HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÁO CÁO BÀI THỰC HÀNH

Kiểm thử xâm nhập

Lỗi vượt giới hạn cấu trúc

Giảng viên: Đinh Trường Duy

Nhóm lớp: 02

Sinh viên: Hoàng Trung Kiên

Mã sinh viên: B20DCAT098

Hà Nội – 2024

Mục lục

1. **Mục đích**.
Minh họa việc chạy quá giới hạn dự định của cấu trúc dữ liệu trong một chương trình C.

2. **Yêu cầu đối với sinh viên**.
Có kinh nghiệm lập trình C cơ bản và có kiến thức về các cấu trúc dữ liệu đơn giản.

Biết cách biên dịch và thực thi một chương trình C bất kỳ.

Biết cách sử dụng chương trình gỡ lỗi GDB.

3. **Nội dung thực hành**.
Khởi động lab:

Chạy lệnh: labtainer -r overrun trong terminal của Labtainer

```
student@ubuntu:~/labtainer/labtainer-student$ labtainer -r overrun
latest: Pulling from labtainers/overrun.overrun.student
de3b12789954: Pull complete
a92cf332ea78: Pull complete
337c643669d0: Pull complete
dc0dde907a21: Pull complete
8f8a581f8062: Pull complete
90ca10620961: Pull complete
594f4e4aa64b: Pull complete
2d31e52f20a5: Pull complete
7e08acd5edf3: Pull complete
0e08b374be84: Pull complete
Digest: sha256:364b466a8ba0ac7f4641ab6c5cc923587558dda48198269f379345eb6cec1cde
Status: Downloaded newer image for labtainers/overrun.overrun.student:latest
non-network local connections being added to access control list

Please enter your e-mail address:  [B20DCAT098]
```

Kiểm tra lại code: less mystuff.c

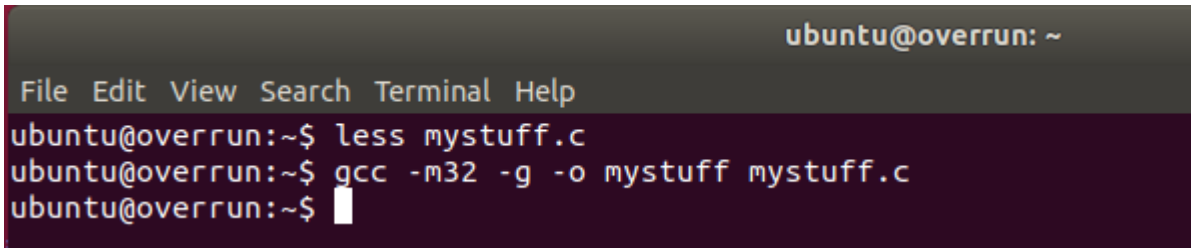
```
ubuntu@overrun: ~
File  Edit  View  Search  Terminal  Help
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
/*
 *  Program to illustrate data references that overrun intended bounds.
 *  Compile this with gcc -m32 -g -o mystuff mystuff.c
 */
/*
 * Structure for holding my information.
 */
struct myData{
     char public_info[20]; // publicly available stuff
     char fav_color[7];
     int pin;  // my pin
     int age;  // my age
};
/*
 * Initialize my information values.
 */
void setData(struct myData *data){
        strcpy(data->public_info, "I yam what I yam.");
        strcpy(data->fav_color, "red");
        data->pin = 99;
        data->age = 61;
:
```

Cấu trúc myData: Nhìn vào struct myData. Trong chương trình khai báo biến my_data là một struct kiểu myData. Lưu ý rằng mảng ký tự public_info có 20 phần tử. Ta có thể tham chiếu đến các phần tử của mảng bằng cách sử dụng chỉ mục. Ví dụ: my data.public_info[4] đề cập đến ký tự thứ 5 trong mảng và my data.public_info[19] đề cập đến ký tự cuối cùng trong mảng.

