LẬP TRÌNH HỆ THỐNG

ThS. Đỗ Thị Thu Hiền
(hiendtt@uit.edu.vn)

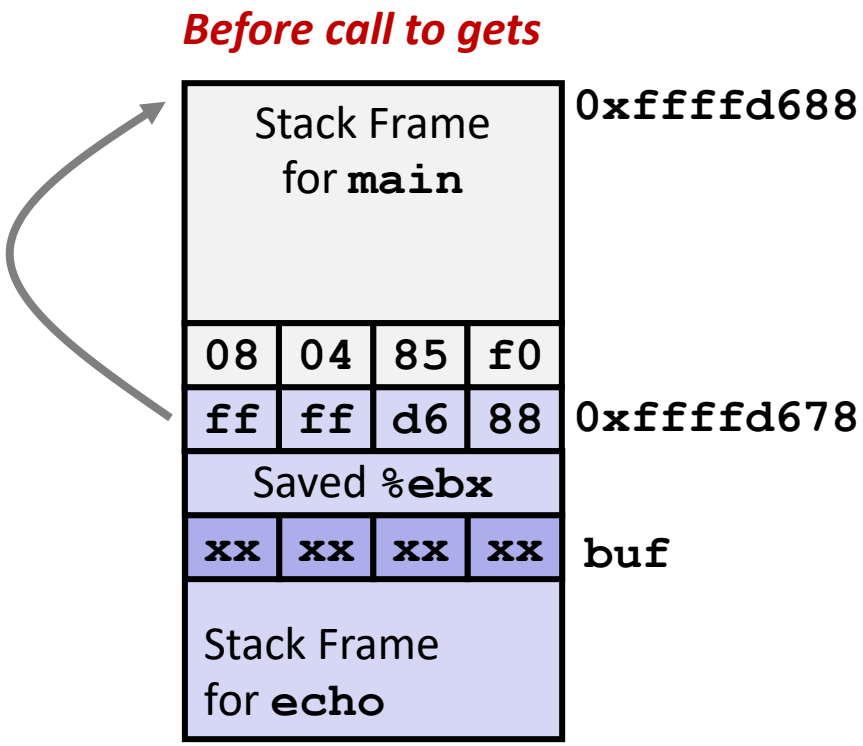
TRƯỜNG ĐH CÔNG NGHỆ THÔNG TIN - ĐHQG-HCM
KHOA MẠNG MÁY TÍNH & TRUYỀN THÔNG
FACULTY OF COMPUTER NETWORK AND COMMUNICATIONS
Tầng 8 - Tòa nhà E, trường ĐH Công nghệ Thông tin, ĐHQG-HCM
Điện thoại: (08)3 725 1993 (122)

