**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:

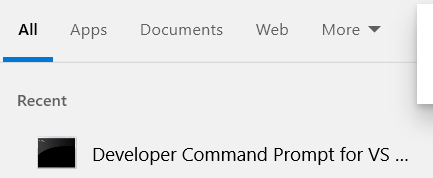
* Xử lý kiểm tra đầu vào hợp lệ khi vừa khởi chạy file. Kiểm tra thư mục lưu trữ có tồn tại hay không, tên có quá dài hay không.
* Tạo đối tượng lưu thông tin địa chỉ máy và gắn vào một socket là socket của server.
* Lắng nghe (listen) và nhận (accept) các kết nối đến.
* Client khi muốn thực hiện một yêu cầu nào sẽ tự động gửi đến server một chuỗi 4 ký tự là FILE, TEXT hoặc EXIT. Nếu chuỗi nhận được là:
  + FILE: 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”. 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.
  + TEXT: 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.
  + 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:**

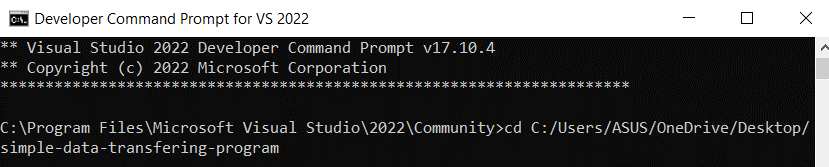
Các bước xử lý chính trên client:

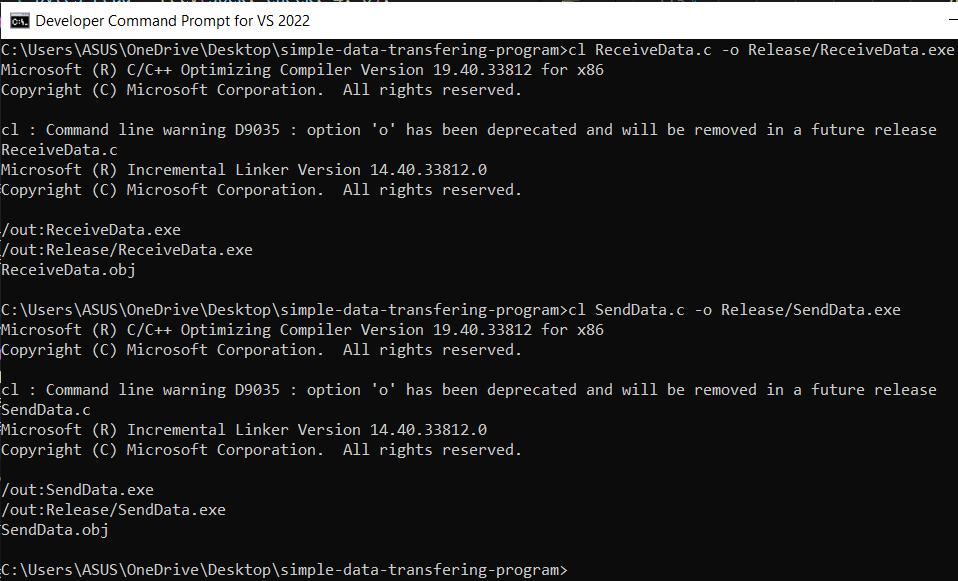
* Xử lý kiểm tra đầu vào hợp lệ khi vừa khởi chạy file.
* Nhận tham số địa chỉ IP của máy server từ đầu vào, tạo một đối tượng lưu thông tin địa chỉ IP, nếu gặp lỗi trong quá trình lưu trữ thì báo lỗi và thoát chương trình. Ngược lại, tạo một socket kết nối đến server lấy thông tin địa chỉ từ đối tượng 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 độ dài đường dẫn, buffer\_size hợp lệ, 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 file (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 xuất ra màn hình 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:**