Địa chỉ của các trường: Sau khi chương trình khởi tạo biến my_data kiểu struct, nó sẽ hiển thị địa chỉ của phần bắt đầu trường public_data và trường pin, đồng thời nó hiển thị các giá trị bộ nhớ của các trường đó

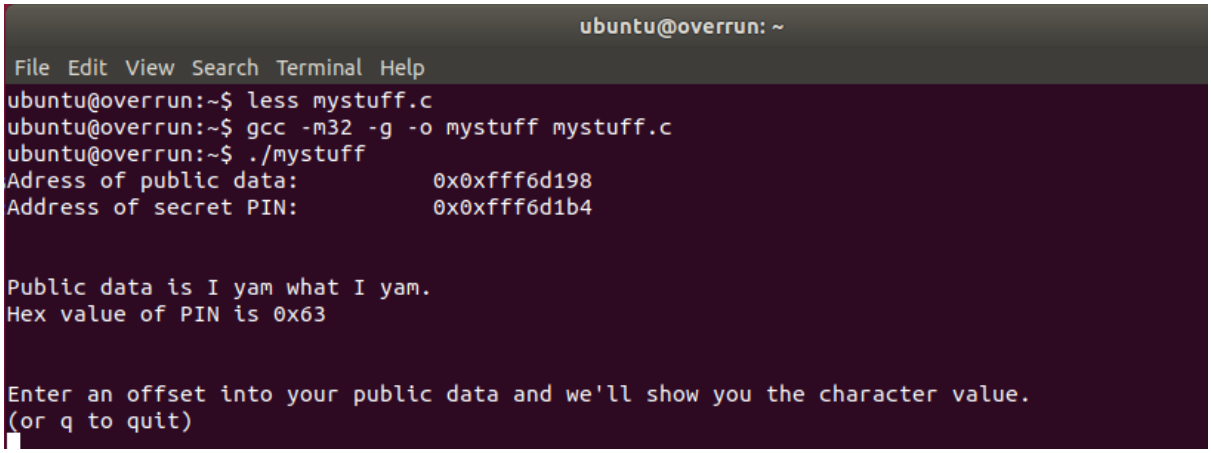
Nội dung bộ nhớ: Chương trình có một vòng lặp cho phép người dùng xem các giá trị hex của các ký tự riêng lẻ trong trường public_info. Chính vòng lặp này sẽ cho chúng ta khám phá câu hỏi được hỏi trước đó: my_data.public_info[20] đề cập đến điều gì?

Biên dịch và chạy chương trình: gcc -m32 -g -o mystuff mystuff.c



Lưu ý rằng -m32 tạo ra một mã nhị phân 32 bit và -g sẽ chứa các ký hiệu trong file nhị phân, cho phép khám phá quá trình thực thi của chương trình bằng cách sử dụng gdb.

