HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA AN TOÀN THÔNG TIN

MÔN HỌC: PHÂN TÍCH MÃ ĐỘC

BÁO CÁO THỰC HÀNH BÀI 4

Giảng viên: PGS.TS. Đỗ Xuân Chợ

Sinh viên: Hoàng Trung Kiên – B20DCAT098

Hà Nội – 5/2023

Contents

1. Tìm hiểu về IDA.

a, Mục đích.

Mục tiêu của bài thực hành là quan sát sự khác nhau giữa mã nguồn gốc và quá trình disassembly (phân tích ngược) từ dạng nhị phân của nó. Sinh viên sẽ sử dụng phần mềm IDA để phân tích ngược một chương trình C mẫu có tên là sample.c.

Sau khi hoàn thành bài thực hành, sinh viên sẽ có thấy được những khó khăn khi phân tích ngược chương trình về mã nguồn gốc

b, Lý thuyết.

Công cụ IDA (Interactive Disassembler) là một công cụ phân tích ngược mạnh mẽ và phổ biến được sử dụng để dịch mã máy thành mã nguồn gốc hoặc mã hợp ngữ. IDA giúp người dùng nghiên cứu và phân tích các chương trình máy tính, phát hiện lỗ hổng bảo mật, tìm hiểu cấu trúc của chương trình và tìm ra cách thức hoạt động của mã.

Khi sử dụng IDA, bạn có thể thực hiện các hoạt động sau:

1. Dịch mã máy: IDA cho phép bạn nhập mã máy từ tệp nhị phân và dùng công cụ của nó để phân tích mã đó. IDA sẽ phân tích mã máy thành mã hợp ngữ và hiển thị các hàm, biến và cấu trúc dữ liệu trong chương trình.
2. Phân tích lưu đồ luồng điều khiển: IDA cho phép bạn xem và phân tích lưu đồ luồng điều khiển của chương trình. Điều này giúp bạn hiểu cách luồng điều khiển của chương trình được thực hiện và tìm ra các nhánh điều kiện, vòng lặp và điểm nhảy trong mã.
3. Đọc và hiểu mã hợp ngữ: IDA cung cấp các công cụ và tính năng giúp bạn đọc và hiểu mã hợp ngữ. Bạn có thể xem mã hợp ngữ và các mã lệnh cụ thể, xem các biểu đồ dữ liệu và thực hiện các phép biến đổi trên mã.
4. Phân tích cấu trúc dữ liệu: IDA giúp bạn phân tích cấu trúc dữ liệu trong chương trình, bao gồm các biến, cấu trúc dữ liệu, và cách thức chúng được sử dụng và truy cập trong mã.

5. Tìm kiếm lỗ hổng bảo mật: IDA giúp bạn tìm kiếm các lỗ hổng bảo mật trong chương trình bằng cách phân tích mã và tìm ra các điểm yếu tiềm năng, chẳng hạn như các lỗ hổng tràn bộ đệm, lỗi truy cập không hợp lệ và lỗi xử lý dữ liệu không an toàn.

IDA là một công cụ mạnh mẽ được sử dụng rộng rãi trong lĩnh vực phân tích ngược và nghiên cứu bảo mật. Tuy nhiên, nó cũng có thể được sử dụng với mục đích hợp pháp như tìm hiểu cách thức hoạt động của các chương trình và cấu trúc mã máy.

