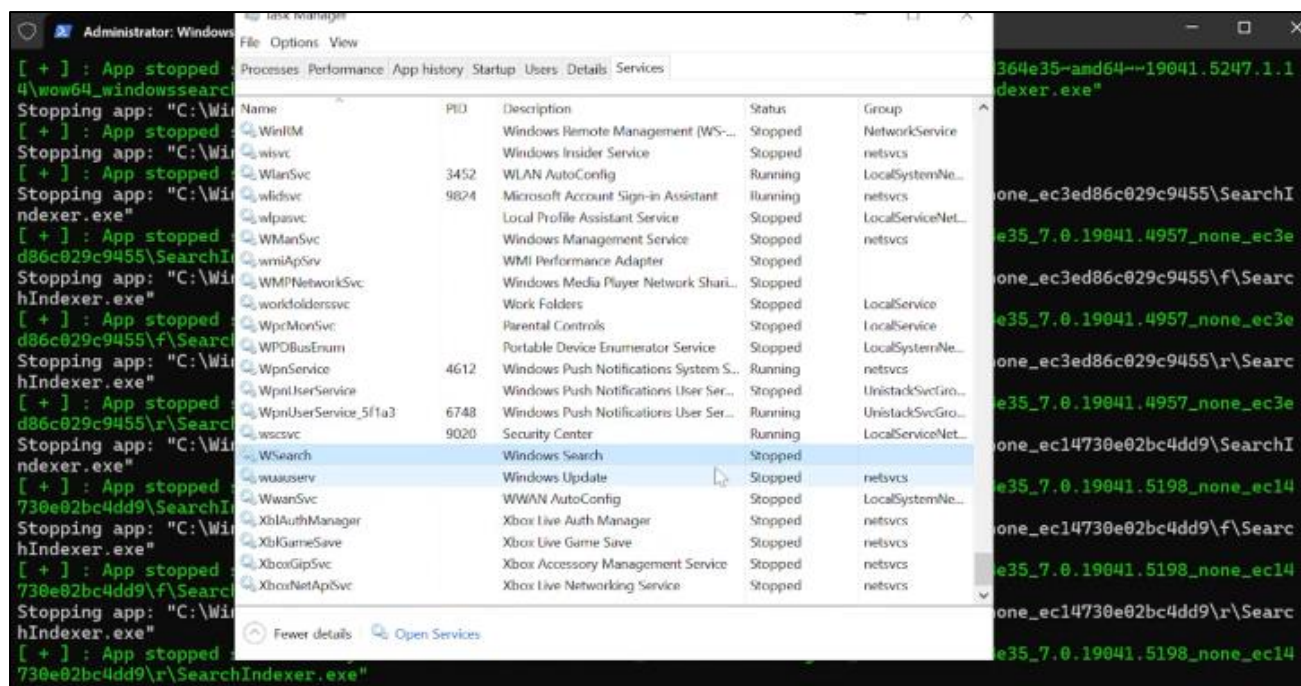
4.1.2.3. Chức năng Stop

Công dụng: Dừng một hoặc nhiều dịch vụ.

Cách hoạt động:

- **Bước 1:** Người dùng gửi lệnh `cmd = stopas` và tham số `type = service` với danh sách tên dịch vụ cần dừng.
- **Bước 2:** Gửi lệnh dừng thông qua PowerShell [8].
- **Bước 3:** Trạng thái khởi động (thành công hoặc thất bại) được ghi nhận và trả về.

Lưu ý: Chức năng này phải được chạy với quyền Administrator nhằm bảo vệ các dịch vụ hệ thống.



Hình 3. Sau khi tắt dịch vụ có tên SearchIndexer

4.1.3. Shutdown/Restart/Sleep

4.1.3.1. Chức năng Shutdown

Công dụng: Tắt thiết bị sau một khoảng thời gian chờ được chỉ định.

Cách hoạt động:

- **Bước 1:** Người dùng gửi lệnh `cmd = shutdown`, tham số `params.interval` (thời gian chờ tính bằng giây trước khi tắt thiết bị).
- **Bước 2:** Quá trình xử lý thông qua một số hàm chính:
 - Xác nhận yêu cầu từ client và gửi phản hồi ban đầu qua hàm `sendMessageOnce()`. Thông báo trạng thái `success` hoặc `failed`.
 - Thực thi lệnh PowerShell [8] để chờ một khoảng thời gian `interval` và thực hiện tắt thiết bị.
 - Hàm xử lý chính `shutdownDeviceHelper(int& sleepTime)` để thực hiện thao tác tắt thiết bị sau một khoảng thời gian chờ nhất định.
- **Bước 3:** Thiết bị tắt hoàn toàn sau khoảng thời gian `interval` được chỉ định.

4.1.3.2. Chức năng Restart

Công dụng: Khởi động lại thiết bị sau một khoảng thời gian chờ được chỉ định.

Cách hoạt động:

- **Bước 1:** Người dùng gửi lệnh `cmd = restart`, tham số `params.interval` (thời gian chờ tính bằng giây trước khi khởi động lại thiết bị).

- Bước 2: Quá trình xử lý thông qua một số hàm chính:
- Xác nhận yêu cầu từ client và gửi phản hồi ban đầu qua hàm `sendMessageOnce()`. Thông báo trạng thái `success` hoặc `failed`.
- Thực thi lệnh PowerShell [8] để chờ một khoảng thời gian `interval` và thực hiện khởi động lại thiết bị.
- Hàm xử lý chính `restartDeviceHelper(int& sleepTime)` để thực hiện thao tác khởi động lại thiết bị sau một khoảng thời gian chờ nhất định.
- Bước 3: Thiết bị khởi động lại sau khoảng thời gian `interval` được chỉ định.

4.1.3.3. Chức năng Sleep (Chức năng cải tiến)

Công dụng: Khởi động lại thiết bị sau một khoảng thời gian chờ được chỉ định.

Cách hoạt động:

- Bước 1: Người dùng gửi lệnh `cmd = sleep`, tham số `params.interval` (thời gian chờ tính bằng giây trước khi đưa thiết bị vào trạng thái ngủ).
- Bước 2: Quá trình xử lý thông qua một số hàm chính:
- Xác nhận yêu cầu từ client và gửi phản hồi ban đầu qua hàm `sendMessageOnce()`. Thông báo trạng thái `success` hoặc lỗi cụ thể, ví dụ: `-303`.
- Thực thi lệnh PowerShell [8] để chờ một khoảng thời gian `interval` và thực hiện đưa thiết bị vào trạng thái ngủ.
- Hàm xử lý chính `sleepDeviceHelper(int& sleepTime)` để đưa thiết bị vào trạng thái ngủ sau một khoảng thời gian chờ nhất định.
- Bước 3: Thiết bị chuyển vào trạng thái ngủ sau khoảng thời gian `interval` được chỉ định.

4.1.4. Tương tác tệp, thư mục

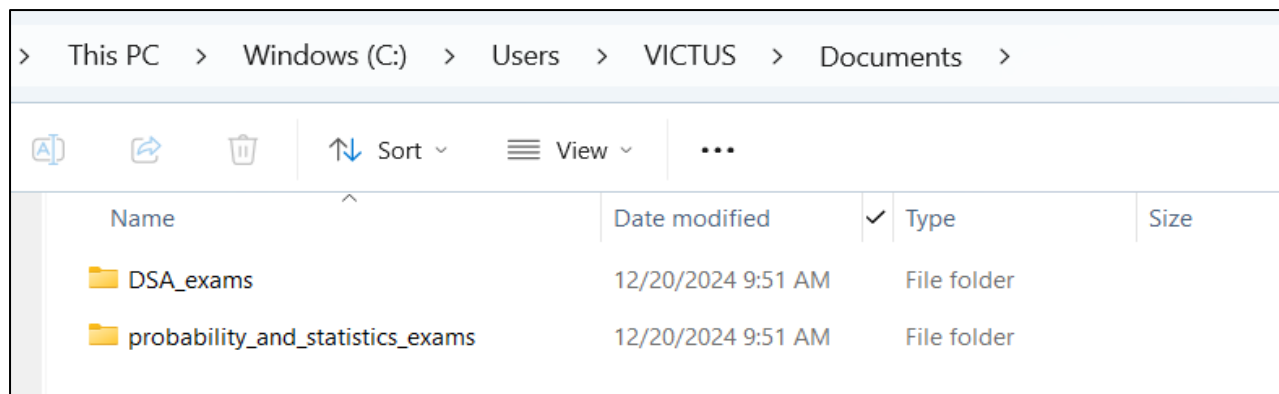
4.1.4.1. Tạo tệp/thư mục [3] [9]

Đây là chức năng mở rộng. Chức năng này nhận vào các tham số:

- `type`: tùy chọn tạo tệp (file) hay thư mục (folder)
- `name`: tên của tệp/thư mục
- `path`: đường dẫn cho tệp/thư mục sẽ được tạo

Khi đã hoàn tất, Server sẽ trả về trạng thái là thành công (`success`) hoặc lỗi (`error`).

Ví dụ: Khi người dùng yêu cầu tạo các thư mục có tên `probability_and_statistics_exams`, `DSA_exams` với đường dẫn `C:\Users\VICTUS\Documents`, Server sau khi thực hiện xong sẽ có kết quả như sau:



Hình 4: Các thư mục được tạo trong đường dẫn

4.1.4.2. Di chuyển tệp/thư mục

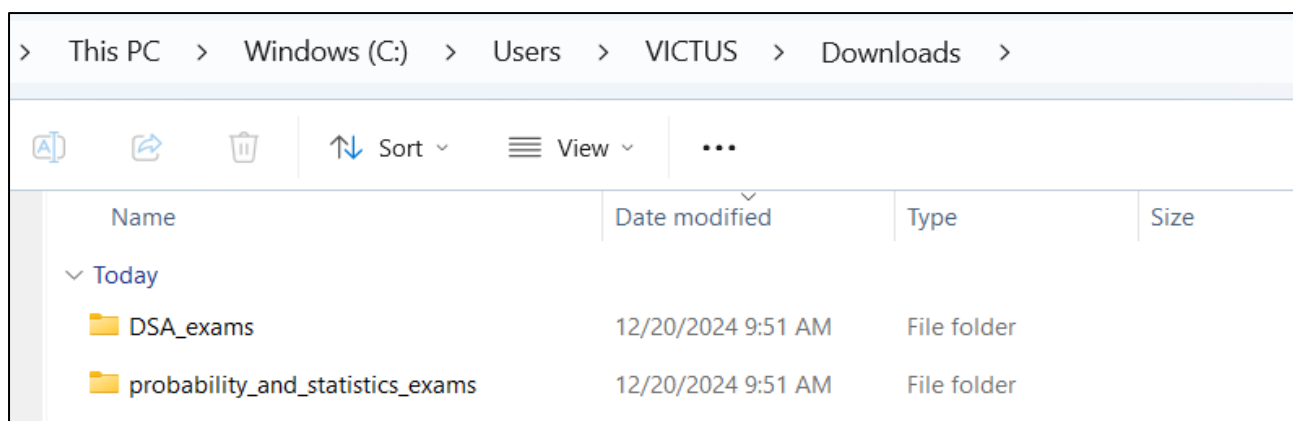
Đây là chức năng mở rộng. Chức năng này nhận vào các tham số:

- **type**: tùy chọn tạo tệp (file) hay thư mục (folder)
- **names**: tên của các tệp/thư mục
- **source**: đường dẫn gốc chứa các tệp/thư mục
- **destination**: đường dẫn đích đến cho các tệp/thư mục

Khi đã hoàn tất, Server sẽ trả về trạng thái là thành công (**success**) hoặc lỗi (**error**).

Ví dụ: Khi người dùng yêu cầu di chuyển các thư mục có tên

probability_and_statistics_exams, **DSA_exams** từ đường dẫn **C:\Users\VICTUS\Documents** đến đường dẫn **C:\Users\VICTUS\Downloads**, Server sau khi thực hiện xong sẽ có kết quả như sau:



Hình 5: Các thư mục được di chuyển đến đường dẫn

4.1.4.3. Sao chép tệp/thư mục

Chức năng này nhận vào các tham số:

- **type**: tùy chọn tạo tệp (file) hay thư mục (folder)
- **names**: tên của các tệp/thư mục
- **source**: đường dẫn gốc chứa các tệp/thư mục
- **destination**: đường dẫn đích đến cho các tệp/thư mục

Khi đã hoàn tất, Server sẽ trả về trạng thái là thành công (**success**) hoặc lỗi (**error**).