Chạy chương trình: ./mystuff



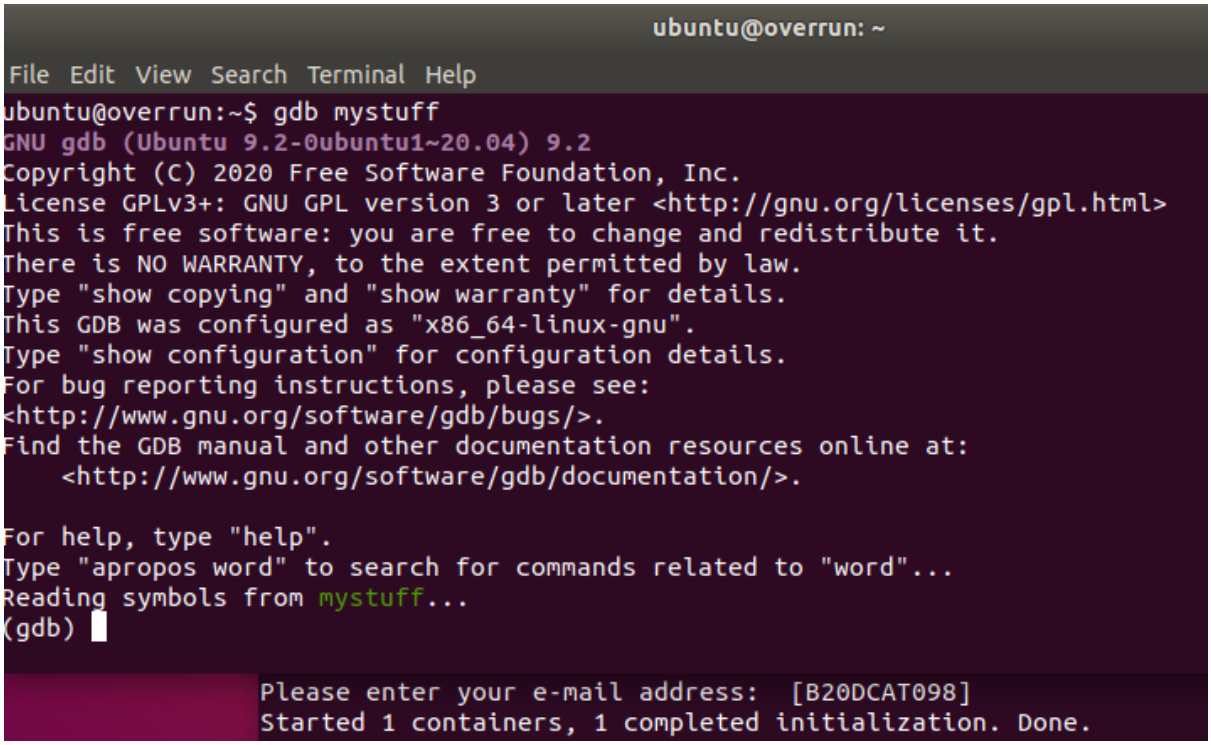
xem các giá trị được hiển thị ở các offset khác nhau trong (và hơn thế nữa) trường public_info. Lưu ý địa chỉ hiển thị của trường public_info và địa chỉ của trường pin. Có bao nhiêu byte phân tách hai trường public_info và pin?

```
0
0
Hex value at offset 0 (address 0x0xfff6d160) is 0x49
Enter an offset into your public data and we'll show you the character value.
(or q to quit)
1
1
Hex value at offset 1 (address 0x0xfff6d161) is 0x20
Enter an offset into your public data and we'll show you the character value.
(or q to quit)
2
2
Hex value at offset 2 (address 0x0xfff6d162) is 0x79
Enter an offset into your public data and we'll show you the character value.
(or q to quit)
```

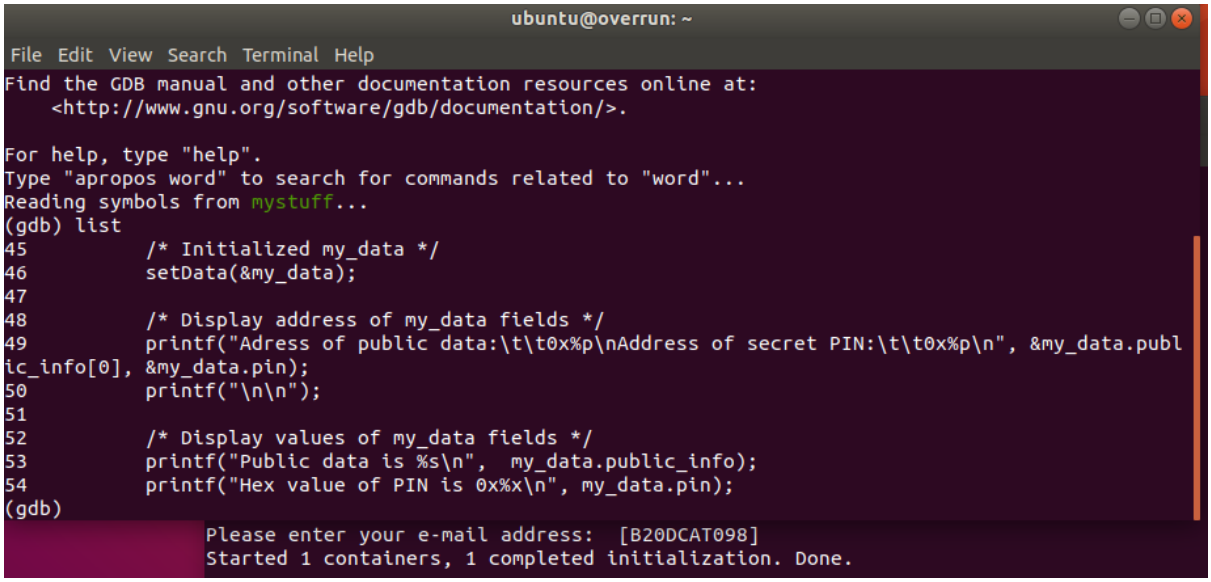
0x0xfffcb7f4 - 0x0xfffcb7c4 = 0x20 = 32 (byte)