2. Thực hành.

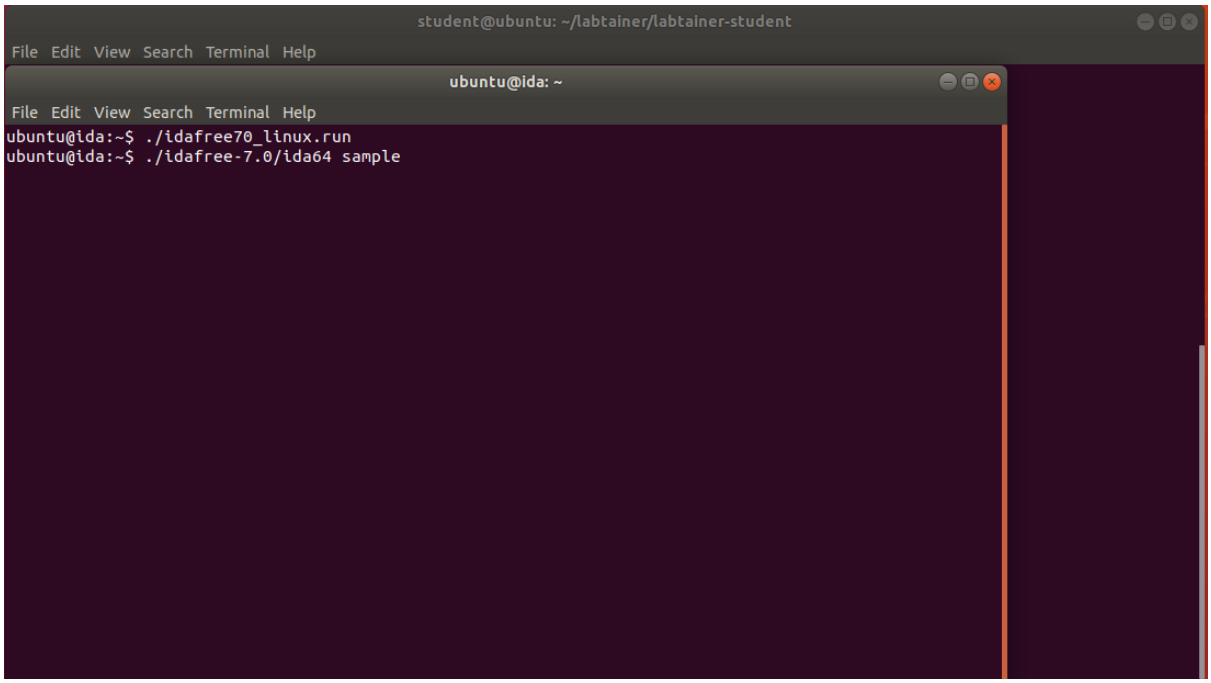
Ở thư mục "home" có chứa file cài đặt của IDA Pro, hãy chạy để cài đặt chương trình

