**BÁO CÁO PROJECT**

Ứng viên: Phạm Lê Quốc Khánh – <https://github.com/khanh892002/simple-data-transfering-program>

1. **Cấu trúc project:**
   1. **ReceiveData:**

Bên server có các bước xử lý chính gồm:

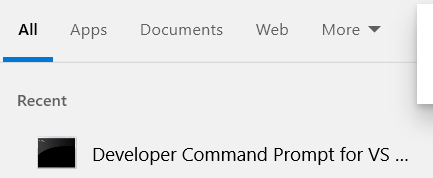
* Xử lý kiểm tra đầu vào hợp lệ khi vừa khởi chạy file. Kiểm tra thư mục lưu trữ có tồn tại hay không.
* Tạo đối tượng lưu thông tin địa chỉ máy và gắn vào một socket là socket chính của server.
* Lắng nghe (listen) và nhận (accept) các kết nối đến.
* Client khi muốn thực hiện một yêu cầu nào sẽ tự động gửi đến trước một chuỗi 4B là FILE, TEXT hoặc EXIT:
  + Nếu chuỗi được gửi đến là FILE thì sẽ đợi nhận tên file từ client gửi qua, nếu gặp lỗi trong quá trình tạo file thì chương trình sẽ gửi lại cho client chuỗi “FAIL” để báo lưu file thất bại. Ngược lại, sẽ gửi chuỗi “SCSS” (viết tắt của SUCCESS) và nhận kích thước file từ client, sau đó bắt đầu nhận nội dung file từ client.
  + Nếu chuỗi nhận được là TEXT, thì chương trình sẽ nhận nội dung tin nhắn mà client gửi qua, tin nhắn sẽ chỉ được giới hạn 511 ký tự, nếu dư sẽ không gửi hết.
  + Nếu chuỗi nhận được là EXIT, thì sẽ thông báo client đóng kết nối và thoát chương trình.
  + Ngược ra thì báo lỗi lệnh gửi đến không hợp lệ.
  1. **SendData:**

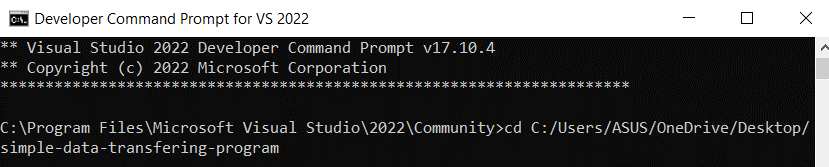
Các bước xử lý chính trên client:

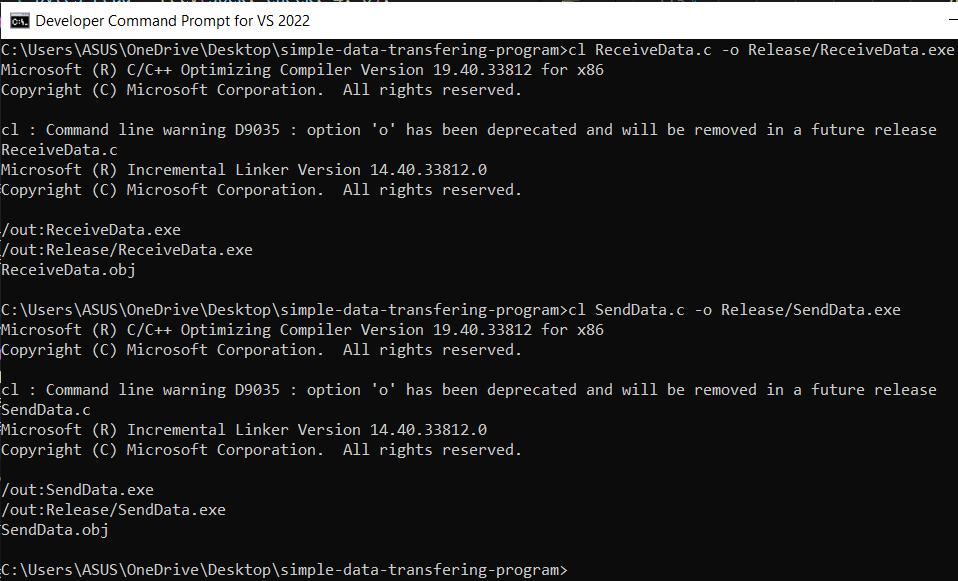
* Xử lý kiểm tra đầu vào hợp lệ khi vừa khởi chạy file.
* Nhận tham số địa chỉ IP của máy server từ đầu vào, tạo một đối tượng lưu thông tin địa chỉ IP, nếu gặp lỗi trong quá trình lưu trữ thì báo lỗi và thoát chương trình. Ngược lại, tạo một socket kết nối đến server lấy thông tin địa chỉ từ đối tượng chứa địa chỉ. Nếu kết nối thất bại thì báo lỗi và thoát chương trình.
* Sau đó, yêu cầu người dùng nhập đúng cú pháp các lệnh, nếu nhập sai sẽ báo lỗi.
* Tạo một mảng ký tự lưu thông tin các lệnh, sẽ chỉ có 3 lệnh chính:
  + SendText: gửi chuỗi TEXT cho server rồi sau đó mới gửi nội dung tin nhắn đến server.
  + SendFile: kiểm tra thông tin đường dẫn file và buffer\_size, kiểm tra cấp phát vùng nhớ buffer để gửi nội dung file, kiểm tra mở file. Nếu thỏa hết điều kiện trên thì gửi chuỗi FILE đến server, xác định và gửi tên file cho server. Đợi nhận chuỗi xác nhận server có thể lưu trữ được file hay không (SCSS/FAIL), nếu được thì xác định và gửi kích thước file, sau đó gửi nội dung file và thông báo sau khi hoàn thành, ngược lại báo lỗi từ server.
  + exit: gửi chuỗi EXIT cho server, thoát vòng lặp, kết thúc chương trình.
  + Ngoài ra: báo lỗi không hợp lệ và yêu cầu người dùng nhập lại đúng cú pháp.