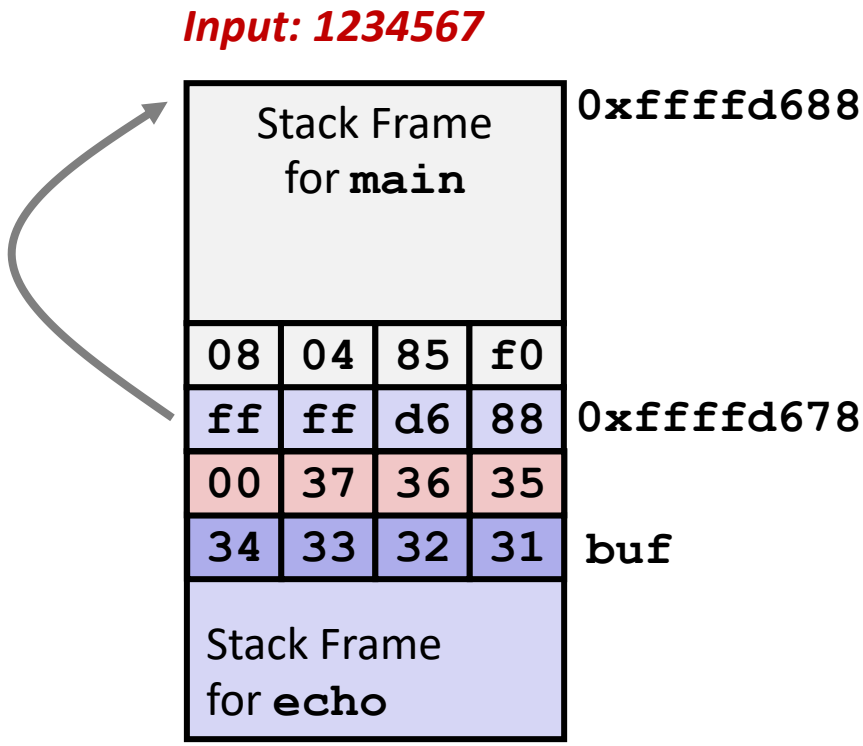
Bài tập Buffer overflow

Nội dung

- **Review**
- **Bài tập**

Buffer Overflow: Ví dụ #1



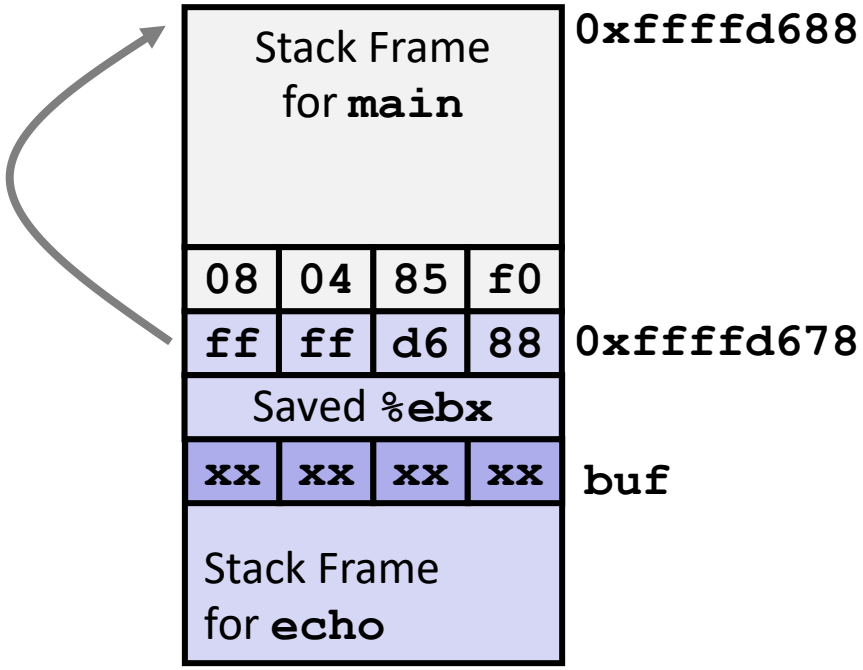



```
unix>./bufdemo
Type a string:1234567
1234567
```

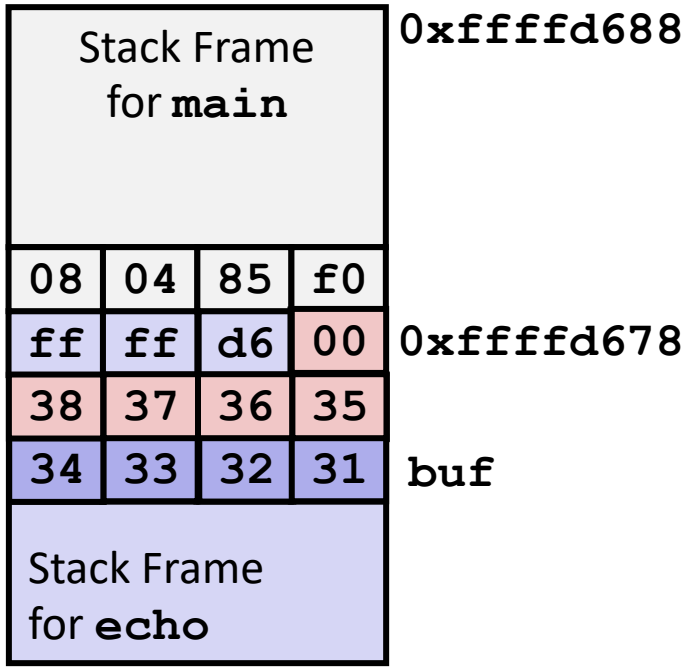
Vượt quá `buf`, ghi đè `%ebx`, nhưng không gây ra vấn đề gì
→ Chỉ làm thay đổi 1 giá trị đã lưu