Sử dụng chương trình để hiển thị giá trị hex của trường pin. Lưu ý rằng kích thước bộ đệm biến fav_color là số lẻ thì trình biên dịch sẽ đệm bộ đệm để biến tiếp theo bắt đầu trên ranh giới từ 4 byte

Chạy chương trình trong trình gỡ lỗi GDB: gdb mystuff

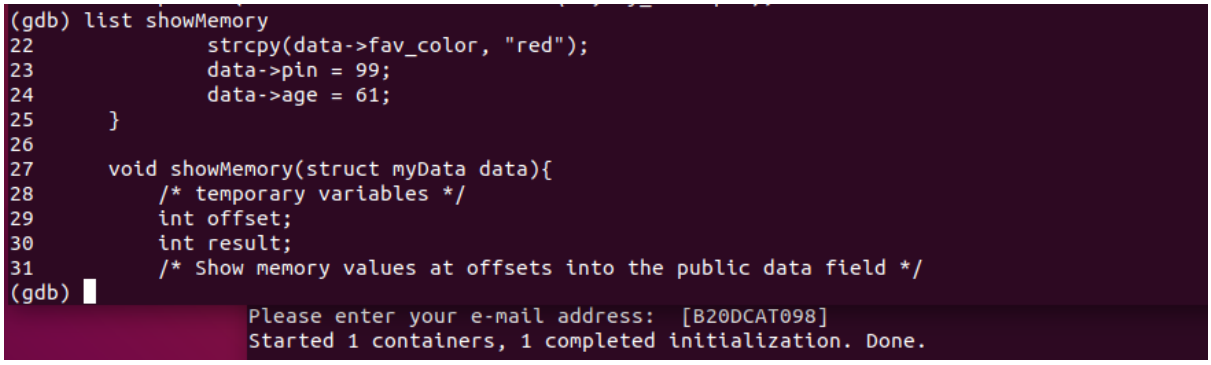


Sử dụng lệnh list để xem mã nguồn.