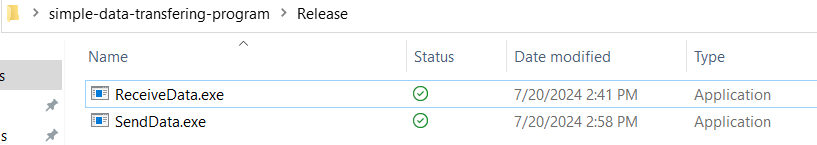
Ta cần biên dịch các file SendData.c và ReceiveData.c để sử dụng trong terminal dưới dạng các lệnh. Để biên dịch, ta sử dụng Developer Command Prompt và chỉ định các file và vị trí lưu kết quả biên dịch:



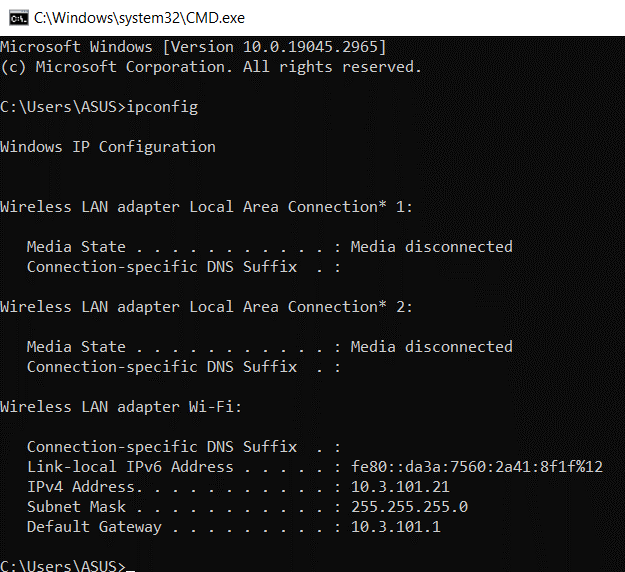


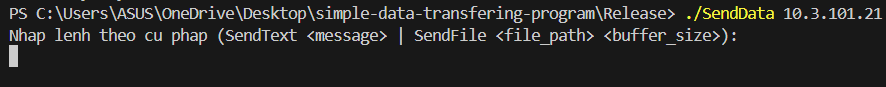


khi này sẽ có 2 file exe được tạo ra trong thư mục Release:

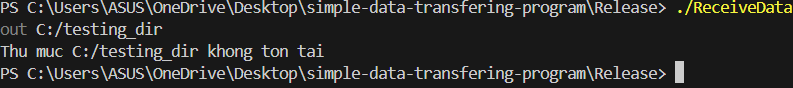


SendData cần chỉ định rõ địa chỉ IP của thiết bị đang chạy file ReceiveData, nếu là cùng một máy thì có thể sử dụng địa chỉ 127.0.0.1:



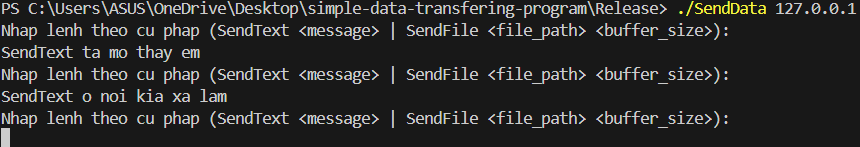


Đối với bên server nếu thư mục được truyền vào không tồn tại thì chương trình sẽ báo lỗi và thoát chương trình:

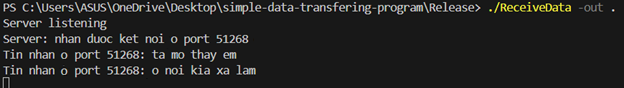




1. **Kết quả:**
   1. **Chức năng SendText:**
      1. Client:

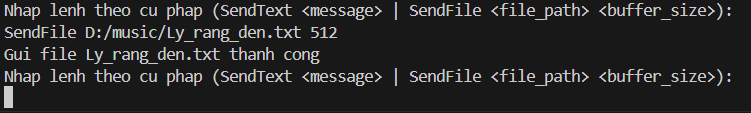
****

* + 1. Server:

****

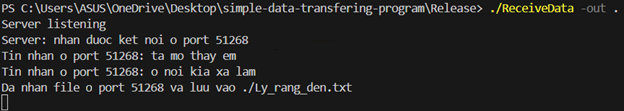
* 1. **Chức năng gửi file SendFile:**
     1. Client:

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:

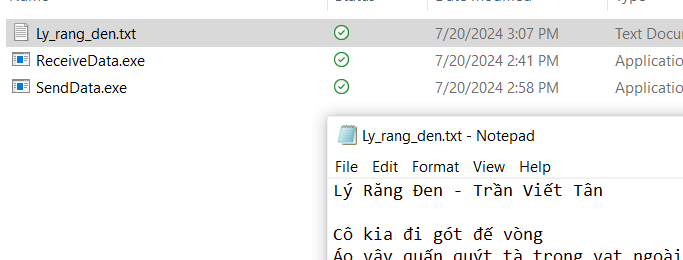
****

* + 1. Server:

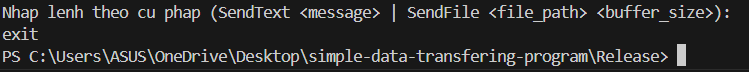
Khi đã lưu thành công file thì chương trình sẽ có thông báo trên server:

****

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:**

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 hệ thống. 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.

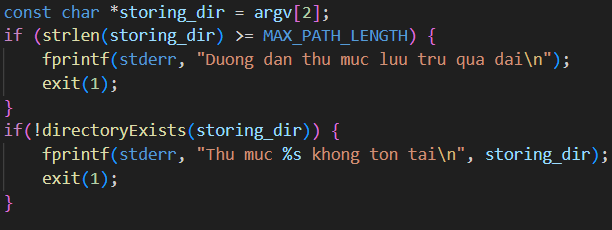
Trong chương trình, nếu socket nhận dữ liệu quá lớn sẽ gây tràn bộ nhớ đệm.

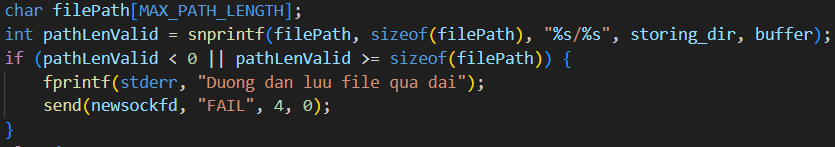
Giải pháp: sử dụng các hàm an toàn để xử lý chuỗi và buffer như strncpy và snprintf thay vì strcpy và sprintf. Các hàm này đều nhận một biến chứa thông tin kích thước tối đa và chỉ ghi dữ liệu không vượt quá kích thước này, đảm bảo tràn bộ nhớ.

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

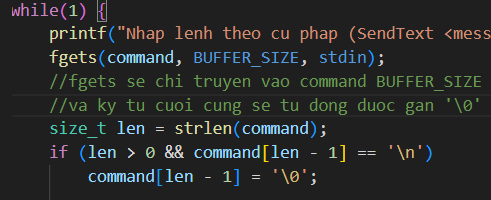
* ReceiveData.c: Các buffer có thể bị overflow gồm có storing\_dir, filePath trích xuất thông tin từ storing\_dir

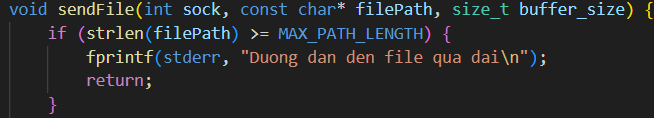
Kiểm tra đường dẫn đến thư mục lưu trữ trên