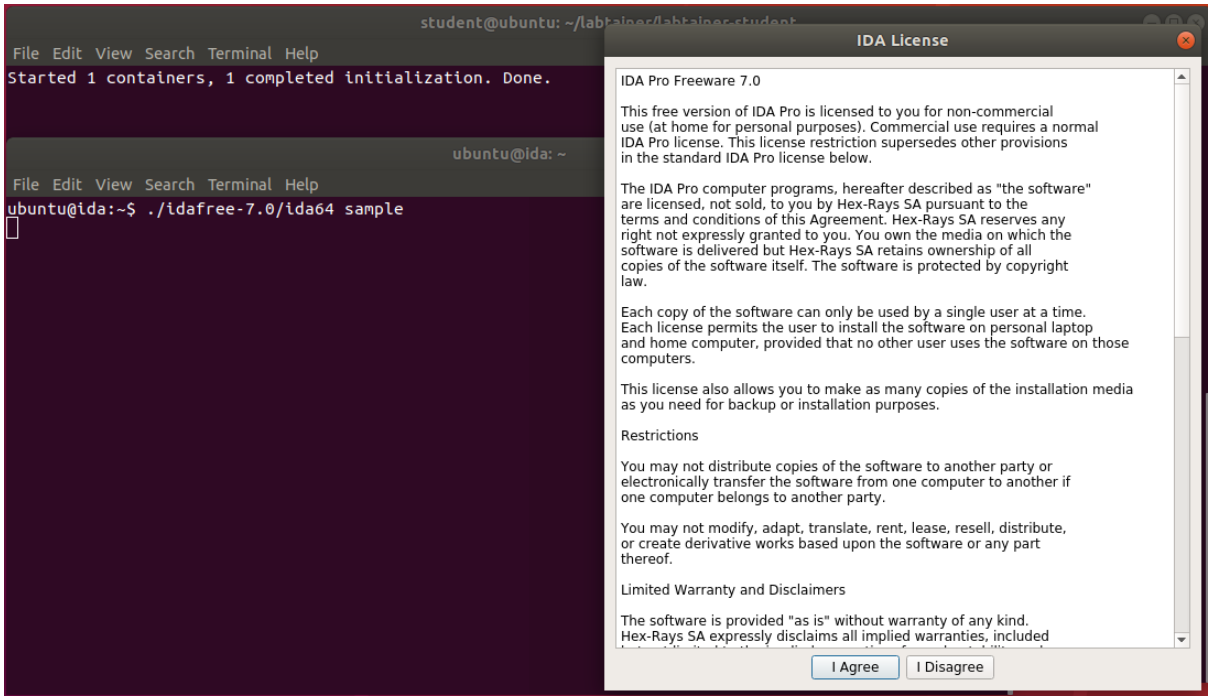
./idafree70_linux.run

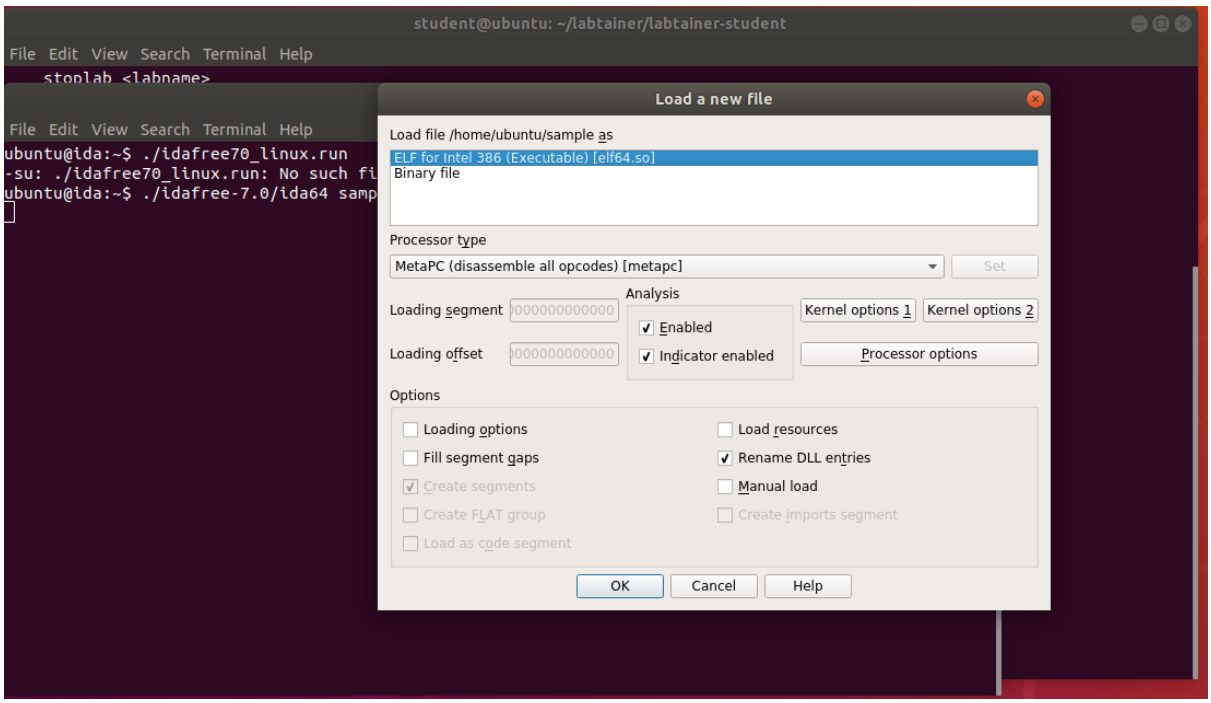
Bước 1: Chạy công cụ IDA để đọc chương trình mẫu