1. **Cách khởi chạy:**

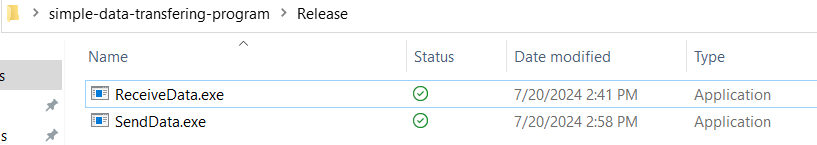
Ta cần biên dịch các file SendData.c và ReceiveData.c để sử dụng trong terminal dưới dạng các lệnh. Để biên dịch, ta sử dụng Developer Command Prompt và chỉ định các file và vị trí lưu kết quả biên dịch:



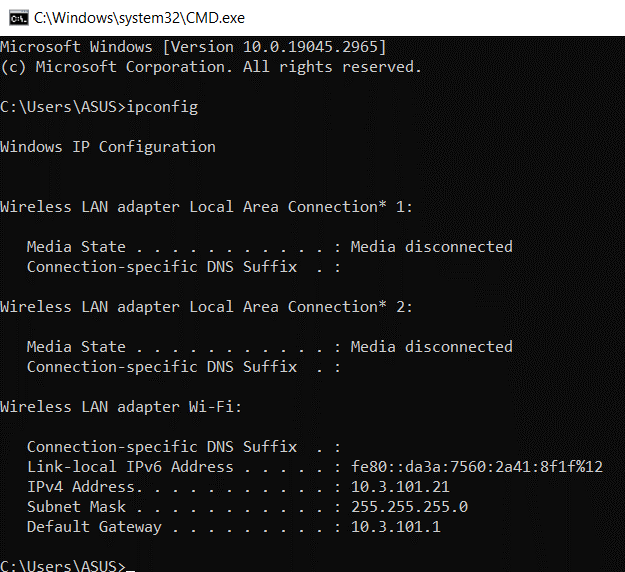


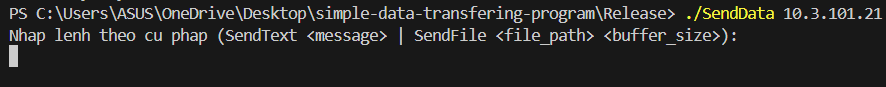


khi này sẽ có 2 file exe được tạo ra trong thư mục Release:

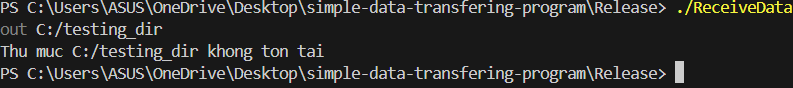


SendData cần chỉ định rõ địa chỉ IP của thiết bị đang chạy file ReceiveData, nếu là cùng một máy thì có thể sử dụng địa chỉ 127.0.0.1:



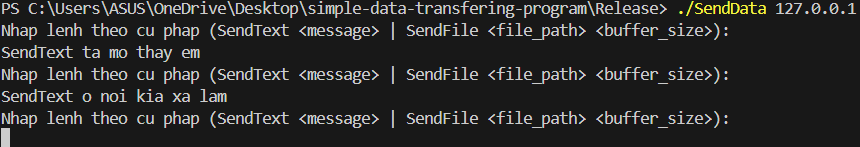


Đối với bên server nếu thư mục được truyền vào không tồn tại thì chương trình sẽ báo lỗi và thoát chương trình:

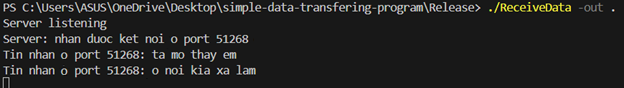




1. **Kết quả:**
   1. **Chức năng SendText:**
      1. Client:

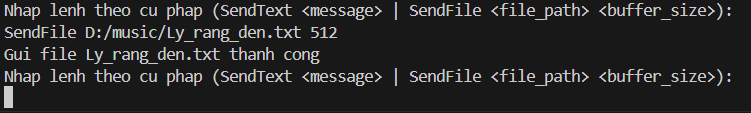
****

* + 1. Server:

****

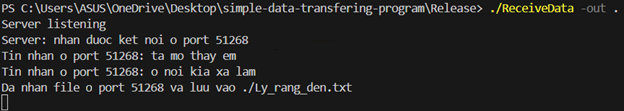
* 1. **Chức năng gửi file SendFile:**
     1. Client:

Sau khi gửi các thông tin của file đến server thì chương trình sẽ nhận phản hồi từ server cho biết file đã được gửi qua hay chưa:

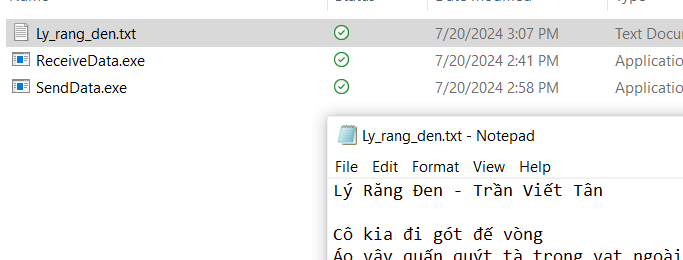
****

* + 1. Server:

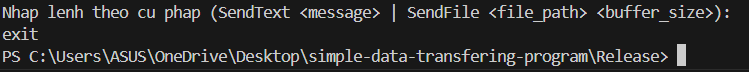
Khi đã lưu thành công file thì chương trình sẽ có thông báo trên server:

****

file sẽ được lưu vào thư mục được gán khi khởi chạy file ReceiveData.exe. (. Là thư mục cha chứa file ReceiveData.exe):

****

* 1. **Thoát (ở SendData):**

****

1. **Những lỗi chưa khắc phục được:**
   1. **Chưa thể xử lý nhiều kết nối đến server:**

Chương trình hiện chỉ có thể nhận kết nối đến từ 1 client và tiếp tục tương tác cho đến khi client gửi yêu cầu kết thúc hoạt ngừng kết nối đột ngột.

Khi xử lý được nhiều client một lúc thì sẽ bỏ đi chức năng gửi EXIT của client và thoát ra của server.

* 1. **Những lỗ hổng có liên quan đến buffer:**
     1. **Buffer Overflow:**

Là một loại lỗ hổng bảo mật nổi tiếng, xảy ra khi chương trình tải dữ liệu vào bộ đệm vượt quá dung lượng lưu trữ được cấp phát, ghi đè ra ngoài vùng đệm gây hỏng chương trình, hỏng dữ liệu, gây ra những tình huống khó lường và thậm chí tạo điều kiện thực thi mã độc.

Lỗ hổng này có thể bị lợi dụng để chèn thêm mã độc vào. Các loại mã độc này sẽ gây ra các vấn đề quá tải vùng đệm trong bảo mật mạng và thêm mã độc vào vùng nhớ nằm ngoài vùng đệm, cho phép tin tặc chạy các chương trình nhằm lấy được quyền quản trị.