4.1.4.4. Xóa tệp/thư mục

Chức năng này nhận vào các tham số:

- **type**: tùy chọn tạo tệp (file) hay thư mục (folder)
- **names**: tên của các tệp/thư mục
- **path**: đường dẫn chứa các tệp/thư mục

Khi đã hoàn tất, Server sẽ trả về trạng thái là thành công (**success**) hoặc lỗi (**error**).

4.1.4.5. Liệt kê các tệp, thư mục từ một đường dẫn [4]

Đây là chức năng mở rộng. Chức năng này nhận vào các tham số:

- **path**: đường dẫn chứa các tệp/thư mục sẽ được liệt kê

Khi đã hoàn tất, Server sẽ trả về một tệp **.csv** chứa các tệp/thư mục hiện có trong đường dẫn với các trường (field):

- **Name**: Tên của tệp/thư mục
- **LastWriteTime**: Thời gian chỉnh sửa gần nhất
- **Length**: Kích thước tệp/thư mục. Đối với một thư mục, hàm sẽ đo kích thước của tất cả các tệp và thư mục con ở mọi tầng để tìm ra kích thước tổng chính xác nhất.
- **Mode**: Thuộc tính hoặc quyền của tệp/thư mục. Các thông tin này được biểu thị bằng một ký tự cụ thể:

| Kí tự | Ý nghĩa |
|----------|---|
| d | Biểu thị một thư mục |
| a | Biểu thị một tệp lưu trữ (được sử dụng bởi các công cụ sao lưu) |
| r | Biểu thị tệp chỉ đọc (không thể ghi hay chỉnh sửa) |
| h | Biểu thị tệp/thư mục ẩn |
| s | Biểu thị tệp hệ thống |
| l | Biểu thị một liên kết tượng trưng (shortcut hay con trỏ đến tệp khác) |
| - | Biểu thị thuộc tính này không có ở tệp/thư mục |

Ví dụ: `d--h--` biểu thị một thư mục ẩn; `-a----` biểu thị một tệp.

`PSIsContainer`: Tài nguyên là tệp hay thư mục.

Câu lệnh **Powershell** để chạy chức năng này có nội dung như sau:

```
Get-ChildItem -Path "C:\My\Directory" -Force | ForEach-Object {
    if ($_.PSIsContainer) {
        $folderSize = (Get-ChildItem -Path $_.FullName -File -Recurse -
Force | Measure-Object -Property Length -Sum).Sum
        [PSCustomObject]@{
            Name           = $_.Name
            LastWriteTime = $_.LastWriteTime
            Length         = $folderSize
            Mode           = $_.Mode
            PSIsContainer = $_.PSIsContainer
            Directory      = "C:\My\Directory"
        }
    } else {
        [PSCustomObject]@{
            Name           = $_.Name
            LastWriteTime = $_.LastWriteTime
            Length         = $_.Length
            Mode           = $_.Mode
            PSIsContainer = $_.PSIsContainer
            Directory      = "C:\My\Directory"
        }
    }
} | ConvertTo-Json -Depth 2
```

Lưu ý: Trong đoạn code trên, tùy chọn `-Force` đảm bảo rằng các tệp/thư mục ẩn đều được đo, `-Depth 2` là tùy chọn để lấy thông tin của các thư mục lồng nhau.

Ví dụ: Khi người dùng yêu cầu liệt kê các tệp và thư mục trong đường dẫn `D:\\vstudio`, Server sau khi thực hiện xong sẽ trả về một tệp `.csv` có nội dung như sau:

| | A | B | C | D | E |
|----|-------------------------------------|------------------|--------|--------------------|---------------|
| 1 | Name | LastWriteTime | Length | Mode | PSIsContainer |
| 2 | 2048 | 6/5/2024 16:56 | 221 MB | Directory | Directory |
| 3 | bt11 | 5/21/2024 11:32 | 14 KB | Directory | Directory |
| 4 | btvn1 | 4/14/2024 18:30 | 131 MB | Directory | Directory |
| 5 | btvn11 | 5/25/2024 10:28 | 16 KB | Directory | Directory |
| 6 | btvn12 | 12/6/2024 9:14 | 217 MB | Directory; Archive | Directory |
| 7 | btvn5 | 4/14/2024 10:10 | 280 MB | Directory | Directory |
| 8 | btvn9 | 5/12/2024 9:47 | 14 KB | Directory | Directory |
| 9 | btvn_test | 4/15/2024 19:17 | 205 MB | Directory | Directory |
| 10 | curl_project | 12/12/2024 21:00 | 14 GB | Directory | Directory |
| 11 | fileReading | 6/3/2024 10:31 | 53 MB | Directory | Directory |
| 12 | 23120143.zip | 11/2/2024 10:14 | 3 KB | File; Archive | File |
| 13 | btvn11.sln | 11/2/2024 10:14 | 1 KB | File; Archive | File |
| 14 | command.exe | 11/2/2024 11:03 | 80 KB | File; Archive | File |
| 15 | folder_contents-20241102_102305.csv | 11/2/2024 11:03 | 2 KB | File; Archive | File |
| 16 | | | | | |

Hình 6: Các tệp và thư mục đã được liệt kê

4.1.4.6. Lưu tệp được gửi từ Gmail vào Server

Đây là chức năng nâng cao. Chức năng này nhận vào các tham số:

- **path**: Đường dẫn trên máy Server mà tệp sẽ được lưu
- **attachment**: Chuỗi JSON chứa thông tin về tệp đính kèm, bao gồm **access_token**, **email_id**, **attachment_id**, **attachment_name**. Các thông tin này sẽ được tự động cung cấp bởi Client.

Cách thức hoạt động của chức năng này được mô tả như sau:

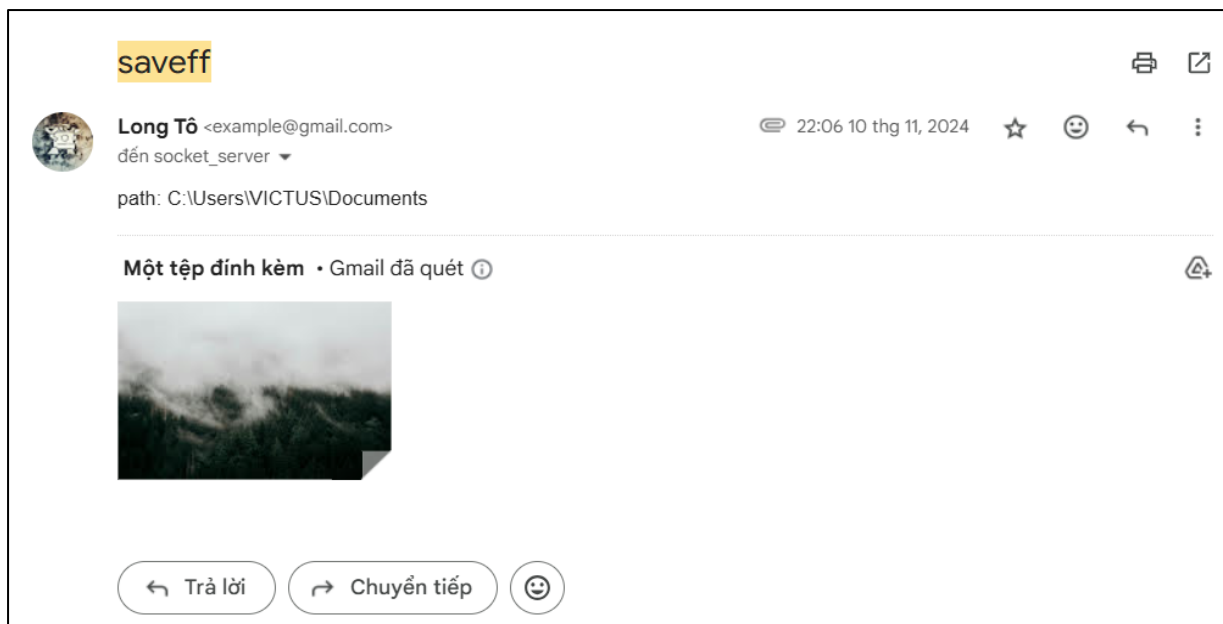
- **Tạo yêu cầu**: Server sẽ tạo một HTTP GET đến endpoint của Gmail xác thực bằng **access_token** ở phần header:

```
https://gmail.googleapis.com/gmail/v1/users/me/messages/{email_id}/
attachments/{attachment_id}
```

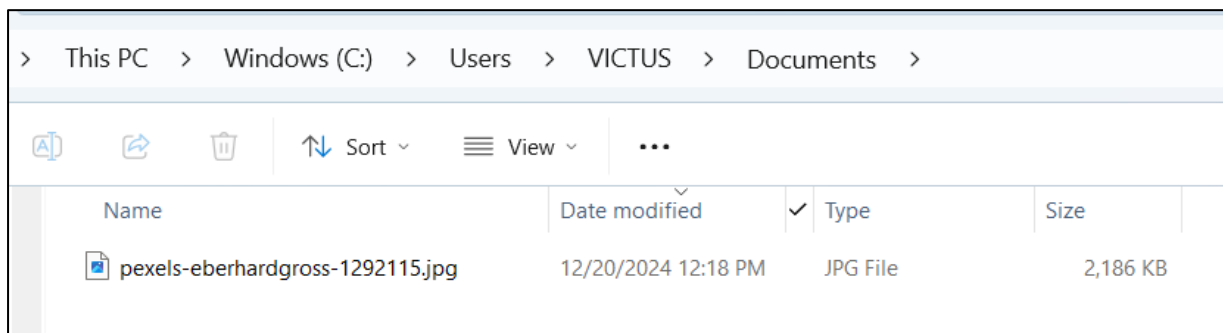
- **Xử lý phản hồi**: Dữ liệu của tệp đính kèm trả về dưới dạng một chuỗi được mã hoá dưới dạng **URL-safe Base64**, một dạng nâng cấp của Base64 giúp dữ liệu được truyền đi một cách an toàn và chính xác qua các liên kết hoặc tệp. Chuỗi mã hoá này sẽ được Server giải mã và ghi vào tệp với định dạng tương ứng.

Khi đã hoàn tất, Server sẽ trả về trạng thái là thành công (**success**) hoặc lỗi (**error**).

Ví dụ: Email người dùng yêu cầu lưu tệp ảnh vào Server có định dạng như sau:



Hình 7: Tệp ảnh được yêu cầu lưu vào Server



Hình 8: Tệp ảnh sau khi được lưu

4.1.4.7. Gửi nhiều tệp/thư mục từ Server về Gmail

Đây là chức năng nâng cao. Chức năng này nhận vào các tham số:

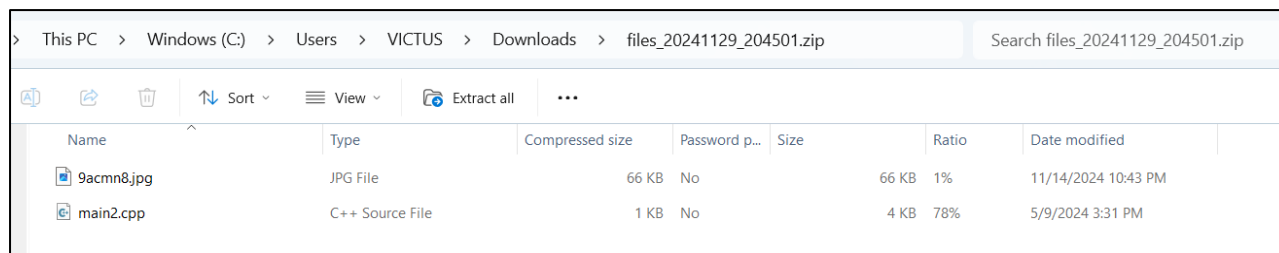
- **paths**: Đường dẫn của các tệp/thư mục sẽ được gửi về

Sau khi hàm đã nén các tệp/thư mục từ các đường dẫn vào một tệp **.zip** duy nhất, Server sẽ gửi tệp nén này về cho Client. Đây là câu lệnh Powershell để thực hiện chức năng này:

```
Compress-Archive -Path PATH1,PATH2,PATH3 -DestinationPath ZIP_FILENAME -Force
```

Ví dụ: Khi người dùng yêu cầu gửi 2 tệp **9acmn8.jpg**, **main2.cpp** từ đường dẫn

C:\Users\VICTUS\Downloads về Gmail, Server sau khi thực hiện xong sẽ gửi một email có chứa tệp nén có nội dung như sau:



Hình 9: Tập nén sau khi được tải về ở máy người dùng

4.1.5. Chụp màn hình [10]

4.1.5.1. Chụp một lần

Chức năng này nhận vào các tham số:

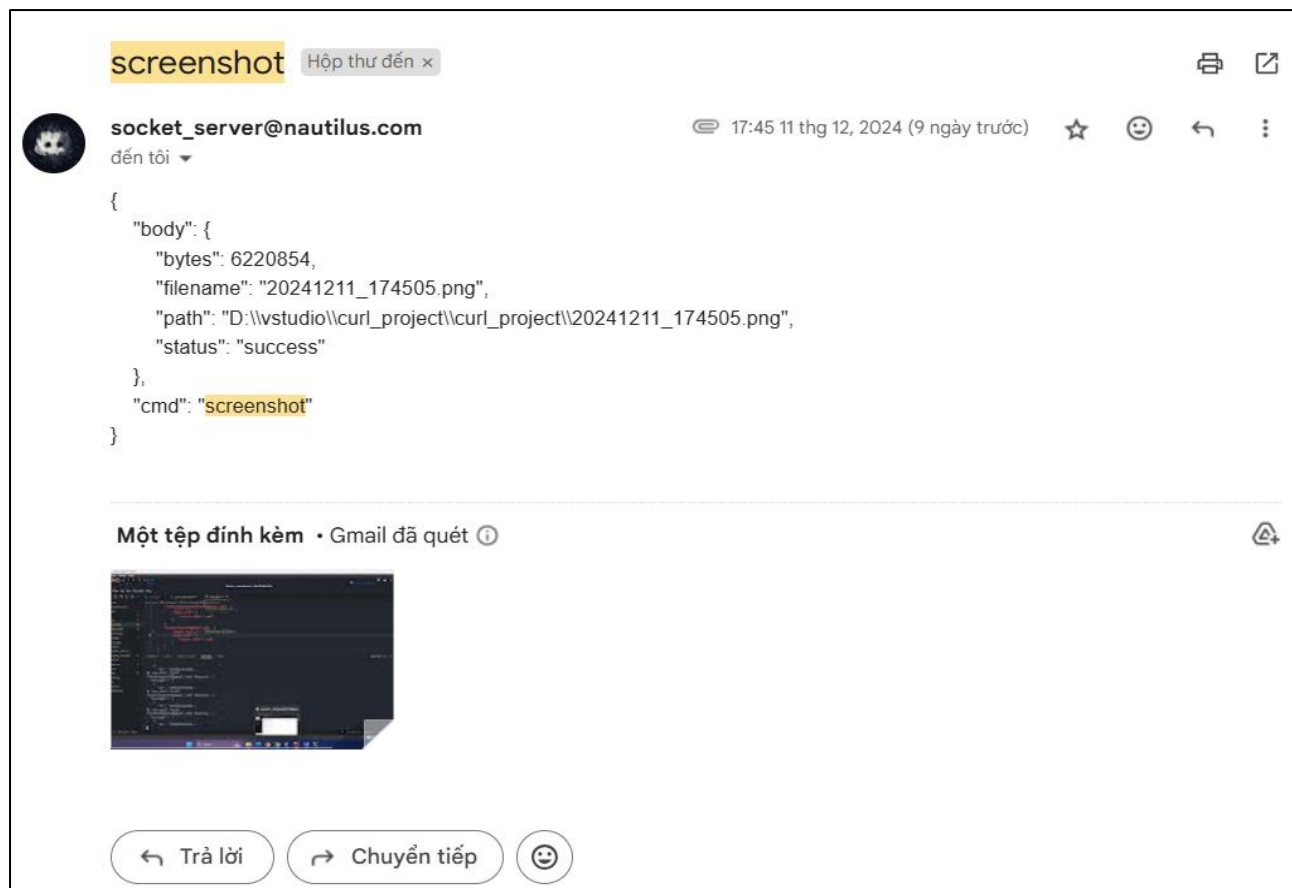
- **save_path**: Đường dẫn để lưu ảnh chụp màn hình (không bắt buộc).

Chức năng này sẽ lập tức chụp màn hình và gửi tệp ảnh đã chụp về cho Client. Có hai hàm chính là **capture()** và **saveBitmapToFile()** hỗ trợ thực hiện chức năng này. Quá trình thực hiện diễn ra như sau:

- **Kiểm tra tương thích DPI (Dots per inch)**: Đảm bảo Server tương thích với các màn hình có độ phân giải cao
- **Xác định thông tin màn hình**: Lấy thông tin về kích thước (chiều dài và chiều rộng) của màn hình chính (chưa kiểm tra màn hình phụ).
- **Tạo bitmap**: Tạo một **bitmap** tương thích với màn hình và sử dụng bitmap này để sao chép nội dung màn hình.
- **Thu thập thông tin bitmap**: Lấy thông tin cấu trúc của bitmap, bao gồm chiều dài, chiều rộng và số lượng bit màu. Hệ màu được sử dụng ở đây là **RGB**, với mỗi màu được biểu diễn bằng 8 bit, tạo thành tổng cộng 24 bit màu.
- **Trích xuất dữ liệu bitmap**: Lấy dữ liệu pixel từ bitmap và lưu trữ trong một vùng nhớ tạm.
- **Ghi tệp ảnh**: Đọc và ghi dữ liệu từ vùng nhớ tạm vào trong tệp ảnh.
- **Dọn dẹp tài nguyên**: Giải phóng vùng nhớ tạm đã tạo ra trước đó.

Sau khi hoàn thành, Server sẽ gửi về tệp ảnh chụp màn hình đã được lưu trước đó.

Ví dụ: Khi người dùng yêu cầu chụp màn hình, email mà họ nhận được sẽ được đính kèm tệp ảnh.



Hình 10: Email đính kèm ảnh chụp màn hình

4.1.5.2. Chụp nhiều lần theo chu kỳ

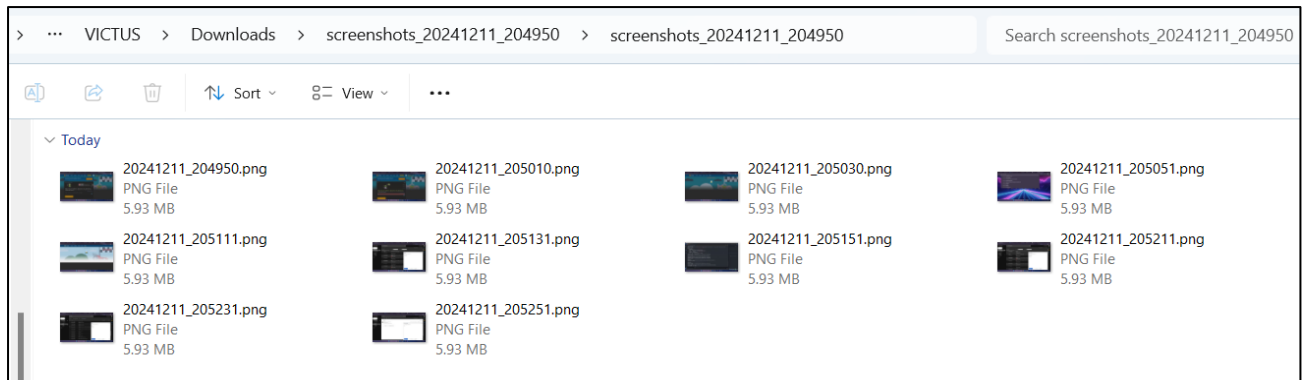
Chức năng này nhận vào các tham số:

- **interval**: Khoảng thời gian của chu kỳ
- **action**: Bật (**on**) hoặc tắt (**off**) luồng chạy chức năng này

Khi người dùng yêu cầu chụp theo chu kỳ, một luồng (thread) sẽ được Server tạo ra, chạy độc lập với luồng chính để tiến hành chụp màn hình 1 lần trong 1 chu kỳ, và lặp lại đến khi nào có yêu cầu dừng từ người dùng. Các ảnh chụp màn hình sẽ được lưu vào một thư mục và sẽ được nén lại thành tệp **.zip** để gửi khi luồng được dừng.

Lưu ý: Vì Gmail chỉ có phép đính kèm tệp có **dung lượng tối đa 25MB**, nên luồng sẽ tiếp tục chạy nhưng không thực hiện chụp màn hình nếu tệp nén đã **vượt quá 25MB**. Khi nào có yêu cầu nhận ảnh từ người dùng thì luồng mới được tắt và tệp nén chứa ảnh mới được gửi về.

Ví dụ: Khi người dùng yêu cầu chụp ảnh với chu kỳ 20 giây, họ sẽ nhận được một tệp nén có chứa các ảnh tương ứng:



Hình 11: Các ảnh chụp màn hình theo chu kỳ

4.1.6. Webcam

4.1.6.1. Chụp ảnh Webcam một lần

Công dụng: Chụp một ảnh bằng Webcam bên phía Server



Đây là chức năng mở rộng, chức năng này nhận vào tham số của mẫu Json:

- **cmd:** Tên của chức năng webcam chụp ảnh một lần
- **save_path:** Nơi lưu trữ các ảnh kết quả để về cho Client, nếu để trống thì mặc định là thư mục hiện tại

Cách hoạt động:

- **Bước 1:** Khi bên phía Admin muốn sử dụng chức năng chụp ảnh Webcam một lần, Client sẽ gửi một mẫu Json với tên lệnh là `cmd = webcam`, tham số ví dụ `save_path = D:\\` hoặc Admin có thể để trống thì bên phía Server sẽ tự hiểu nơi lưu là thư mục hiện tại.
- **Bước 2:** Khi nhận được yêu cầu, hệ thống sẽ kiểm tra xem Webcam của Server có đang bị chiếm giữ không. Nếu Webcam đang bị chiếm giữ thì gửi một phản hồi thất bại với mã `error_code` tương ứng. Nếu Webcam đang rảnh rỗi thì chuyển sang bước 3.
- **Bước 3:** Khi nhận hiệu lệnh thành công, Server sẽ tạo một folder làm nơi lưu kết quả theo ngày tháng năm, bằng thư viện `OpenCv[1]`, ta bật webcam nhờ khai báo biến `cv::VideoCapture cap(0)` với tham số `0` là tên webcam mặc định của máy tính. Sau đó ta sẽ chuyển từng frame ảnh của webcam vào một biến `cv::Mat frame` để lưu giữ frame đó. Bằng hàm `cv::imwrite()`, ta sẽ truyền vào tham số đầu tiên là đường dẫn nơi ta muốn lưu và kết thúc bằng tên của tên file ảnh, ví dụ: `folder_path + "\\ " + getUIDName() + ".png"`, tham số thứ hai chính là frame ảnh mà bạn muốn ghi vào trong file. Lưu ý nên bỏ qua khoảng 10-20 frame ảnh đầu tiên vì các frame đó sẽ bị đen do Webcam chưa thực sự sẵn sàng.
- **Bước 4:** Sau khi lưu kết quả vào trong thư mục đích, ta sẽ nén thư mục bằng hàm `zipFolder()` nếu không thể nén thì gửi một phản hồi và in ra `error_code`, ngược lại nếu nén được thì gửi một phản hồi thành công bằng hàm `sendMessage()` và gửi file bằng hàm `sendZipFile()`

VD: Khi người muốn chụp ảnh và lưu nó vào ổ đĩa D: thì kết quả bên phía Server sẽ như sau:

| | |
|---|-----|
|  5112024_1325_WEBCAM.zip | D:\ |
|  5112024_1347_WEBCAM | D:\ |

Hình 12: Ảnh chụp bằng webcam trả về file .zip

4.1.6.2. Chụp ảnh webcam trong một khoảng thời gian

Công dụng: Chụp ảnh Webcam trong một khoảng thời gian cho đến khi có lệnh tắt.

Đây là chức năng mở rộng, chức năng này nhận vào tham số là mẫu Json như sau:

- **cmd**: là tên của chức năng.
- **status**: on/off để điều khiển trạng thái chức năng.
- **interval**: second(s) là khoảng thời gian chụp lại một lần.
- **save_path**: là nơi lưu trữ thư mục kết quả.