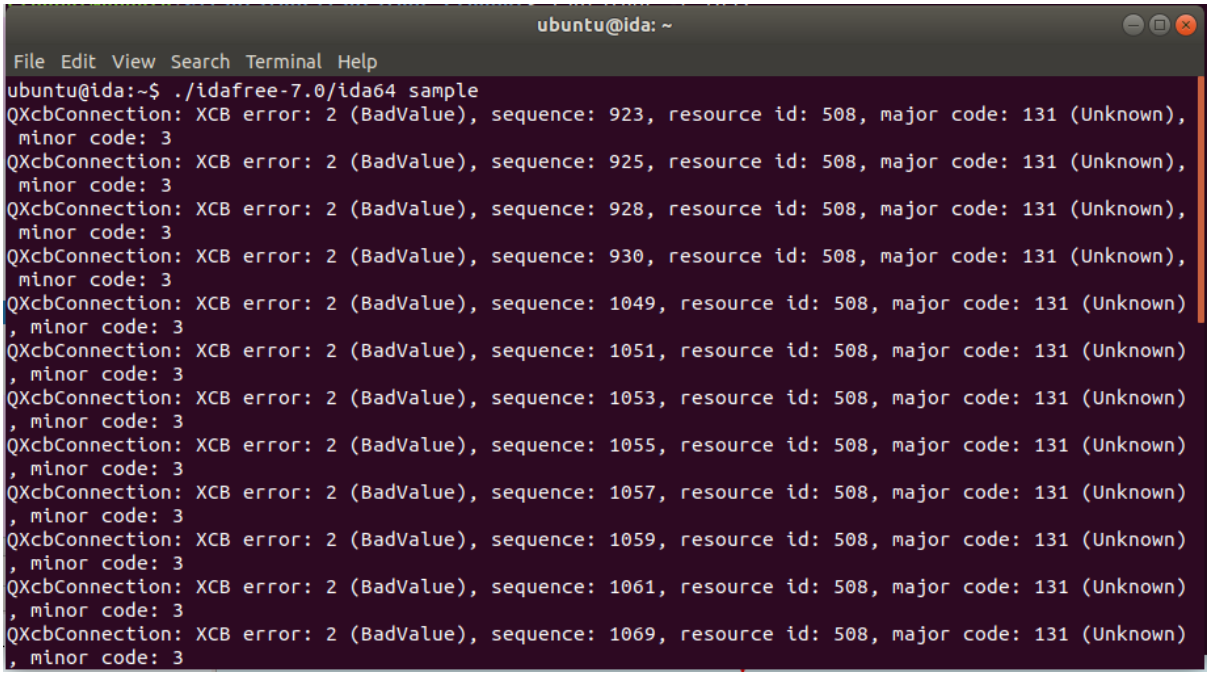
./idafree-7.0/ida64 sample

Bước 2: Trong giao diện của IDA, sinh viên sẽ thấy phiên bản đã được phân tích ngược của chương trình mẫu.




student@ubuntu: ~/labtainer/labtainer-student
File Edit View Search Terminal Help
Started 1 containers, 1 completed initialization. Done.
ubuntu@ida: ~
File Edit View Search Terminal Help
ubuntu@ida:~$ ./idafree-7.0/ida64 sample
IDA License
IDA Pro Freeware 7.0
This free version of IDA Pro is licensed to you for non-commercial use (at home for personal purposes). Commercial use requires a normal IDA Pro license. This license restriction supersedes other provisions in the standard IDA Pro license below.
The IDA Pro computer programs, hereafter described as "the software" are licensed, not sold, to you by Hex-Rays SA pursuant to the terms and conditions of this Agreement. Hex-Rays SA reserves any right not expressly granted to you. You own the media on which the software is delivered but Hex-Rays SA retains ownership of all copies of the software itself. The software is protected by copyright law.
Each copy of the software can only be used by a single user at a time. Each license permits the user to install the software on personal laptop and home computer, provided that no other user uses the software on those computers.
This license also allows you to make as many copies of the installation media as you need for backup or installation purposes.
Restrictions
You may not distribute copies of the software to another party or electronically transfer the software from one computer to another if one computer belongs to another party.
You may not modify, adapt, translate, rent, lease, resell, distribute, or create derivative works based upon the software or any part thereof.
Limited Warranty and Disclaimers
The software is provided "as is" without warranty of any kind. Hex-Rays SA expressly disclaims all implied warranties, included
I Agree
I Disagree


student@ubuntu: ~/labtainer/labtainer-student
File Edit View Search Terminal Help
stoplab <labname>
File Edit View Search Terminal Help
ubuntu@ida:~$ ./idafree70_linux.run
-su: ./idafree70_linux.run: No such fi
ubuntu@ida:~$ ./idafree-7.0/ida64 samp
Load a new file
Load file /home/ubuntu/sample as
ELF for Intel 386 (Executable) [elf64.so]
Binary file
Processor type
MetaPC (disassemble all opcodes) [metapc]
Set
Loading segment
Loading offset
Analysis
Enabled
Indicator enabled
Kernel options 1
Kernel options 2
Processor options
Options
Loading options
Fill segment gaps
Create segments
Create FLAT group
Load as code segment
Load resources
Rename DLL entries
Manual load
Create imports segment
OK
Cancel
Help



Bước 3: so sánh với bản vừa phân tích ngược được với code nguồn sau:

```
#include <stdio.h>
int main(int argc, char * argv[])
{
    char string[100];
    int c = 0, count[26] = {0};
    printf("Enter a string:\n");
   gets(string);
   while ( string[c] != '\0' )
   {
     if ( string[c] >= 'a' && string[c] <= 'z' )
        count[string[c]-'a']++;
     c++;
   }
   for ( c = 0 ; c < 26 ; c++ )
   {
     if ( count[c] != 0 )
        printf("%d %d.\n",c+'a',count[c]);
   }
   return 0;
}
```