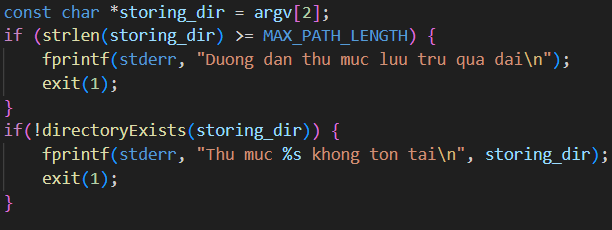
Trong chương trình, nếu socket nhận dữ liệu mà không kiểm tra độ dài dữ liệu, nếu dữ liệu quá lớn sẽ gây tràn bộ nhớ đệm.

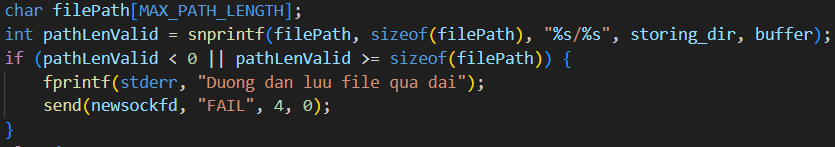
Giải pháp: sử dụng các hàm an toàn để xử lý chuỗi và buffer như strncpy và snprintf thay vì strcpy và sprintf. Các hàm này đều nhận một biến chứa thông tin kích thước tối đa và chỉ ghi dữ liệu không vượt quá kích thước này, đảm bảo tràn bộ nhớ.

Ta sẽ tạo thêm một hằng số MAX\_PATH\_LENGTH để sử dụng trong 2 file c:

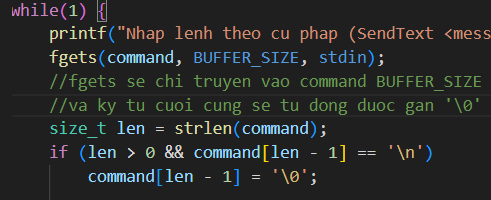
* ReceiveData.c: Các buffer có thể bị overflow gồm có storing\_dir, filePath trích xuất thông tin từ storing\_dir

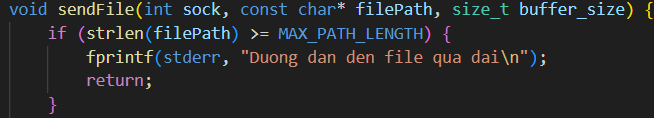
Kiểm tra đường dẫn đến thư mục lưu trữ trên

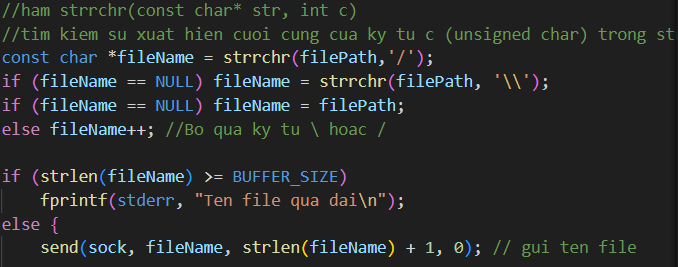




* SendData.c: biến command, filePath và fileName trích xuất thông tin từ filePath







* + 1. **Uncontrolled Format String:**

Lỗ hổng này bao gồm việc chấp nhận kkết quả đầu vào không được kiểm soát hoặc trái phép dưới dạng chuỗi định dạng để thực thi một hàm. Điểm yếu này có thể dẫn đến việc thực thi mã độc và thậm chí làm hỏng hệ thống (VD: SQL Injection).

* + 1. **Integer Overflow:**

Lỗ hổng này tồn tại khi một phép tính cố gắng tăng giá trị số nguyên cao hơn giá trị tối đa trong cách lưu trữ của kiểu dữ liệu. Khi lỗi này xảy ra, giá trị số nguyên thường sẽ quay trở về giá trị nhỏ nhất theo cách lưu trữ. Thông thường điểm yếu này trở thành một vấn đề trong bảo mật khi kết quả tính toán được sử dụng trong vòng lặp điều kiện, xác định kích thước hoặc thực thi các nhiệm vụ như sao chép, cấp phát bộ nhớ, ghép nối…

* + 1. **Denial of Service (DoS):**
  1. **Lỗ hổng tiềm năng:**

Nếu một thiết bị kết nối đến server khi đang chạy, thì có thể gửi đến một chuỗi “FILE”, gửi một tên file và truyền đến nội dung file độc hại vào server.

1. **Tài liệu tham khảo:**

<https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c/>

<https://www.bogotobogo.com/cplusplus/sockets_server_client.php>

<https://quantrimang.com/cong-nghe/lo-hong-bao-mat-nhung-hieu-biet-can-ban-93098>

https://avinetworks.com/glossary/buffer-overflow/