Buffer Overflow: Ví dụ #2

Before call to gets



Input: 12345678



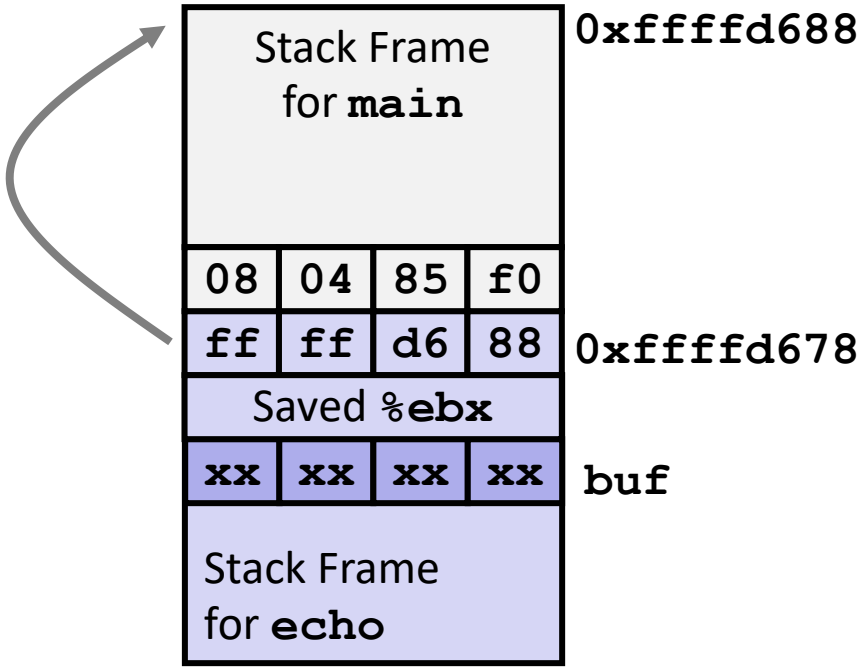

```
unix>./bufdemo
Type a string:12345678
Segmentation Fault
```

Ghi đè `%ebp` cũ → lỗi

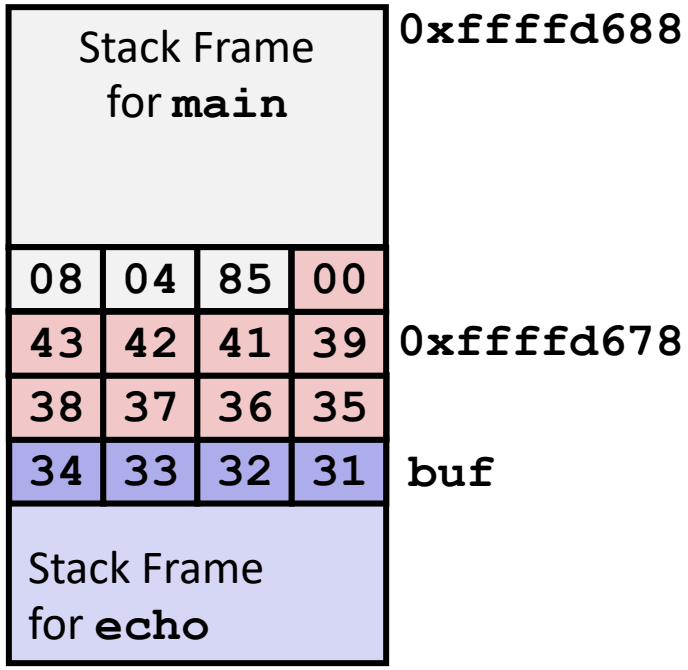
```
. . .
80485eb:    e8 d5 ff ff ff    call   80485c5 <echo>
80485f0:    c9                leave    # Set %ebp to corrupted value
80485f1:    c3                ret
```

Buffer Overflow: Ví dụ #3

Before call to gets



Input: 123456789ABC




```
unix>./bufdemo
Type a string:123456789ABC
Segmentation Fault
```

Return address bị ghi đè

```
80485eb:    e8 d5 ff ff ff    call   80485c5 <echo>
80485f0:    c9                leave    # Desired return point
```

Nội dung

- Review
- **Bài tập**

Buffer Overflow
BÀI TẬP

- **Bài tập cá nhân**
  - simple-buffer: File thực thi Linux 32 bit
- **Bài tập nhóm**
  - Làm trên hệ thống phòng thực hành ảo **vLab**

Bài tập 1: Cá nhân

- **simple-buffer:** File thực thi Linux 32 bit, không canary
- **Hàm `smash_my_buffer`:** đọc 1 input lưu vào **`buf`** với **`gets`**

```
int student_id;         # biến lưu giá trị số nguyên của MSSV từ tham số khi chạy
char student_name[8];   # biến lưu tên của SV từ tham số khi chạy
void smash_my_buffer()
{
    unsigned int var = 0x12345678; ...
    int another_var = 0x40; ...
    char my_name[8] = "student";
    ...
    char buf[12];
    gets(buf);
    if (strcmp(my_name, "student") || var != 0x12345678 || another_var != 0x40){
        printf("You changed my local variables.\n");
        if (strcmp(my_name, student_name) == 0)
            printf("Level 1: DONE..."); ...
        if (another_var == 0x3141)
            printf("Level 2: DONE..."); ...
        if (var == student_id)
            printf("Level 3: DONE..."); ...
    }
}
```

Buffer Overflow: Bài tập 1 (cá nhân)

- **Mục tiêu**: nhập input **`buf`** sao cho gây ra buffer overflow làm thay đổi biến cục bộ:
  - **`my_name`** thành **tên SV**
  - **`another_var`** thành **0x3141**
  - **`var`** thành giá trị số nguyên của **MSSV**

- **Chạy file:** cần cung cấp MSSV và tên SV làm tham số cho file

```
./simple-buffer <MSSV> <ten>
```

- **Vẽ stack**: xác định vị trí tương đối giữa **`buf`** và các biến cục bộ trong stack.