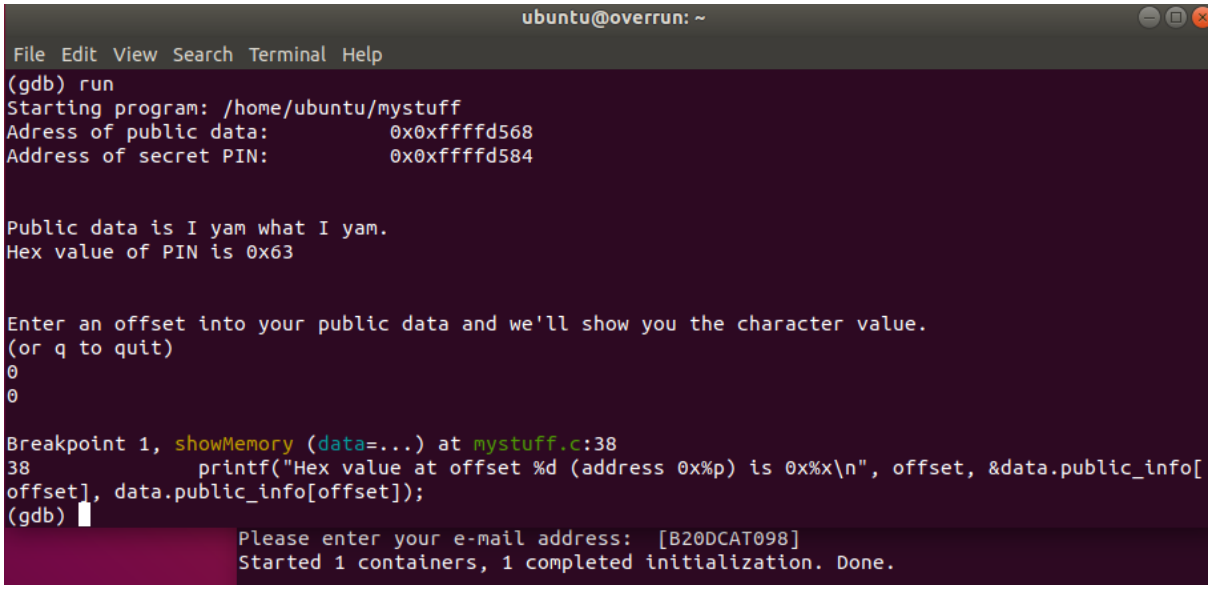


Đặt một điểm ngắt trong hàm showMemory trên dòng in giá trị tại offset đã cho: break <dòng>

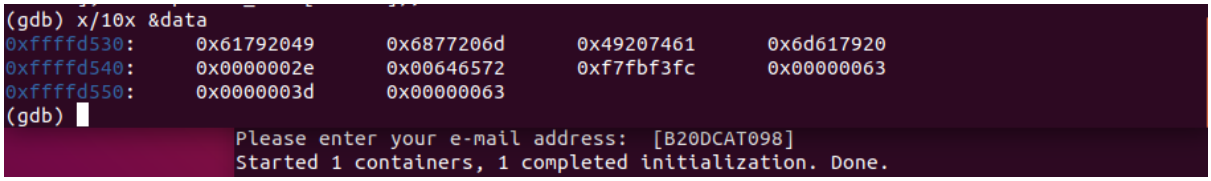
Sử dụng list showMemory để xem mã nguồn cho hàm đó



Break 38 và sau đó chạy chương trình từ bên trong gdb: run



Khi chương trình chạm điểm ngắt, hiển thị 10 word (40 byte) trong bộ nhớ hệ thống dưới dạng giá trị hex bắt đầu từ cấu trúc dữ liệu: x/10x &data



Nội dung bộ nhớ có tương ứng với những gì sinh viên đã quan sát trong khi chạy chương trình không?: có

Đặt một điểm ngắt ở cuối hàm handleMyStuff, tức là trên dòng của dấu ngoặc nhọn cuối cùng bên phải (}) trong hàm đó. Sau đó tiếp tục với lệnh c. Tại lời nhắc cho offset tiếp theo, hãy nhập q. Sau đó, khi chương trình chạm điểm ngắt, hãy hiển thị chương trình đã dịch ngược bằng cách sử dụng: display /i $pc stepi

```
(gdb) list handleMyStuff
36                      break;
37              }
38              printf("Hex value at offset %d (address 0x%p) is 0x%x\n", offset, &data.public_info[
offset], data.public_info[offset]);
39          }
40      }
41      void handleMyStuff(){
42          /* Declare myData variable my_data */
43          struct myData my_data;
44
45          /* Initialized my_data */
(gdb) list
46          setData(&my_data);
47
48          /* Display address of my_data fields */
49          printf("Adress of public data:\t\t0x%p\nAddress of secret PIN:\t\t0x%p\n", &my_data.publ
ic_info[0], &my_data.pin);
50          printf("\n\n");
51
52          /* Display values of my_data fields */
53          printf("Public data is %s\n",  my_data.public_info);
54          printf("Hex value of PIN is 0x%x\n", my_data.pin);
55          printf("\n\n");
(gdb) list
56          showMemory(my_data);
57      }
58      int main(int argc, char *argv[])
59      {
60          handleMyStuff();
61          printf("\nBye\n");
62      }
(gdb)
```