Cách hoạt động:

- **Bước 1:** Khi bên phía Admin muốn sử dụng chức năng chụp ảnh nhiều lần trong một khoảng thời gian của Webcam, Client sẽ gửi một mẫu Json với tên lệnh là **cmd = webcamitv**, **interval = 5** (giả sử là 5s/1 lần), **action = on/off** dựa vào yêu cầu người dùng và **save_path = D:** (có thể để trống, tự hiểu là thư mục hiện tại).
- **Bước 2:** Sau khi nhận được lệnh xong thì máy chủ xem Webcam có đang bị chiếm dụng bởi một chức năng khác thông qua biến **running != 0**. Nếu bị chiếm dụng bởi một chức năng khác từ trước thì gửi phản hồi và in ra **error_code**. Ngoài ra nếu Webcam đang bị chiếm dụng bởi chính chức năng này mà bạn cố tình gửi thêm lệnh **status = on** lần nữa thì Server sẽ báo không thành công và in ra mã **error_code** và tương tự như trường hợp Webcam tắt mà Client gửi lệnh **action = off**.
- **Bước 3:** Đối với trường hợp Webcam đang rảnh và bạn gửi lệnh **action = on**, Server sẽ gửi một phản hồi đã bắt đầu chức năng với **status = started** bằng hàm **sendConfirmMessage()**. Sau đó Server bắt đầu chức năng bằng hàm **getWebcamEveryInterval()**, gán biến **running = 9** để ra dấu hiệu rằng webcam đã bị chiếm dụng, đồng thời tạo một luồng bằng biến luồng toàn cục **webcamThread** để lưu giữ luồng này. Mục đích của việc đa luồng là để cho Server có thể tiếp tục nhận Request từ Client mà vẫn thực hiện chức năng của Webcam.
- **Bước 4:** Cách hoạt động của chức năng chụp nhiều lần trong một khoảng thời gian khá giống với chức năng chụp Webcam một lần, chỉ khác ở chỗ bây giờ hàm của bạn sẽ được đặt trong một cái **while(running)** và chạy cho đến khi nào nhận được lệnh **action = off** của người dùng.
- **Bước 5:** Khi bạn muốn tắt chức năng này bằng cách gửi lệnh từ Client với **action = off** Server sẽ gán biến **running = 0** để giải phóng Webcam. Khi đó vòng lặp **while(running)** sẽ kết thúc. Hàm sẽ đóng gói folder bằng hàm **zipFolder()**, sau đó gửi một phản hồi bằng hàm **sendMessage()** và gửi file bằng hàm **sendZipFile()**.

VD: Khi người muốn chụp ảnh mỗi 5s/1 lần và lưu nó vào ổ đĩa **D:** sau 10s thì Admin tắt thì kết quả sẽ như sau:



Hình 13: Ảnh chụp bằng webcam theo chu kỳ và lưu vào ổ D

4.1.6.3. Chức năng quay Video bằng Webcam

Công dụng: Quay một đoạn video bằng Webcam của Server đến khi nhận lệnh tắt của Server.

Đây là chức năng trọng tâm, chức năng này nhận vào tham số là mẫu Json như sau:

- **cmd:** là tên của chức năng.
- **action:** on/off để điều khiển trạng thái chức năng.
- **save_path:** là nơi lưu trữ thư mục kết quả.

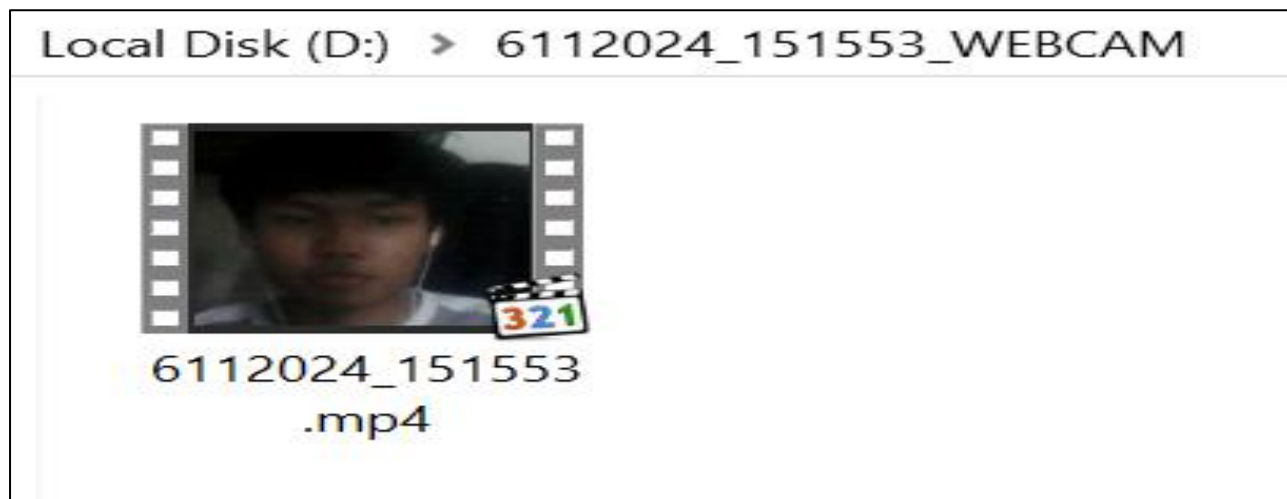
Cách hoạt động:

- **Bước 1:** Khi bên phía Admin muốn sử dụng chức năng chụp ảnh nhiều lần trong một khoảng thời gian của Webcam, Client sẽ gửi một mẫu Json với tên lệnh là **cmd = webcamvideo**, **action = on/off** tùy vào yêu cầu người dùng và **save_path = D:** (có thể để trống, tự hiểu là thư mục hiện tại).
- **Bước 2:** Sau khi nhận được lệnh xong thì máy chủ xem Webcam có đang bị chiếm dụng bởi một chức năng khác thông qua biến **running != 0**. Nếu bị chiếm dụng bởi một chức năng khác từ trước thì gửi phản hồi và in ra **error_code**. Ngoài ra nếu Webcam đang bị chiếm dụng bởi chính chức năng này mà bạn cố tình gửi thêm lệnh **action = on** lần nữa thì Server sẽ báo không thành công và in ra mã **error_code** và tương tự như trường hợp Webcam tắt mà Client gửi lệnh **action = off**.
- **Bước 3:** Đối với trường hợp Webcam đang rảnh và bạn gửi lệnh **action = on**, Server sẽ gửi một phản hồi đã bắt đầu chức năng với **status = started** bằng hàm **sendConfirmMessage()**. Sau đó Server bắt đầu chức năng bằng hàm **getWebcamVideo()** gán biến **running = 11** để ra dấu hiệu rằng webcam đã bị chiếm dụng, đồng thời tạo một luồng bằng biến luồng toàn cục

`webcamThread` để lưu giữ luồng này. Mục đích của việc đa luồng là để cho Server có thể tiếp tục nhận Request từ Client mà vẫn thực hiện chức năng của Webcam.

- **Bước 4:** Khi vào hàm `getWebcamVideo()`, ta sẽ chuẩn bị sẵn Webcam bật bằng cách sử dụng biến `cv::VideoCapture cap(0)` với `0` là Webcam mặc định bằng thư viện OpenCv [7]. Sau đó ta sẽ điều chỉnh thông số cấu hình cho video của mình như `FRAME_SIZE` là chiều dài, chiều rộng của từng frame, khai báo biến `fps` để thông báo muốn lấy bao nhiêu frame trong một giây và định dạng `('M', 'P', '4', 'V')` sử dụng để ghi các video có định dạng `.mp4` và `.avi`
- **Bước 5:** Sau khi có các thông số cấu hình cho đoạn video, ta sẽ khai báo biến `cv::VideoWriter video` và dùng hàm `video.write(frame)` có sẵn và ghi frame mình muốn lưu vào. Hàm quay video đến khi nào dùng lệnh `stauts = off` hoặc video đã vượt qua dung lượng 25MB.
- **Bước 6:** Khi bạn muốn tắt chức năng này bằng cách gửi lệnh từ Client với `action = off`, Server sẽ gán biến `running = 0` để giải phóng Webcam. Khi đó vòng lặp `while(running)` sẽ kết thúc. Hàm sẽ đóng gói folder bằng hàm `zipFolder()`, sau đó gửi một phản hồi bằng hàm `sendMessage()` và gửi file bằng hàm `sendZipFile()`.

VD: Khi người quay video qua Webcam và lưu nó vào ổ đĩa `D:\\` sau 10s thì Admin tắt thì kết quả sẽ như sau:



Hình 14: Ảnh quay video bằng webcam và lưu vào ổ D

4.1.7. Keylogger [2]

Công dụng: Cung cấp chức năng ghi lại các thao tác bàn phím từ thiết bị người dùng, phục vụ mục đích giám sát hoạt động hoặc thu thập dữ liệu đầu vào từ bàn phím theo yêu cầu.

Cách hoạt động:

- **Bước 1:** Người dùng gửi yêu cầu thông qua lệnh `cmd = keyloggeritv` kèm các tham số:
 - `exclude_keys`: Danh sách phím cần loại trừ (nếu có), định dạng là một chuỗi các phím cách nhau bằng dấu phẩy.

- `exclude_apps` : Danh sách ứng dụng cần loại trừ (nếu có), định dạng là một chuỗi tên ứng dụng cách nhau bằng dấu phẩy.
- `interval` : Thời gian (tính bằng giây) giữa các lần tạo file ghi dữ liệu.
- `save_path` : Thư mục lưu trữ dữ liệu, nếu không được cung cấp, hệ thống sẽ tạo một thư mục mới với tên mặc định.
- Bước 2: Quá trình xử lý thông qua một số hàm chính:

Xác nhận yêu cầu từ client và gửi phản hồi ban đầu qua hàm `send()`. Thông báo trạng thái `started`, keylogger đã bắt đầu chạy hoặc `stopped`, `keylogger` dừng và dữ liệu đã được nén, sẵn sàng gửi.

 - *Ghi nhận thao tác bàn phím*:

Thiết lập một `Hook` hệ thống thông qua hàm `SetWindowsHookEx()` để theo dõi và ghi nhận các phím được nhấn.

Loại trừ phím và ứng dụng:

 - Kiểm tra ứng dụng đang hoạt động thông qua hàm `GetWindowApp()`.
 - Bỏ qua các phím/ứng dụng có trong danh sách loại trừ `exclude_keys` và `exclude_apps`.
 - *Tạo và quản lý file log*:

Sử dụng hàm `steady_clock` để xác định thời gian giữa các lần ghi file.

Tạo file mới với tên chứa timestamp (một giá trị thể hiện thời điểm cụ thể trong thời gian) khi khoảng thời gian `interval` kết thúc hoặc khi lần ghi đầu tiên bắt đầu.

Ghi các phím nhấn, bao gồm cả các phím đặc biệt như `Enter`, `Backspace`, `Caps Lock`, `Tab`,... vào file.
 - *Kết thúc keylogger*:

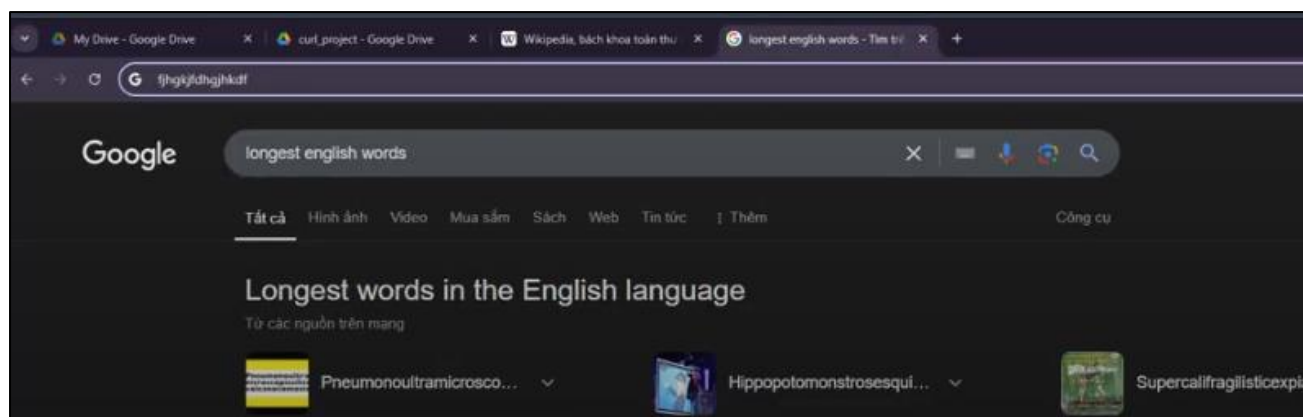
Dừng `Hook` hệ thống qua hàm `UnhookWindowsHookEx()` khi lệnh stop được gửi từ client.