→ Suy ra được độ dài **`buf`** và vị trí các biến cục bộ tương ứng với phần nào trong **`buf`**?

Buffer Overflow: Bài tập 1 (cá nhân)

- **Mục tiêu**: nhập input **`buf`** sao cho gây ra buffer overflow làm thay đổi biến cục bộ:
  - **`my_name`** thành **tên SV**
  - **`another_var`** thành **0x3141**
  - **`var`** thành giá trị số nguyên của **MSSV**
- Gợi ý:
  - Chú ý khi lưu các kiểu dữ liệu lớn hơn 1 byte trên Linux (Little Endian).
  - Chuỗi kết thúc bằng byte NULL (0).
  - Dùng code python để truyền các byte không gõ được từ bàn phím.

  Lưu ý: check phiên bản python trước khi viết code!

input.py

```
exploit_str = "0"*44
exploit_str += "\x0A\x0B\x0C\x0D"

print exploit_str
```

python 2.x.x

input3.py

```
import sys
str = 'a'*44
sys.stdout.buffer.write(str.encode())
x = bytes.fromhex('0A 0B 0C 0D')
sys.stdout.buffer.write(x)
```

python 3.x.x

`$ python` <python file> **`|`** **`./simple-buffer`** <MSSV> <name>

Buffer Overflow: Bài tập 2 (nhóm)

- Truy cập website: **https://vlab.uit.edu.vn/**
- Đăng nhập bằng **tài khoản chứng thực** UIT
- Tên bài tập: **Bài tập Buffer overflow 2**
  - Lớp: **NT209.O21.ANTN.1 – Lập trình hệ thống**

Nội dung

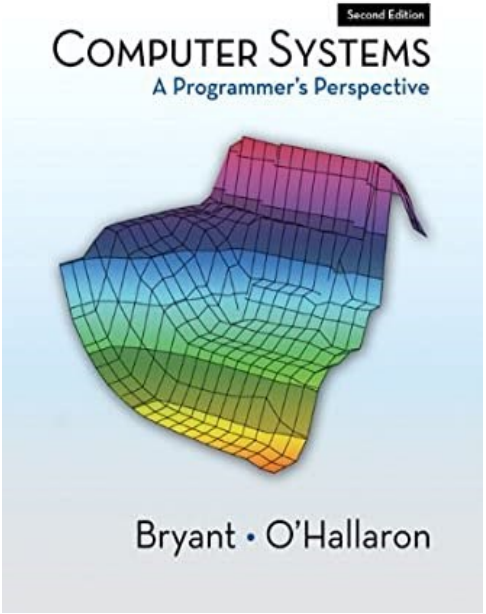
- **Các chủ đề chính:**
  1) Biểu diễn các kiểu dữ liệu và các phép tính toán bit
  2) Ngôn ngữ assembly
  3) Điều khiển luồng trong C với assembly
  4) Các thủ tục/hàm (procedure) trong C ở mức assembly
  5) Biểu diễn mảng, cấu trúc dữ liệu trong C
  6) Một số topic ATTT: reverse engineering, bufferoverflow
  7) Linking trong biên dịch file thực thi
  8) Phân cấp bộ nhớ, cache
- **Lab liên quan**
  - Lab 1: Nội dung **1**
  - Lab 2: Nội dung 1, **2, 3**
  - Lab 3: Nội dung 1, **2, 3, 4, 5, 6**
  - Lab 4: Nội dung 1, **2,** 3**,** **4**, 5, **6**
  - Lab 5: Nội dung 1, **2,** 3**,** **4**, 5, **6**
  - Lab 6: Nội dung **1, 2, 3, 4, 5, 6**

Giáo trình

- **Giáo trình chính**

Computer Systems: A Programmer's Perspective

  - Second Edition (CS:APP2e), Pearson, 2010
  - Randal E. Bryant, David R. O'Hallaron
  - http://csapp.cs.cmu.edu



- **Tài liệu khác**
  - *The C Programming Language*, Second Edition, Prentice Hall, 1988
    - Brian Kernighan and Dennis Ritchie
  - *The IDA Pro Book: The Unofficial Guide to the World's Most Popular Disassembler*, 1st Edition, 2008
    - Chris Eagle
  - *Reversing: Secrets of Reverse Engineering*, 1st Edition, 2011
    - Eldad Eilam


KEEP
CALM
AND
ENJOY YOUR
SEMESTER :)