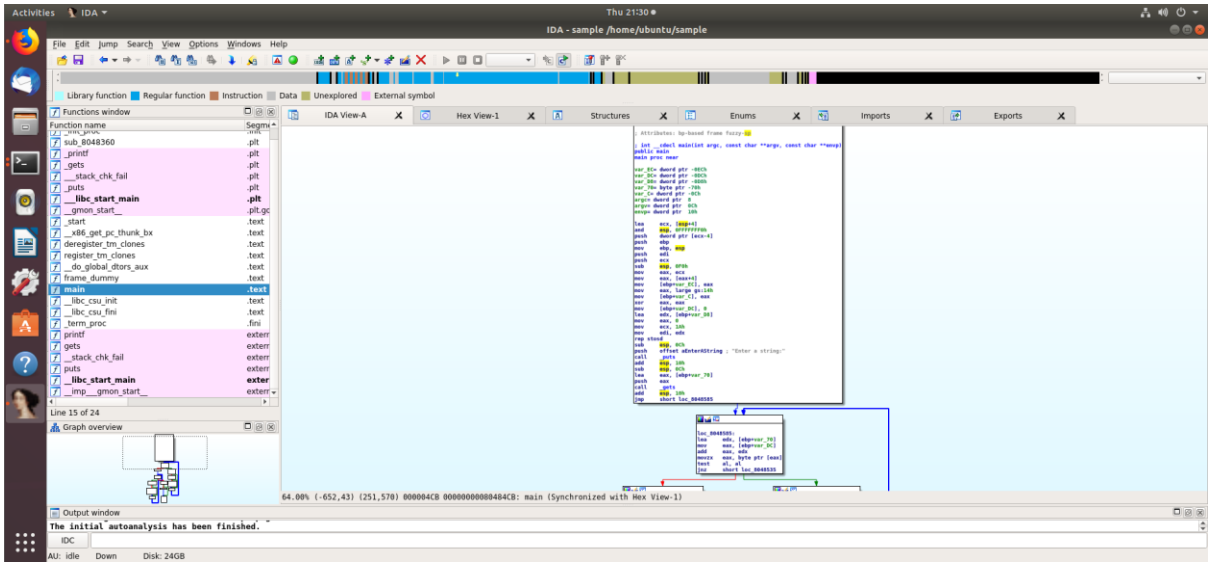
Bản phân tích ngược:

- Chương trình bắt đầu bằng việc khai báo các biến, bao gồm một mảng `string` có kích thước 100 để lưu trữ chuỗi nhập từ người dùng, biến `c` để duyệt qua chuỗi và mảng `count` gồm 26 phần tử để đếm số lần xuất hiện của các ký tự từ 'a' đến 'z'.
- Tiếp theo, chương trình in ra dòng thông báo "Enter a string:".
- Hàm `gets()` được sử dụng để đọc chuỗi nhập từ người dùng và lưu vào biến `string`. Lưu ý rằng hàm `gets()` không an toàn và đã bị loại bỏ từ phiên bản mới nhất của ngôn ngữ C. Để thay thế, bạn có thể sử dụng `fgets()` để đọc chuỗi với độ dài tối đa cho phép.
- Tiếp theo, chương trình sử dụng vòng lặp `while` để duyệt qua từng ký tự trong chuỗi `string`.
- Trong vòng lặp, điều kiện `if` kiểm tra xem ký tự hiện tại có nằm trong khoảng từ 'a' đến 'z' hay không. Nếu đúng, biểu thức `count[string[c]-'a']++` được thực hiện để tăng giá trị tương ứng trong mảng `count`.
- Sau khi duyệt qua toàn bộ chuỗi, chương trình sử dụng vòng lặp `for` để duyệt qua các phần tử trong mảng `count`.
- Trong vòng lặp, điều kiện `if` kiểm tra xem giá trị tương ứng trong mảng `count` có khác 0 hay không. Nếu khác 0, chương trình in ra ký tự tương ứng và số lần xuất hiện của nó.
- Cuối cùng, chương trình trả về giá trị 0 để kết thúc.