Break 57

```
(gdb) break 57
Breakpoint 2 at 0x565563d9: file mystuff.c, line 57.
(gdb)
```

```
(gdb) c
Continuing.
Hex value at offset 0 (address 0x0xffffd530) is 0x49
Enter an offset into your public data and we'll show you the character value.
(or q to quit)
q
q

Breakpoint 2, handleMyStuff () at mystuff.c:57
57      }
(gdb)
```

```
57      }
(gdb) display /i  $pc
1: x/i $pc
=> 0x565563d9 <handleMyStuff+192>:      nop
(gdb)
```

```
(gdb) stepi
0x565563da      57      }
1: x/i $pc
=> 0x565563da <handleMyStuff+193>:      mov     -0xc(%ebp),%eax
(gdb)
```

Và từng bước để dịch ngược phần còn lại của hàm *handleMyStuff* bằng cách nhấn liên tục phím *Enter* cho đến khi chương trình chuyển sang lệnh *ret*.

```
(gdb) display /i  $pc
1: x/i $pc
=> 0x565563d9 <handleMyStuff+192>:      nop
(gdb) stepi
0x565563da      57      }
1: x/i $pc
=> 0x565563da <handleMyStuff+193>:      mov    -0xc(%ebp),%eax
(gdb)
0x565563dd      57      }
1: x/i $pc
=> 0x565563dd <handleMyStuff+196>:      xor    %gs:0x14,%eax
(gdb)
0x565563e4      57      }
1: x/i $pc
=> 0x565563e4 <handleMyStuff+203>:      je     0x565563eb <handleMyStuff+210>
(gdb)
0x565563eb      57      }
1: x/i $pc
=> 0x565563eb <handleMyStuff+210>:      mov    -0x4(%ebp),%ebx
(gdb)
0x565563ee      57      }
1: x/i $pc
=> 0x565563ee <handleMyStuff+213>:      leave
(gdb)
0x565563ef      57      }
1: x/i $pc
=> 0x565563ef <handleMyStuff+214>:      ret
(gdb)
main (argc=1, argv=0xffffd654) at mystuff.c:61
61          printf("\nBye\n");
```