Nén thư mục log thành file `.zip` bằng PowerShell (Compress-Archive) và gửi file nén về client qua hàm `sendFile()`.
- Bước 3: Quá trình keylogger hoàn tất với các kết quả sau:

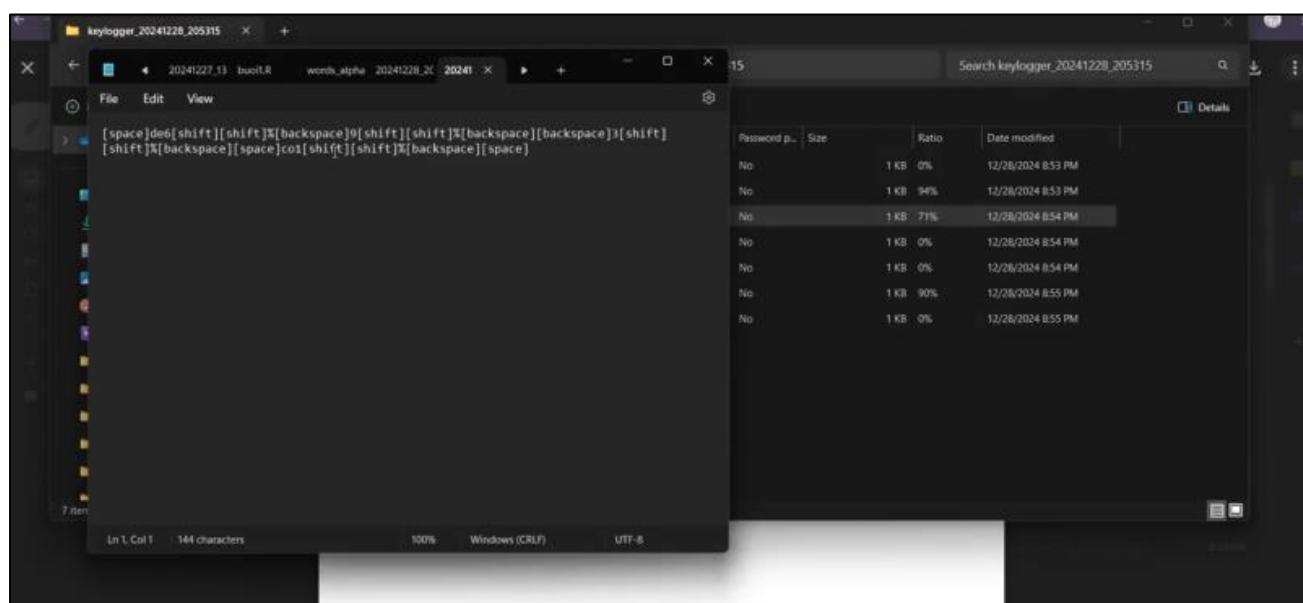
File log chứa thao tác bàn phím được ghi lại và lưu trong thư mục đã chỉ định hoặc mặc định.

Thư mục log được nén thành file `.zip` và gửi trả về client.

Hệ thống phản hồi đầy đủ trạng thái của lệnh qua mô hình `client-server`.



Hình 15: Gõ phím trên một ứng dụng bất kỳ trên thiết bị server



Hình 16: Trả về các file lưu phím ấn sau một khoảng chu kỳ cho client

4.2. Phía client (Linux)

Linux được lựa chọn làm hệ điều hành cho máy Client vì nó các yếu tố kỹ thuật phù hợp, chi phí và tính linh hoạt. Dưới đây là một số lý do chính:

- Linux là một hệ điều hành mã nguồn mở, cho phép người dùng kiểm tra, sửa đổi và tối ưu hóa mã nguồn theo nhu cầu riêng. Điều này giúp giảm chi phí tổng thể khi vận hành máy Client.
- Linux được biết đến với khả năng xử lý hiệu quả, đặc biệt khi xử lý khối lượng công việc lớn hoặc duy trì hoạt động trong thời gian dài.
- Linux có mô hình bảo mật mạnh mẽ với quyền truy cập dựa trên người dùng, nhóm và quyền hạn (permissions).
- Cho phép tắt hoặc loại bỏ các thành phần không cần thiết, giúp tối ưu hóa tài nguyên hệ thống.
- Linux hỗ trợ nhiều loại phần cứng khác nhau, từ các server mạnh mẽ đến những máy cấu hình thấp, đảm bảo khả năng chạy trên hầu hết mọi thiết bị.

- Đặc biệt, Linux được trang bị các công cụ mạnh mẽ để quản lý mạng, như `iptables`, `ip`, `tcpdump`, và các tính năng tích hợp sẵn để hỗ trợ mạng phức tạp.

4.2.1. Gmail API [6]

4.2.1.1. Thiết lập giao tiếp bằng API

Trong phần này, ta sẽ tập trung vào quá trình thiết lập giao tiếp với **Gmail Server** thông qua API bằng **Access Token**. Quy trình được chia thành các giai đoạn cụ thể, bao gồm: cấu hình dự án và thông tin xác thực trên **Google Cloud Console**, triển khai luồng xác thực **OAuth 2.0** để lấy **Access Token**, và thực hiện các truy vấn đến Gmail API.

Để bắt đầu, một dự án cần được tạo và cấu hình trên **Google Cloud Console**. Đây là một giao diện quản lý trực tuyến, cung cấp các công cụ cần thiết để cấu hình, quản lý, và giám sát các dịch vụ được cung cấp bởi **Google Cloud Platform**. Các bước chính để cấu hình như sau:

- **Tạo dự án:** Một dự án mới được thiết lập để quản lý quyền truy cập API.
- **Kích hoạt Gmail API:** Dịch vụ Gmail API được kích hoạt trong mục "API & Services".
- **Tạo thông tin xác thực:** Thông tin xác thực loại OAuth 2.0 Client ID được tạo ra, với các trường **client_id** và **client_secret** đóng vai trò quan trọng trong quá trình xác thực.

Thông tin xác thực này sẽ được lưu trữ trong tệp JSON và sử dụng trong các bước tiếp theo để thực hiện giao tiếp với Gmail Server.

Tiếp theo đó, ta cần thực hiện một bước quan trọng là xác thực **OAuth 2.0**. Thay vì chia sẻ trực tiếp thông tin đăng nhập của người dùng, **OAuth 2.0** sử dụng cơ chế cấp quyền thông qua **Access Token**, đảm bảo tính bảo mật và giảm thiểu rủi ro rò rỉ thông tin nhạy cảm. Quá trình này được thực hiện theo các bước sau:

- **Lấy mã xác thực (Authorization Code):**

Ứng dụng gửi yêu cầu HTTP GET đến endpoint của Google, bao gồm các tham số: **client_id**, **redirect_uri**, **response_type**, và **scope**. Endpoint này trả về mã xác thực, biểu thị sự đồng ý của người dùng. Đây là một URL để minh họa:

```
https://accounts.google.com/o/oauth2/v2/auth?client_id=1234.abcd.apps.googleusercontent.com&redirect_uri=http://example.com&response_type=code&scope=https://mail.google.com/
```

- **Khi xác thực thành công, máy chủ sẽ chuyển hướng người dùng qua **redirect_uri** có chứa mã xác thực:**

```
http://localhost/?code=1234-abcd&scope=https://mail.google.com/
```

- **Trao đổi mã xác thực để lấy Access Token:**

Để lấy Access Token, người dùng tiếp tục gửi mã xác thực đến endpoint của Google thông qua HTTP POST. Các tham số trong payload của yêu cầu bao gồm **code**, **client_id**, **client_secret**, **redirect_uri** và **grant_type**. Đây là một câu lệnh **curl** minh họa:

```
curl --request POST \
--data "code=1234-abcd&client_id=5678-efgh.apps.googleusercontent.com&client_secret=XXXX_YYYY&redirect_uri=http://example.com&grant_type=authorization_code" \
https://oauth2.googleapis.com/token
```

- **Nếu các thông tin được gửi đúng và hợp lệ, máy chủ sẽ trả về một phản hồi dạng JSON như sau:**

```
{
  "access_token": "ya29.a0ARrdaM...",
  "expires_in": 3600,
  "refresh_token": "1//0gJv...",
  "scope": "https://mail.google.com/",
  "token_type": "Bearer"
}
```

- **Gia hạn Access Token:**

Khi Access Token hết hạn, Client sử dụng **Refresh Token** để lấy Access Token mới mà không cần yêu cầu người dùng xác thực lại. Điều này đảm bảo tính liên tục trong quá trình giao tiếp với Gmail API. Tương tự với thao tác lấy Access Token, các tham số trong payload của HTTP POST bao gồm **client_id**, **client_secret**, **refresh_token** và **grant_type**. Đây là một câu lệnh **curl** minh họa:

```
curl --request POST \
--data "client_id=5678-efgh.apps.googleusercontent.com&client_secret=XXXX_YYYY&refresh_token=1234-abcd&grant_type=refresh_token" \
```

```
https://oauth2.googleapis.com/token
```

Phản hồi JSON trả về tương tự như phản hồi ở trên.

4.2.1.2. Xử lý yêu cầu từ người dùng

Quá trình người dùng yêu cầu thực thi lệnh trên server cần phải thông qua một bước trung gian, chính là Client. Client sẽ đọc các emails của người dùng để lấy yêu cầu thực thi và gửi email phản hồi về khi đã thực thi xong. Với Access Token đã lấy được, Client có thể thực hiện các yêu cầu đến Gmail API để truy cập tài nguyên trong Gmail Server, bao gồm:

- **Gửi một email:**

Thao tác gửi một HTTP POST đến endpoint với dữ liệu được định dạng **MIME (Multipurpose Internet Mail Extensions)**. Đây là một tiêu chuẩn định dạng email, cho phép gửi các nội dung phức tạp như văn bản, hình ảnh, tệp đính kèm, hoặc HTML thông qua email. Đây mà một ví dụ về định dạng này:

```
From: sender@example.com
To: recipient@example.com
Subject: Email with Attachment
Content-Type: multipart/mixed; boundary="boundary123"

--boundary123
Content-Type: text/plain; charset="UTF-8"

This is the body of the email.

--boundary123
Content-Type: application/pdf
Content-Disposition: attachment; filename="example.pdf"
Content-Transfer-Encoding: base64

<BASE64_ENCODED_FILE_CONTENT>

--boundary123--
```

Trong ví dụ này, các thành phần gồm có:

- **From**: Địa chỉ email của người gửi
- **To**: Địa chỉ email của người nhận
- **Subject**: Dòng tiêu đề của email, mô tả nội dung email
- **Content-Type**: Định nghĩa các loại thành phần.
- **multipart/mixed**: Cho biết email này chứa nhiều thành phần với các nội dung khác nhau
- **boundary="boundary123"**: Định nghĩa ranh giới để phân biệt giữa các phần trong nội dung email.
- **text/plain; charset="UTF-8"**: Nội dung văn bản, sử dụng mã hoá ký tự **UTF-8**
- **Content-Disposition**: Chỉ định loại tệp đính kèm attachment; filename="example.pdf": Tên của tệp.
- **Content-Transfer-Encoding**: Phương pháp mã hoá để truyền tệp qua email, sử dụng mã hoá base64
- **<BASE64_ENCODED_FILE_CONTENT>**: Chuỗi dữ liệu của tệp đã được mã hoá base64 Client sẽ gửi một HTTP POST đến endpoint <https://gmail.googleapis.com/gmail/v1/users/me/messages/send>.

Đây là ví dụ yêu cầu để gửi:

```
curl --request POST \
  --url https://gmail.googleapis.com/gmail/v1/users/me/messages/send \
  --header "Authorization: Bearer ACCESS_TOKEN" \
  --header "Content-Type: application/json" \
  --data '{
    "raw": "BASE64_ENCODED_MIME_MESSAGE"
  }'
```

Thao tác này được thực thi khi Client nhận được phản hồi từ Server sau khi được yêu cầu chạy một chức năng bất kì. Phản hồi này có thể là thông báo chức năng chạy thành công, gửi về kết quả (đối với chức năng cơ bản); hoặc đã bắt đầu/tắt luồng (đối với chức năng yêu cầu chạy đa luồng); hoặc chức năng bị lỗi trong quá trình thực thi.