So sánh với bản phân tích ngược, bản code nguồn có một số điểm cần lưu ý:

1. Sử dụng hàm `gets()` không an toàn: Hàm `gets()` đã bị loại bỏ từ phiên bản mới nhất của ngôn ngữ C do nguy cơ tràn bộ đệm. Để thay thế, bạn nên sử dụng hàm `fgets()` để đọc chuỗi với độ dài tối đa cho phép và loại bỏ ký tự xuống dòng (`'\n'`) nếu cần thiết.
2. Không xử lý trường hợp chuỗi nhập dài hơn kích thước mảng: Trong bản code nguồn, không có kiểm tra hoặc xử lý nếu chuỗi nhập từngười dùng dài hơn kích thước tối đa của mảng `string`. Điều này có thể dẫn đến tràn bộ nhớ và gây lỗi trong chương trình. Bạn có thể giới hạn độ dài chuỗi đầu vào bằng cách sử dụng hàm `fgets()` và kiểm tra độ dài của chuỗi trước khi xử lý nó.
3. Sử dụng đúng phạm vi ký tự 'a' đến 'z': Bản code nguồn giả định rằng các ký tự trong chuỗi nhập là chữ cái thường từ 'a' đến 'z'. Nếu có các ký tự khác, chương trình sẽ không xử lý chúng. Bạn có thể cải thiện chương trình bằng cách kiểm tra phạm vi ký tự trước khi tăng giá trị tương ứng trong mảng `count`.
4. In ra kết quả không rõ ràng: Cách chương trình in ra kết quả là in ra giá trị số nguyên tương ứng với ký tự và số lần xuất hiện của nó. Điều này có thể làm cho kết

quả khó hiểu. Bạn có thể cải thiện bằng cách in ra ký tự thực tế bằng cách sử dụng `%c` trong hàm `printf()`.



Bước 4. Đề xuất ít nhất hai sửa đổi cho mã nguồn gốc có thể được sử dụng để bảo vệ mã đó khỏi bị phân tích ngược khi sử dụng IDA.

Để tránh bị phân tích ngược bằng IDA hoặc các công cụ phân tích ngược khác, bạn có thể áp dụng một số biện pháp bảo mật vào mã nguồn. Dưới đây là một số gợi ý sửa đổi cho mã nguồn của bạn:

1. Sử dụng mã hóa chuỗi nhập:

- Thay vì sử dụng hàm `gets` để đọc chuỗi nhập từ người dùng, bạn có thể mã hóa chuỗi nhập để làm cho nó khó hiểu hơn. Ví dụ: bạn có thể sử dụng một thuật toán mã hóa như XOR để mã hóa chuỗi nhập và sau đó giải mã nó trong chương trình.
- Điều này làm cho việc phân tích ngược khó khăn hơn vì người tấn công không thể dễ dàng đọc được chuỗi nhập từ mã nguồn.

2. Sử dụng các phép biến đổi phức tạp hơn:

- Thay vì sử dụng phép biến đổi đơn giản như `count[string[c]-'a']++`, bạn có thể áp dụng các phép biến đổi phức tạp hơn để làm cho mã nguồn khó hiểu hơn. Ví dụ: bạn có thể sử dụng các phép biến đổi như phép XOR, phép dịch bit, hoặc phép toán logic phức tạp hơn để tính toán giá trị `count`.

3. Tạo các điểm nhảy ngẫu nhiên:

- Thay vì sử dụng vòng lặp `for` để duyệt qua các phần tử trong mảng `count`, bạn có thể tạo các điểm nhảy ngẫu nhiên trong mã nguồn để làm cho luồng điều khiển trở nên khó đoán.
- Ví dụ: bạn có thể sử dụng một mảng các số nguyên ngẫu nhiên và sử dụng chúng làm chỉ số cho mảng `count`, thay vì sử dụng các chỉ số từ 0 đến 25 theo thứ tự tăng dần.

4. Gộp mã nguồn:

- Bạn có thể gộp mã nguồn thành các hàm nhỏ hơn và sử dụng các hàm con phức tạp hơn để thực hiện các phép toán.
- Điều này làm cho mã nguồn trở nên phức tạp hơn và khó hiểu hơn khi phân tích ngược.

5. Sử dụng mã máy tùy chỉnh:

- Thay vì viết mã bằng C, bạn có thể viết mã trực tiếp bằng mã máy tùy chỉnh.
- Mã máy tùy chỉnh khó hiểu hơn và yêu cầu kiến thức cao hơn để phân tích ngược.

3. Checkwork.

```
student@ubuntu:~/labtainer/labtainer-student$ checkwork
Results stored in directory: /home/student/labtainer_xfer/ida2
Labname ida2

Student              |
==================== |
B20DCAT098           |
What is automatically assessed for this lab:
```