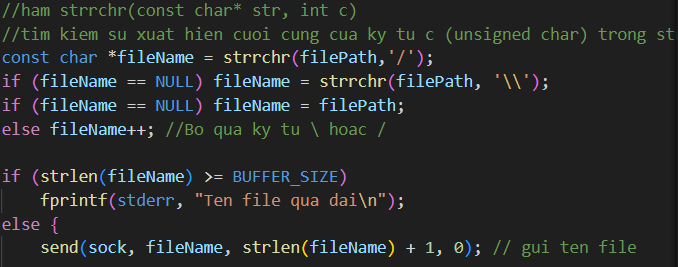




* SendData.c: biến command, filePath và fileName trích xuất thông tin từ filePath





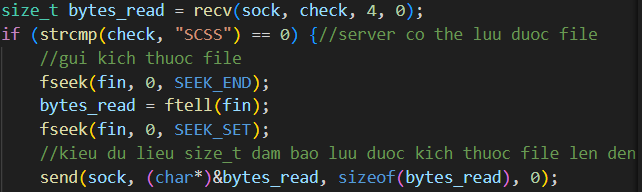


* + 1. **Integer overflow:**

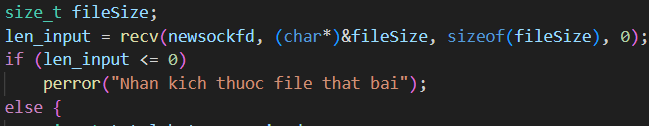
Lỗ hổng này xảy ra khi một phép tính tăng giá trị số nguyên lớn hơn giá trị tối đa mà kiểu dữ liệu có thể lưu trữ, dẫn đến không lưu được kết quả chính xác. Lỗ hổng này thường được tìm thấy xảy ra trong kết quả tính toán được sử dụng trong vòng lặp điều kiện, xác định kích thước hoặc thực thi các nhiệm vụ như sao chép, cấp phát bộ nhớ, ghép nối…

Do chương trình có thể truyền tải file có dung lượng lên đến hàng Gigabyte nên biến fileSize lưu trữ kích thước file sẽ được khai báo với kiểu dữ liệu size\_t (unsigned long long 16Byte):

* SendData.c:



* ReceiveData.c:



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

Lỗ hổng này bao gồm việc nhận dữ liệu đầu vào không được kiểm soát hoặc không hợp lệ dưới dạng chuỗi định dạng (Format string) để thực thi một hàm. Điểm yếu này có thể dẫn đến việc thực thi mã độc và thậm chí làm hỏng hệ thống.

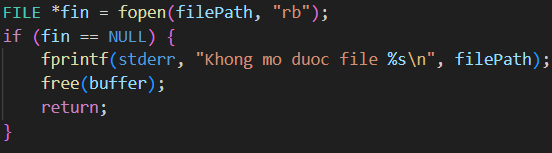
Format string là một tham số được dùng trong các hàm printf, fprintf, snprintf… để xác định cách biểu diễn kết quả của các tham số khác được truyền vào sử dụng các thông số như %s, %d…

Hay như trong hàm system, chỉ cần chuỗi đầu vào có liên quan đến dữ liệu người dùng nhập vào đều cần phải được kiểm tra trước khi truyền vào hàm system để chạy.

Trên lý thuyết, các hàm trên đều không an toàn nếu không sử dụng format string mà chỉ truyền các tham số từ các biến. Nếu các biến được truyền vào là chuỗi thì hàm sẽ nhận các biến đó giống như format string và nếu các chuỗi được truyền vào có các thông số gọi ra các thông tin hệ thống thì sẽ gây rò rỉ thông tin. Từ đó, tin tặc có thể làm rò rỉ dữ liệu, gây crash hệ thống và thậm chí là chạy mã độc.

Giải pháp: Để khác phục thì ta chỉ cần sử dụng thêm format string cho các hàm trên và sử dụng đúng thông số tương ứng cho các tham số mà ta cần xử lý.