- **Truy xuất danh sách các emails:**

Để truy xuất danh sách các emails, Client sẽ gửi HTTP GET đến endpoint: <https://gmail.googleapis.com/gmail/v1/users/me/messages> với tham số truy vấn

q=from: là emails từ địa chỉ củ thể và **maxResults** chỉ số lượng tối đa các emails gần nhất. Đây là ví dụ yêu cầu để gửi:

```
curl --request GET \
  --url "https://gmail.googleapis.com/gmail/v1/users/me/messages?
q=from:example@gmail.com&maxResults=10" \
  --header "Authorization: Bearer ACCESS_TOKEN"
```

- **Phản hồi trả về có dạng như sau:**

```
{
  "messages": [
    { "id": "178e8f6df601dc41", "threadId": "178e8f6df601dc41" },
    { "id": "179e9f7ef702ed52", "threadId": "179e9f7ef702ed52" }
  ],
  "resultSizeEstimate": 2
}
```

Thao tác này sẽ tự động chạy trong Client mỗi 5 giây để kiểm tra email mới nhất được gửi từ người dùng. Nếu phát hiện email mới, lập tức lấy thông tin chi tiết để tạo yêu cầu gửi về Server. Lưu ý: Lý do ta chọn 5 giây để kiểm tra email thay vì khoảng thời gian ngắn hơn vì việc sử dụng Gmail API phải tuân theo giới hạn sử dụng hằng ngày. Mỗi **phương thức (Method)** sẽ có **một đơn vị hạn ngạch (Quota Units)** riêng và giới hạn cho mỗi tài khoản là 250 quota units mỗi giây. Trong đó phương thức gửi email (message.send) tốn 100 quotas, phương thức truy xuất email (message.list) tốn 5 quotas và phương thức lấy thông tin email (message.get) tốn 5 quotas. Ngoài ra, Client còn hỗ trợ đa người dùng, tức là thực hiện các thao tác này trên nhiều email cùng một lúc, nên trong 1 giây chỉ có thể hỗ trợ tối đa 2 người dùng.

- **Lấy thông tin chi tiết email:**

Ta có thể lấy thông tin chi tiết email bằng việc gửi HTTP GET đến endpoint

<https://gmail.googleapis.com/gmail/v1/users/me/messages/{id}>. Đây là một ví dụ:

```
curl --request GET \
  --url "https://gmail.googleapis.com/gmail/v1/users/me/messages/
178e8f6df601dc41 " \
```

```
--header "Authorization: Bearer ACCESS_TOKEN"
```

- **Phản hồi trả về có dạng JSON (đã được rút gọn):**

```
{
  "id": "178e8f6df601dc41",
  "threadId": "178e8f6df601dc40",
  "labelIds": ["INBOX", "IMPORTANT"],
  "snippet": "This is the first line or summary of the email content.",
  "historyId": "45758",
  "internalDate": "1672502395000",
  "payload": {
    "partId": "",
    "mimeType": "multipart/mixed",
    "filename": "",
    "headers": [...],
    "body": {
      "size": 0
    },
    "parts": [...]
  },
  "sizeEstimate": 1650
}
```

Với phản hồi trên, tên lệnh nằm ở phần **Tiêu đề** của email và các tham số (tùy vào từng chức năng) nằm ở phần **Nội dung** đã được giải mã. Từ đó, Client có thể khởi tạo một yêu cầu (request) gửi về server với định dạng JSON như sau:

```
{
  "cmd": COMMAND_ONE,
  "params": {
    "param_one": VALUE,
    ...
  }
}
```

4.2.2. Giao tiếp với Server

4.2.2.1. Gửi yêu cầu cho Server

Từ tiêu đề và nội dung email đã trích xuất được, Client có thể khởi tạo một **yêu cầu (request)** gửi về server với định dạng JSON như sau:

```
{
    "cmd": COMMAND_NAME,
    "params": {
        "param_one": VALUE,
        ...
    }
}
```

Sau đó, yêu cầu này sẽ được gửi đi dưới dạng một mảng ký tự (`buffer[BUFFER_SIZE]`) với số bytes bằng với kích thước của chuỗi yêu cầu này. Dữ liệu trong buffer sẽ được chuyển vào hàng đợi gửi của socket và được đóng gói thành các gói tin gửi bằng **giao thức TCP**. Có thể tham khảo hàm sau như ví dụ:

```
ssize_t send(int socket, const void *buffer, size_t length, int flags);
```

Server sẽ lắng nghe và nhận đủ số lượng bytes đã được gửi từ Client. Sau đó sao chép dữ liệu từ hàng đợi nhận vào vùng nhớ (mảng ký tự) buffer và chuyển thành dạng JSON để phân tích các yêu cầu. Có thể tham khảo hàm sau như ví dụ:

```
ssize_t recv(int socket, const void *buffer, size_t length, int flags);
```

4.2.2.2. Nhận phản hồi từ Server

Khi chức năng đã được thực thi xong, Client nhận một phản hồi (response) từ Server và phân loại theo tên lệnh để xử lý. Mẫu phản hồi có định dạng JSON như sau:

```
{
    "cmd": COMMAND_NAME,
    "body": {
        "status": VALUE,
        ...
    }
}
```

```
}
```

Đối với các lệnh như `screenshot`, `getff`, `listas`, kết quả được trả về ngay lập tức nên Client sẽ gửi email về cho người dùng (có thể đính kèm tệp) khi xử lý xong. Các chức năng này được phân thành 2 loại như sau:

- **Chức năng trả về trạng thái:** Server gửi một phản hồi chỉ chứa trạng thái của chức năng sau khi được thực thi.
- **Chức năng trả về tệp dữ liệu:** Trước tiên, Server gửi một phản hồi có chứa tên tệp (`filename`), đường dẫn (`path`) và kích thước tệp (`bytes`). Sau đó, dựa trên số lượng bytes đã có mà Client sẽ nhận từng đoạn (`chunks`) với kích thước `BUFFER_SIZE < bytes` và ghi vào tệp tương ứng đến khi nào đủ hết thì dừng.

Đối với các lệnh như `screenshotitv`, `webcamitv`, `keyloggeritv`, Client chỉ gửi kết quả về email người dùng khi họ yêu cầu dừng luồng chạy chức năng này. Trong khoảng thời gian luồng được chạy, người dùng có thể yêu cầu các chức năng khác vì nó sẽ được chạy song song với luồng chính. Mô hình đa luồng này sẽ được mô tả ở phần sau của báo cáo.

4.2.3. Mô hình quản lý đa người dùng

4.2.3.1. Tự động kết nối Server thông qua MAC

Trong các mạng nội bộ (LAN), **địa chỉ IP (Internet Protocol)** thường là yếu tố chính để nhận dạng các thiết bị. Tuy nhiên, địa chỉ IP có thể thay đổi theo thời gian do các nguyên nhân như sử dụng **giao thức DHCP**, cấu hình mạng không ổn định hoặc các yếu tố liên quan đến quản trị hệ thống. Điều này đặt ra thách thức lớn trong việc theo dõi và quản lý các thiết bị trong mạng. Do đó, Client sẽ xác định các máy Server trong mạng bằng địa chỉ **MAC (Media Access Control)**. Đây là địa chỉ vật lý duy nhất được gán cho card mạng (**Network Interface Card**) và không bị thay đổi theo thời gian.

Client sẽ chạy một hàm sử dụng công cụ **Nmap** để quét tất cả các địa chỉ IP khả dụng và trích xuất địa chỉ IP và MAC từ những thiết bị này. Câu lệnh trên Linux để thực hiện chức năng này:

```
sudo nmap -sn 192.168.1.0/24
```

Tuỳ chọn `-sn (Ping Scan)` chỉ kiểm tra thiết bị nào đang hoạt động. Với địa chỉ đường mạng `192.168.1.0/24`, subnet mask `255.255.255.0`, ta chỉ cần ping từ IP `192.168.1.1` đến `192.168.1.254`, tức là 254 lần để tìm ra tất cả các thiết bị. Dưới đây là ví dụ kết quả khi chạy câu lệnh trên:


```
Nmap scan report for 192.168.1.1
Host is up (0.0012s latency).
MAC Address: 3C:37:86:1A:4B:5C (Cisco Systems)

Nmap scan report for 192.168.1.10
Host is up (0.0045s latency).
MAC Address: B8:27:EB:45:89:AB (Raspberry Pi Foundation)

... more devices
```

Sau đó hàm sẽ tiến hành so sánh với địa chỉ MAC của các máy Server đã được cung cấp trong tệp JSON để chọn ra chính xác máy Server được kết nối. Địa chỉ IP sẽ được gán sau khi quá trình quét hoàn tất. Tệp JSON lưu trữ có dạng như sau:

```
{
  "example1@gmail.com": {
    "MAC_ADDRESS": "9A:8B:7C:6D:5E:4F",
    "IP_ADDRESS": "",
    "PORT": 8080,
    "STATUS": "active"
  },
  "example2@gmail.com": {
    "MAC_ADDRESS": "A1:B2:C3:D4:E5:F6",
    "IP_ADDRESS": "",
    "PORT": 8080,
    "STATUS": "inactive"
  }
}
```

Lưu ý: Quá trình quét toàn bộ mạng nội bộ chỉ mất thời gian khi máy Client khởi động lần đầu.

4.2.3.2. Quản lí song song nhiều Server

Mô hình quản lí này sử dụng một máy Client (chạy hệ điều hành Linux) duy nhất và thông qua cơ chế **đa luồng (multithreading)** để kiểm soát nhiều máy Server (chạy hệ điều hành Windows) cùng một lúc. Mỗi kết nối giữa Client và Server được quản lí độc lập trong một luồng (thread) riêng, có một kết

nổi TCP riêng, đảm bảo quá trình giao tiếp và xử lý các chức năng trở nên hiệu quả với nhiều người dùng.

Lý do sử dụng Linux cho máy Client là một lựa chọn lý tưởng nhờ vào khả năng xử lý đa luồng mạnh mẽ, sự ổn định và khả năng mở rộng cao, rất phù hợp với hệ thống Client cần duy trì nhiều kết nối TCP đồng thời và xử lý dữ liệu lớn từ nhiều Server. Ngoài ra Linux được đánh giá cao về bảo mật, đặc biệt trong các hệ thống giao tiếp mạng. Hơn nữa đây là hệ điều hành mã nguồn mở và miễn phí, giúp giảm thiểu chi phí vận hành và triển khai so với việc sử dụng các hệ điều hành thương mại. Điều này giúp tối ưu hóa hiệu suất và đảm bảo tính hiệu quả của hệ thống quản lý socket với nhiều kết nối đồng thời.

Về cách thức vận hành, mô hình được thiết lập thông qua các bước sau:

- **Tạo và quản lý các luồng:** Đối với mỗi email trong danh sách lưu trữ, hệ thống sẽ tạo một luồng riêng biệt và thiết lập kết nối với Server tương ứng. Client sử dụng cách thức tự động kết nối ở trên để xác định chính xác Server cần được kết nối. Dưới đây là đoạn code minh họa:

```
vector<thread> threads;
for (auto& [email, machine] : users.items()) {
    string ip_address = machine["IP_ADDRESS"];
    int port = machine["PORT"];
    threads.push_back(thread(clientThread, ip_address, port,
                             "client@gmail.com", email));
}

for (auto& thread : threads) {
    thread.join();
}
```

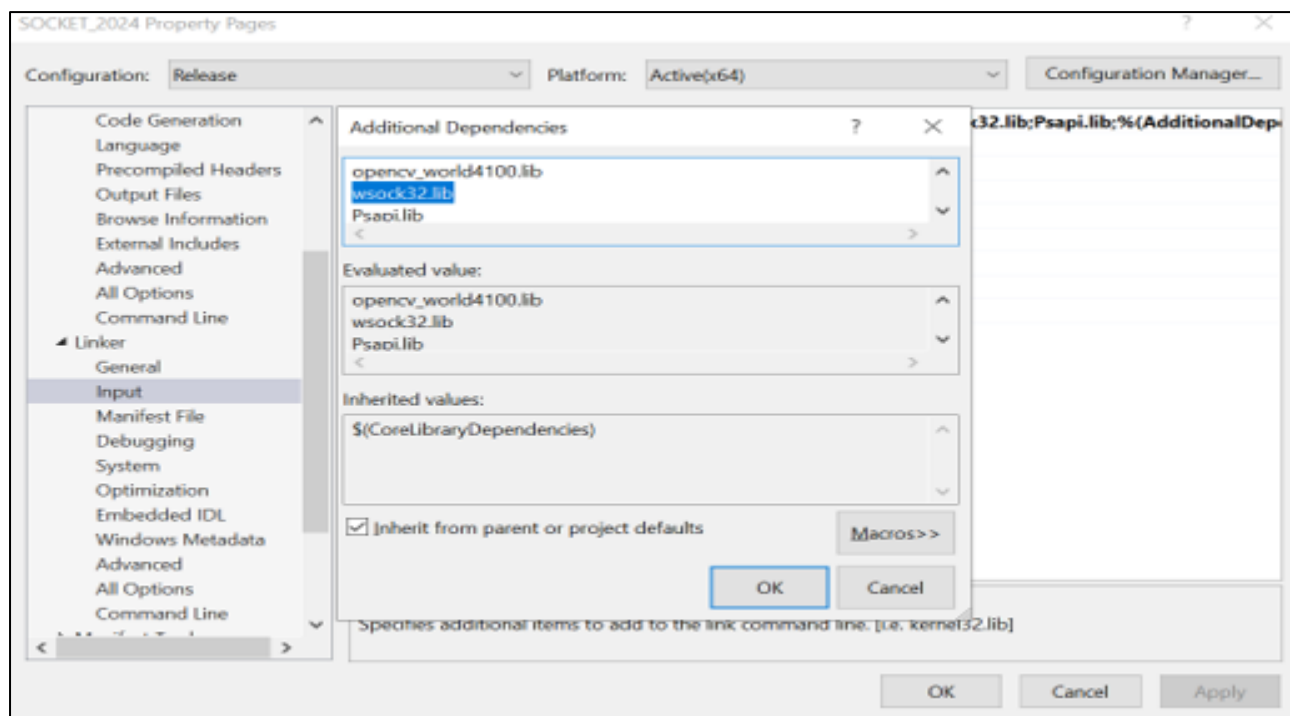
- **Thiết lập socket:** Khi đã có địa chỉ IP và Port (mặc định là 8080), mỗi luồng sẽ tự động thiết lập các bước cần thiết để xây dựng socket và kết nối TCP với Server.