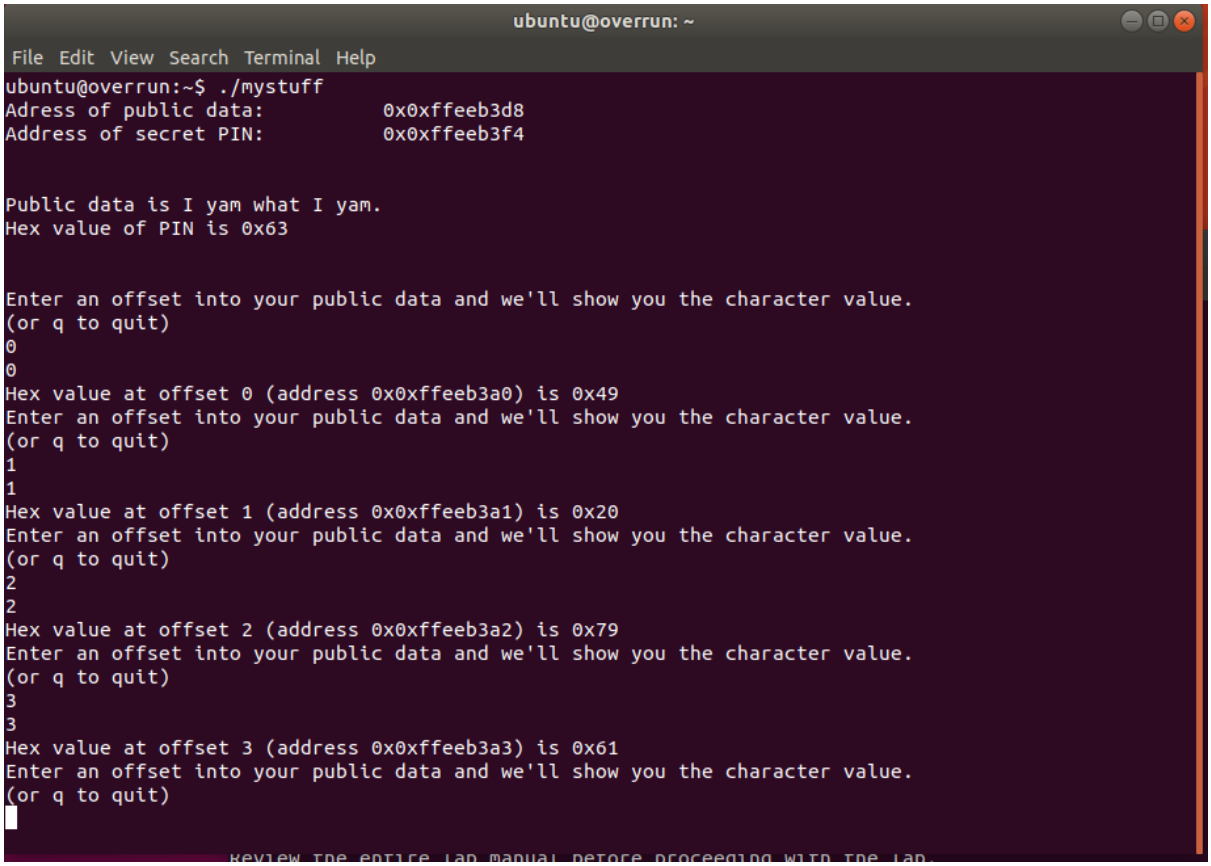
Đây là điểm trong chương trình mà tại đó hàm handleMyStuff sẽ trở lại hàm chính. Lệnh ret chỉ đạo bộ xử lý chuyển đến lệnh tại địa chỉ chứa trong con trỏ ngăn xếp hiện tại. Hiển thị nội dung bộ nhớ được trỏ đến bởi thanh ghi ngăn xếp bằng cách sử dụng:x $esp

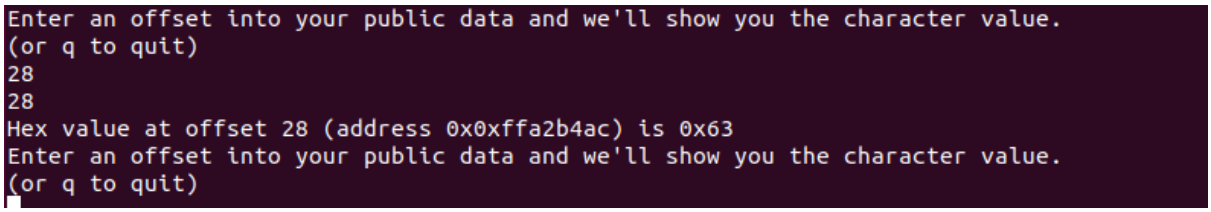
```
1: x/i $pc
=> 0x565563ef <handleMyStuff+214>:      ret
(gdb) x $esp
0xffffd59c:     0x56556413
(gdb)
```

Giá trị được hiển thị sẽ trở thành địa chỉ lệnh tiếp theo, ta có thể xác nhận bằng một nexti nữa. Ghi lại con trỏ lệnh hiện tại. Hãy xem lại địa chỉ ngăn xếp chứa giá trị trả về này. Lưu ý rằng nó cao hơn địa chỉ của cấu trúc dữ liệu được quan sát trong hàm showMemory. Tính toán và ghi lại sự khác biệt giữa hai địa chỉ.