File SendData.c có phần xử lý không mở được file để gửi sẽ có xuất thông tin ra màn hình nên cần phải xử lý cẩn thận ở các tham số của hàm fprintf:

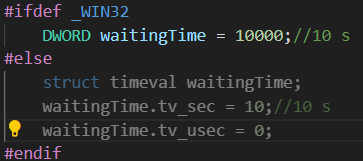


* + 1. **Denial of Service (DoS):**

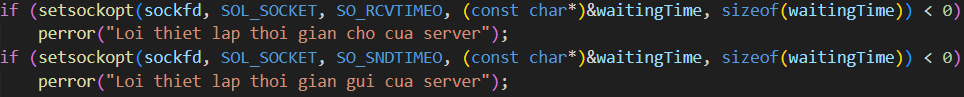
Chủ yếu xảy ra bên server, tấn công DoS xảy ra khi kẻ tấn công làm tiêu tốn tài nguyên của hệ thống, khiến cho hệ thống không thể cung cấp dịch vụ cho người dụng.

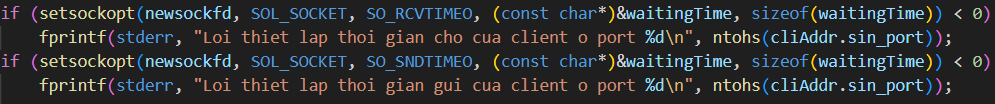
|  |  |  |
| --- | --- | --- |
| Hình thức tấn công | Mô tả | Giải pháp |
| Tấn công băng thông | Tin tặc gửi một lượng lớn dữ liệu đến server, sử dụng hết băng thông và không thể cung cấp dịch vụ cho người dùng. | sử dụng các cơ chế kiểm soát lưu lượng như sử dụng iptables hoặc các giải pháp tường lửa để giới hạn băng thông cho mỗi IP. Sử dụng dịch vụ bảo vệ bên thứ 3 như Cloudflare. |
| Làm đầy bộ nhớ đệm | Nếu không kiểm soát kích thước dữ liệu nhận được, bên tấn công có thể gửi lượng lớn dữ liệu vào bộ nhớ đệm gây tràn bộ đệm. | Sử dụng các hàm snprintf, strncpy… để tránh tràn bộ nhớ đệm, giới hạn kích thước dữ liệu nhận được ở hàm recv thông qua tham số của hàm. |
| Làm đầy bộ nhớ | Tin tặc gửi các yêu cầu hợp lệ nhưng với lượng dữ liệu rất lớn, sử dụng hết bộ nhớ. Chương trình sẽ có thể gặp vấn đề này ở quá trình tải file, cụ thể ở kích thước file. | Giới hạn kích thước tin nhắn/file tải lên, kiểm tra kích thước kết quả nhận được. |
| Request flooding | Tin tặc có thể gửi nhiều yêu cầu liên tiếp gây nghẽn server | Sử dụng netfilter hoặc iptables (Linux) để giới hạn số lượng kết nối từ một IP. |
| Tấn công tài nguyên hệ thống | Tin tặc có thể tạo một lượng lớn kết nối đến server và không gửi dữ liệu hoặc gửi dữ liệu rất chậm, khiến server bị quá tải và từ chối các kết nối hợp lệ. | tăng giới hạn lượng kết nối và sử dụng các kỹ thuật như SO\_RCVTIMEO và SO\_SNDTIMEO để thiết lập thời chờ cho các kết nối. |

Với các thiết lập về SO\_RCVTIMEO và SO\_SNDTIMEO, server có thể tự động ngắt kết nối đến client nếu không có tương tác gì sau 10 giây, ta chỉ cần sử dụng hàm setsockopt để thiết lập 2 tham số trên:

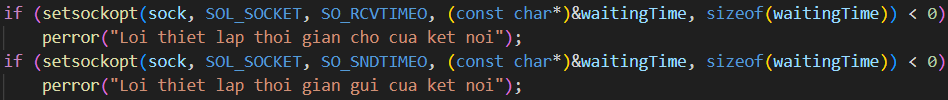


* ReceiveData.c:





* SendData.c:



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/