Cuối cùng, mỗi luồng sẽ chạy một vòng lặp while, thời gian nghỉ 5 giây để nhận lệnh từ Gmail, yêu cầu Server xử lý và gửi email kết quả về cho người dùng.

4.3. Giao tiếp Socket

4.3.1. Cài thư viện

Sử dụng thư viện winsock32 và liên kết nó trong dự án của mình



Hình 17. Cài đặt thư viện winsock32

4.3.2. Tạo môi trường và cấu hình

4.3.2.1. Cấu hình thông tin của thư viện

```
WSADATA wsa;
if (WSAStartup(MAKEWORD(2, 2), & wsa) == 0) {
    cout << "Success in creating Environment!";
} else {
    cout << "Failed in creating Environment!";
    return -1;
}
```

Trong đó có các thông số sau:

- **WSAStartup**: Khởi tạo thư viện Winsock.
- **MAKEWORD(2, 2)**: Yêu cầu phiên bản Winsock 2.2.
- **WSADATA wsa**: Biến chứa thông tin cấu hình Winsock.

Nếu như cấu hình thành công thì in ra như trong đoạn mã và ngược lại.

4.3.2.2. Tạo Socket

```
SOCKET serverSocket = socket(AF_INET, SOCK_STREAM, 0);
if (serverSocket < 0) {
    cout << "Can not creat a socket " << endl;
```

```
    return -1;
}
```

Hàm `socket()` để tạo một socket với các tham số:

- `AF_INET`: Sử dụng IPv4.
- `SOCK_STREAM`: Dùng giao thức TCP (kết nối đáng tin cậy).
- `0`: Sử dụng giao thức mặc định cho `SOCK_STREAM` (TCP).

4.3.2.3. Cấu hình địa chỉ Server

```
serverAddress.sin_family = AF_INET;
serverAddress.sin_port = htons(PORT);
serverAddress.sin_addr.s_addr = inet_addr(IP.c_str());
```

Trong đó có các ý nghĩa như sau:

- `serverAddress.sin_family`: Thiết lập IPv4.
- `serverAddress.sin_port`: Chuyển PORT từ dạng số nguyên sang dạng network byte order (big-endian) bằng `htons`.
- `serverAddress.sin_addr.s_addr`: Chuyển đổi địa chỉ IP từ chuỗi (IP) sang dạng nhị phân sử dụng `inet_addr`.

4.3.2.4. Gán địa chỉ cho Socket bằng hàm `bind()`

```
int configServer = ::bind(serverSocket, (struct sockaddr * ) & serverAddress,
sizeof(serverAddress));
if (configServer < 0) {
    cout << "Cannot config socket with your port and IP!" << endl;
}
```

4.3.2.5. Lắng nghe kết nối của từ Client bằng hàm `Listen()` và chấp nhận bằng hàm `accept()`

```
int status = listen(serverSocket, 5);
if (status < 0) {
    cout << "Cannot listen to Client!" << endl;
} else {
    cout << "Server is ready for listening at port " << PORT << " ....." <<
endl;
```

| |
|---|
| } |
|---|

5. Bảng phân công

5.1. Bảng phân công chức năng và công việc trong hệ thống

| Chức năng / Công việc | Phân công | Cải tiến | Hoàn thiện |
|--|-----------|-------------|------------|
| List App | Phong | Phong, Long | Long, Ngọc |
| Start App | Phong | Ngọc | Ngọc |
| Stop App | Phong | | Phong |
| List Services | Phong | Long | Phong |
| Start Services | Phong | | Phong |
| Stop Services | Phong | | Phong |
| Shutdown | Phong | | Phong |
| Restart | Phong | | Phong |
| Sleep | Phong | Ngọc | Ngọc |
| Tạo tệp/thư mục | Long | | Long |
| Di chuyển tệp/ thư mục | Long | | Long |
| Sao chép tệp/ thư mục | Long | | Long |
| Xoá tệp/ thư mục | Long | | Long |
| Liệt kê tệp/thư mục từ một đường dẫn | Long | | Long |
| Lưu tệp gửi từ Gmail vào Server | Long | | Long |
| Gửi nhiều tệp thư mục từ Server về Gmail | Long | | Long |
| Chụp màn hình một lần/ theo chu kì | Long | | Long |
| Gmail API | Long | | Long |
| Client Socket | Long | | Long |
| Tự động kết nối Server thông qua MAC | Long | | Long |
| Quản lí song song nhiều Server bằng đa luồng | Long | | Long |
| Server Socket | Ngọc | | Ngọc |

| | | | |
|---------------------------------------|------------|-------------|------------|
| Webcam chụp ảnh một lần / theo chu kì | Ngọc | | Ngọc, Long |
| Webcam quay Video | Ngọc | | Ngọc |
| Đa luồng Webcam | Ngọc | | Ngọc, Long |
| Chức năng Keylogger | Phong | Long, Phong | Long |
| Đa luồng Keylogger | Ngọc | | Ngọc, Long |
| Xây dựng hệ thống nội bộ Client | Long | | Long |
| Xây dựng hệ thống nội bộ Server | Ngọc, Long | | Ngọc, Long |
| Hợp nhất toàn bộ mô hình | Long, Ngọc | | Long, Ngọc |

5.2. Bảng phân công viết báo cáo

| Phân công | Người thực hiện |
|------------------------------------|-------------------|
| Soạn sườn báo cáo | Ngọc |
| Chỉnh sửa, căn lề, làm đẹp báo cáo | Phong |
| Soạn nội dung báo cáo | Ngọc, Phong, Long |
| Soạn mẫu báo cáo Request | Long |
| Soạn mẫu báo cáo Response | Ngọc |
| Bảng mã Error_Code | Ngọc |
| Kịch bản video | Ngọc |
| Edit Video | Long |

5.3. Phân công hoạt động nhóm do nhóm trưởng tổ chức

| Công việc | Hình thức | Người Thực hiện/ Tham gia |
|------------------------------------|-----------------------------|---------------------------|
| Phân công công việc | Ứng dụng Trello, Google Doc | Ngọc, Phong, Long |
| Họp nhóm định kì do Ngọc chủ trì | Online Google Meet | Ngọc, Phong, Long |
| Bản phác thảo hệ thống mô hình | Google Doc | Ngọc |
| Soạn biên bản họp nhóm | Google Doc | Ngọc |
| Lưu trữ Source và quản lí mã nguồn | Git, Github | Ngọc, Phong, Long |
| Tổ chức buổi chạy thử | Offline tại trường | Ngọc, Long |

| | | |
|-----------------------------|-----------------------|-------------------|
| Thảo luận về báo cáo | Offline tại trường | Ngọc, Phong, Long |
| Đi quay Video | Offline tại nhà riêng | Ngọc, Phong, Long |

6. Trình bày các mẫu, bảng tra cứu

6.1. Mẫu Request của Client

6.1.1. Chức năng List Apps/ Services

```
{
  "cmd": "listas", // "as" stands for "app/service"
  "params": {
    "type": "app/service",
    "status": "running/stopped/all",
    "sort": "name/id/cpu/status/...", // optional
    "order": "asc/desc", // optional
  }
}
```

6.1.2. Chức năng Start Apps/Services

```
{
  "cmd": "startas",
  "params": {
    "type": "app/service",
    "names": "app/service name"
  }
}
```

6.1.3. Chức năng Stop Apps/Services

```
{
  "cmd": "stopas",
  "params": {
    "type": "app/service",
    "names": "app/service"
  }
}
```

6.1.4. Chức năng Shutdown

```
{
  "cmd": "shutdown",
  "params": {
    "time": "second(s) from now",
  }
}
```

6.1.5. Chức năng Sleep

```
{
  "cmd": "sleep",
  "params": {
    "time": "second(s) from now",
  }
}
```

6.1.6. Chức năng Restart

```
{
  "cmd": "restart",
  "params": {
    "time": "second(s) from now",
  }
}
```

6.1.7. Chức năng Copy File

```
{
  "cmd": "copyff",
  "params": {
    "type": "file/folder",
    "sources": ["source path 1", "source path 2", ...],
    "destination": "destination path",
    "overwrite": "true/false", // optional, default is false
  }
}
```

```
}
```

6.1.8. Chức năng Move File/Folder

```
{  
  "cmd": "copyff",  
  "params": {  
    "type": "file/folder",  
    "sources": ["source path 1", "source path 2", ...],  
    "destination": "destination path",  
    "overwrite": "true/false", // optional, default is false  
  }  
}
```

6.1.9. Chức năng Delete File/Folder

```
{  
  "cmd": "deleteff",  
  "params": {  
    "type": "file/folder",  
    "paths": ["path 1", "path 2", ...],  
  }  
}
```

6.1.10. Chức năng Create File/Folder

```
{  
  "cmd": "createff",  
  "params": {  
    "type": "file/folder",  
    "name": "file/folder's name",  
    "path": "path",  
  }  
}
```

6.1.11. Chức năng List Files/Folders

```
{  
  "cmd": "listff",  
  "params": {  
    "path": "path",  
  }  
}
```

6.1.12. Chức năng Save Files

```
{  
  "cmd": "saveff",  
  "params": {  
    "path": "path",  
  }  
}
```

6.1.13. Chức năng Get Files/Folder

```
{  
  "cmd": "getff",  
  "params": {  
    "paths": ["path1", "path2", "..."]  
  }  
}
```

6.1.14. Chức năng Screenshot

```
{  
  "cmd": "screenshot",  
  "params": {  
    "save_path": "path", // optional, default is current directory  
  }  
}
```

6.1.15. Chụp màn hình sau mỗi X giây. Chức năng có thể bật/tắt.

```
{
```

```

    "cmd": "screenshotitv", // "itv" stands for "interval"
    "params": {
        "action": "start/stop",
        "interval": X,
    }
}

```

6.1.16. Chức năng Keylogger

```

{
    "cmd": "keyloggeritv", // "itv" stands for "interval"
    "params": {
        "action": "start/stop",
        "interval": X, // seconds
        "save_path": "path", // optional, default is current directory
        "exclude_keys": ["key 1", "key 2", ...], // optional
        "exclude_apps": ["app 1", "app 2", ...], // optional
    }
}

```

6.1.17. Chức năng webcam

6.1.17.1. Chụp ảnh từ webcam và gửi về client.

```

{
    "cmd": "webcam",
    "params": {
        "save_path": "path", // optional, default is current directory
    }
}

```

6.1.17.2. Chụp ảnh từ webcam sau mỗi X. Chức năng có thể bật/tắt.

```

{
    "cmd": "webcamitv", // "itv" stands for "interval"
    "params": {
        "action": "start/stop",
        "interval": X,
    }
}

```

```
    "save_path": "path", // optional, default is current directory
  }
}
```

6.2. Mẫu Response của Server

6.2.1. Trả về ngay lập tức kết quả sau và có file đi kèm

```
{
  "cmd": "listas",
  "body": {
    "error_code": 1,
    "bytes": X,
    "filename": Y,
    "path": "path",
  }
}
```

6.2.2. Trả về ngay lập tức kết quả và không có file đi kèm

```
{
  "cmd": "stopas",
  "params": {
    "type": "app",
    "names": "Zalo",
    "status": "succeed/failed",
    "error_code": 1
  }
}
```

6.2.3. Trả về thông báo khi thực hiện bật/tắt chức năng có chu kì

```
{
  "cmd": "webcamitv",
  "body": {
    "status": "started"
  }
}
```

7. Định hướng phát triển trong tương lai

7.1. Những vấn đề chưa giải quyết

7.1.1. Bảng mã Error_Code

Trong thực tế, việc nắm bắt vấn đề khi hệ thống xảy ra lỗi là một điều hết sức quan trọng trong lĩnh vực công nghệ thông tin. Hệ thống không những chỉ hoạt động tốt mà còn phải dễ bảo trì và dễ khắc phục khi sự xảy ra sự cố. Khi một lỗi xảy ra, hệ thống phải nhận ra và ngăn chặn ngay lập tức, tránh trường hợp lỗi toàn hệ thống và hệ thống đột ngột dừng lại, sau đó gửi những thông tin về lỗi đó cho các nhà phát triển. Việc **bảng mã Error_Code** ra đời là để giúp cho các Developer khoanh vùng và đưa ra cách giải quyết phù hợp. Trong đồ án này cũng có triển khai chức năng đó nhưng vẫn còn các mặt hạn chế trong đồ án này:

- Mã lỗi chỉ được in ra màn hình Console ở nội bộ máy Server.
- Bảng mã lỗi chưa chi tiết cho từng chức năng.
- Bảng mã lỗi chỉ sử dụng cho Server.
- Không thể bắt tất cả các lỗi có thể có trong Server.