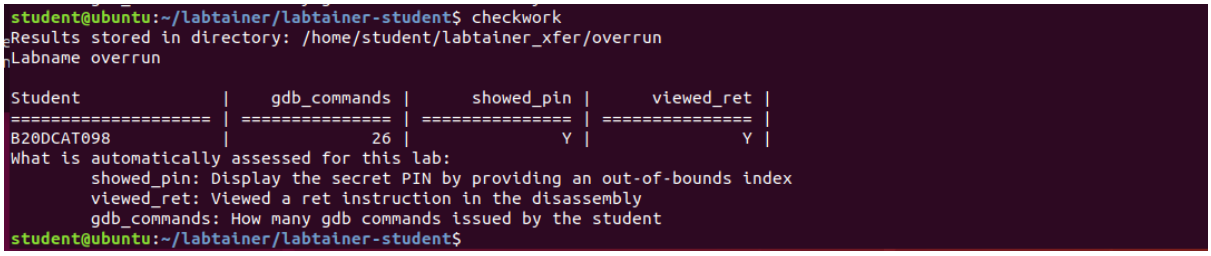
```
(gdb) x $esp
0xffffd59c:     0x56556413
(gdb) nexti
main (argc=1, argv=0xffffd654) at mystuff.c:61
61          printf("\nBye\n");
1: x/i $pc
=> 0x56556413 <main+35>:        sub    $0xc,%esp
(gdb)
                Please enter your e-mail address:  [B20DCAT098]
                Started 1 containers, 1 completed initialization. Done.
```

Chạy lại chương trình bên ngoài trình gỡ lỗi và sử dụng nó để hiển thị giá trị địa chỉ trả về, mỗi lần một byte. Xác nhận rằng địa chỉ là những gì sinh viên đã quan sát thấy trong gdb. Tưởng tượng rằng chương trình cho phép chúng ta sửa đổi các mục riêng lẻ trong mảng public_info. Khi chương trình truy cập vào lệnh ret đã xem trong gdb, nó sẽ quay trở lại địa chỉ đã viết.


ubuntu@overrun: ~
File Edit View Search Terminal Help
ubuntu@overrun:~$ ./mystuff
Adress of public data: 0x0xffeeb3d8
Address of secret PIN: 0x0xffeeb3f4
Public data is I yam what I yam.
Hex value of PIN is 0x63
Enter an offset into your public data and we'll show you the character value.
(or q to quit)
0
0
Hex value at offset 0 (address 0x0xffeeb3a0) is 0x49
Enter an offset into your public data and we'll show you the character value.
(or q to quit)
1
1
Hex value at offset 1 (address 0x0xffeeb3a1) is 0x20
Enter an offset into your public data and we'll show you the character value.
(or q to quit)
2
2
Hex value at offset 2 (address 0x0xffeeb3a2) is 0x79
Enter an offset into your public data and we'll show you the character value.
(or q to quit)
3
3
Hex value at offset 3 (address 0x0xffeeb3a3) is 0x61
Enter an offset into your public data and we'll show you the character value.
(or q to quit)


Enter an offset into your public data and we'll show you the character value.
(or q to quit)
28
28
Hex value at offset 28 (address 0x0xffa2b4ac) is 0x63
Enter an offset into your public data and we'll show you the character value.
(or q to quit)

4. **Checkwork**.


student@ubuntu:~/labtainer/labtainer-student$ checkwork
Results stored in directory: /home/student/labtainer_xfer/overrun
Labname overrun
Student | gdb_commands | showed_pin | viewed_ret |
B20DCAT098 | 26 | Y | Y |
What is automatically assessed for this lab:
showed_pin: Display the secret PIN by providing an out-of-bounds index
viewed_ret: Viewed a ret instruction in the disassembly
gdb_commands: How many gdb commands issued by the student
student@ubuntu:~/labtainer/labtainer-student$