Khi hệ thống càng lớn, thêm nhiều chức năng thì bảng mã lỗi sẽ càng lớn, càng khó kiểm soát vì có thể sẽ có nhiều khả năng xảy ra lỗi. Điều này cần có thời gian để kiểm thử hệ thống nhưng vì bị giới hạn thời gian nên chỉ làm bảng mã cho các tình huống cơ bản và thường xuyên xảy ra. Ngoài ra bảng mã này chỉ áp dụng ở nội bộ Server, và in ra màn hình Server, chưa có thông báo qua mail cho người dùng vì lí do không đủ thời gian để xây dựng hàm phân tích lỗi ở phía Client.

7.1.2. Xử lý độ trễ giữa các gói tin

Một Server hoạt động trong thực tế sẽ phải xử lý nhiều yêu cầu từ phía của Client. Trong đồ án này, Server cũng có thể xử lý nhiều yêu cầu một lúc nhưng lại không thể trả kết quả cùng một lúc. Đối với các chức năng như **List Apps/Services**, hệ thống xử lý và trả về gần như là không có độ trễ. Nhưng với cách hàm của Webcam, cần phải có lệnh **on/off** của người dùng thì máy Server mới trả kết quả. Nhưng trong thực tế khi ta yêu cầu một đoạn phim dài 10 giây, máy sẽ tự động trả về mà không cần ra lệnh. Vậy thì tại sao lại phải có lệnh **on/off**? Lí do là trong 10 giây quay video, Client có thể gửi lệnh yêu cầu **List Apps** và lúc này Server có thể gửi cùng lúc cả đoạn Video và danh sách các Apps. Client không thể phân biệt gói tin nào là danh sách các App, gói tin nào có video, lúc này sẽ xảy ra độ trễ và gây ra lỗi. Vậy nên chúng em đã thiết kế có lệnh **on/off** cho các chức năng **Webcam**, **Keylogger** và **ScreenShot** nhằm để ra hiệu cho Client và Server đây chính là gói tin của các chức năng này, nhưng vẫn còn các mặt hạn chế sau:

- Dù đã hạn chế nhưng khả năng độ trễ vẫn còn.

- Chưa xử lý được khi xảy ra đùng độ.
 - Chưa tự động, phụ thuộc vào lệnh **on/off** từ phía người dùng
- Khi chế độ **on/off** vẫn còn, phải có lệnh này thì Client mới phân biệt được, chưa thực sự hoàn toàn giống với chức năng mô tả trong thực tế.

7.1.3. Xác thực người dùng

Nhằm tăng cường tính bảo mật của việc truy cập và sử dụng Server từ xa, bên cạnh việc sử dụng Gmail, Client sẽ yêu cầu người dùng phải đăng nhập lần đầu bằng username và password đã đăng kí trước đó

- Đối với password, cơ sở dữ liệu được sử dụng là **MongoDB** vì nó cung cấp tính linh hoạt cao trong việc lưu trữ dữ liệu phức tạp và không có cấu trúc cứng nhắc. Mật khẩu khi đăng kí sẽ được lưu vào cơ sở dữ liệu dưới một chuỗi mã hoá **SHA-256 (password_hash)** và kèm thêm một **giá trị ngẫu nhiên (salt)** đi cùng với nó.
- Đối với quy trình đăng nhập, người dùng sẽ gửi một email bao gồm username và password. Dựa trên username, hệ thống sẽ tìm đến nơi lưu trữ mật khẩu đã được mã hoá tương ứng. Mật khẩu người mà người dùng nhập được kết hợp với salt và băm lại bằng **SHA-256** để so sánh với chuỗi mã hoá đã được lưu. Nếu hai chuỗi trùng khớp, tức là người dùng đăng nhập đúng và được cấp quyền truy cập Server.

Password hashing mang lại nhiều lợi ích quan trọng trong việc bảo vệ thông tin đăng nhập người dùng, đảm bảo tính an toàn và tính toàn vẹn của hệ thống. Phương pháp này giúp chống lại các cuộc tấn công của hacker vào cơ sở dữ liệu nhằm đánh cắp thông tin người dùng. Dù tin tặc có thể lấy được tất cả mật khẩu, nhưng vì chúng đã được băm, kết hợp với **giá trị salt**, nên việc dò ngược (**reverse lookup**) trở nên bất khả thi.

7.2. Giải pháp đề xuất và chức năng tương lai

7.2.1. Giải pháp cho các vấn đề

Đối với **bảng mã Error_Code**: Đề xuất có một hệ thống nhỏ để kiểm thử và phân hoạch hệ thống thành nhiều phần nhỏ hơn để dễ dàng quản lí, có thể thiết kế theo hướng đối tượng khi hệ thống càng lớn.

Đối với xử lý đùng độ các gói tin: Thiết kế đa luồng cho Client, phân chia thành các luồng chuyên dụng cho một chức năng và chỉ để phục vụ gửi nhận cho chức năng đó theo ý tưởng của Circuit Switching.

7.2.2. Chức năng trong tương lai

Trong tương lai, vượt qua khỏi phạm vi của một môn học Mạng Máy Tính, chúng em sẽ có thể phát triển thêm các chức năng như:

Quản lý dữ liệu bên Server: Sử dụng một cơ sở dữ liệu để lưu trữ lịch sử truy vấn hay quản lý các tệp dữ liệu. Khi người dùng cần thực hiện một hành động lặp lại trong quá khứ, Server sẽ lấy kết quả truy vấn cũ và trả về. Cải thiện tốc độ và tiết kiệm tài nguyên.

Kiểm soát lượng gói tin nhận được, xây dựng hàng đợi: Sau này không chỉ là một Client mà có thể nhiều Client cùng gửi yêu cầu, lúc này sẽ cần có một hàng đợi để xử lý từng yêu cầu, tránh bị ùn tắc và quá tải khi xử lý nhiều yêu cầu cùng lúc.

8. Thư viện sử dụng

8.1. Thư viện OpenCV (Open Source Computer Vision Library)

Thư viện OpenCV [1] tập trung vào xử lý hình ảnh và video, hỗ trợ các chức năng như chụp màn hình và webcam.

- OpenCV có thể xử lý hình ảnh được chụp từ màn hình (screenshot) sau khi sử dụng các công cụ hệ thống khác để lấy ảnh màn hình. Thư viện này hỗ trợ thao tác chỉnh sửa, lưu trữ, và hiển thị hình ảnh.
- OpenCV cung cấp giao diện mạnh mẽ để tương tác với webcam. Thư viện này cho phép mở luồng video từ webcam, ghi lại video từ webcam và xử lý các khung hình trong thời gian thực. Ví dụ: nhận diện khuôn mặt, lọc hình ảnh...

8.2. Thư viện cURL (Client URL Library)

Thư viện cURL chủ yếu được sử dụng để thực hiện các yêu cầu HTTP/HTTPS và tương tác với các giao thức mạng.

- cURL có thể tải xuống hoặc tải lên các tệp từ/đến một máy chủ thông qua giao thức FTP, SFTP, HTTP/HTTPS. Trong trường hợp cần tương tác với tệp từ xa (ví dụ: tải bản cập nhật hoặc gửi dữ liệu đến máy chủ), cURL đóng vai trò quan trọng.
- cURL có thể được sử dụng để gửi dữ liệu thu thập từ keylogger[2] (ví dụ: tổ hợp phím) tới một máy chủ từ xa qua HTTP/HTTPS hoặc các giao thức mạng khác.

8.3. Thư viện json (JavaScript Object Notation)

Thư viện JSON [5] (JavaScript Object Notation) được sử dụng để xử lý dữ liệu dưới dạng JSON, một định dạng phổ biến cho việc trao đổi dữ liệu giữa các hệ thống, hỗ trợ trong việc lưu trữ, đọc, hoặc gửi/nhận dữ liệu dưới dạng JSON.

- **Start/Stop/List các Apps:** Thông tin về các ứng dụng đang chạy, trạng thái, hoặc danh sách ứng dụng có thể được lưu hoặc truyền dưới dạng JSON. Ví dụ: trả về danh sách ứng dụng đang chạy trong một định dạng dễ phân tích.
- **Start/Stop/List các Services:** Thông tin về các dịch vụ đang chạy hoặc trạng thái dịch vụ có thể được lưu và gửi đi dưới dạng JSON để dễ dàng quản lý và trao đổi dữ liệu.
- **Shutdown/Restart/Sleep:** JSON có thể được dùng để lưu trữ các thiết lập hoặc lệnh điều khiển liên quan đến trạng thái của hệ thống.
- **Tương tác tệp/thư mục:** JSON có thể được dùng để lưu thông tin về tệp/thư mục, chẳng hạn như cấu trúc cây thư mục, danh sách tệp, kích thước tệp,...

- **Chụp màn hình:** JSON có thể lưu trữ thông tin về thiết lập chụp màn hình (ví dụ: định dạng, độ phân giải, thời gian chụp) hoặc gửi dữ liệu về ảnh chụp qua mạng.
- **Webcam:** JSON có thể được dùng để lưu trữ hoặc gửi cấu hình webcam, như độ phân giải, thời gian quay/chụp, hoặc trạng thái hoạt động.
- **Keylogger:** Dữ liệu ghi nhận từ keylogger (ví dụ: tổ hợp phím, thời gian gõ) có thể được lưu trữ và truyền đi dưới dạng JSON, đảm bảo tính linh hoạt và dễ phân tích.

9. Tài liệu tham khảo

Tham khảo qua các văn bản

- [1] dab0bby, "Streaming OpenCV C++ video," [Online]. Available: <https://stackoverflow.com/questions/35479211/streamign-opencv-c-video-to-the-browser>. [Accessed 27 11 2024].
- [2] A. Gatsoev, "socket-keylogger," [Online]. Available: <https://github.com/1M50RRY/socket-keylogger/blob/master/Keylogger/Keylogger.cpp>. [Accessed 30 11 2024].
- [3] GeeksforGeeks, "C++ Program to Create a File in a Directory," [Online]. Available: <https://www.geeksforgeeks.org/c-program-to-create-a-file/>. [Accessed 12 12 2024].
- [4] GeeksforGeeks, "C++ Program to Get the List of Files in a Directory," [Online]. Available: <https://www.geeksforgeeks.org/cpp-program-to-get-the-list-of-files-in-a-directory/>. [Accessed 11 12 2024].
- [5] N. Lohmann, "json repository," [Online]. Available: <https://github.com/nlohmann/json>. [Accessed 30 10 2024].
- [6] Mailtrap, "Send Mail with Gmail API," [Online]. Available: <https://mailtrap.io/blog/send-emails-with-gmail-api/>. [Accessed 25 11 2024].
- [7] OpenCV, "OpenCV Univeristy," [Online]. Available: <https://opencv.org/>. [Accessed 1 12 2024].
- [8] PDQ, Powershell, [Online]. Available: PDQ.com. [Accessed 30 11 2024].
- [9] Quora, "How do you create a folder in C++?," [Online]. Available: <https://www.quora.com/How-do-you-create-a-folder-in-C>. [Accessed 11 12 2024].

- [10] tshepang, "C++ Program to take a screenshot," [Online]. Available: <https://stackoverflow.com/questions/24985315/c-program-to-take-a-screenshot>. [Accessed 27 11 2024].