

# 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



# Bài tập 1 – easy\_buffer

```
#include <stdio.h>
void secret_fun(){
    printf("You've called secret_func. Nice work\n");
    exit(0);
}
void weak_func(){
    int temp = 0;
    char buf[10];
    gets(buf);
    if (temp != 0)
        if (temp == 0x4050)
            printf("Oh wow! You've changed temp to 0x%x\n", temp);
        else
            printf("Humh! temp is 0x%x now\n", temp);
    else
        printf("Nah! temp is still 0x%x\n", 0);
    return;
}
void main(){
    weak_func();
    printf("Back to main. End of program\n");
}
```

# Bài tập 1 – easy\_buffer

Chạy thử:

```
ubuntu@ubuntu: ~/LTHT/LT-demo/buffer/Exercise/Ex1
ubuntu@ubuntu:~/LTHT/LT-demo/buffer/Exercise/Ex1$ ./easy_buffer
Hello world
Humh! temp is 0x64 now
Back to main. End of program
ubuntu@ubuntu:~/LTHT/LT-demo/buffer/Exercise/Ex1$
```

# Bài 1 – easy\_buffer

- **Mục tiêu 1: Thay đổi giá trị biến temp**
  - 1. Giá trị khác 0
  - 2. Giá trị cụ thể 0x4050
- **Vẽ stack:** xác định vị trí tương đối giữa `buf` và biến `temp` trong stack.
  - Cần nhập chuỗi như thế nào để thay đổi biến `temp`?

# Bài 1 – easy\_buffer

- **Mục tiêu 2: Điều hướng gọi hàm secret\_func**
- **Vẽ stack:** xác định vị trí tương đối giữa `buf` và ô nhớ lưu địa chỉ trả về trong stack.  
→ Cần nhập chuỗi như thế nào để thay đổi biến `địa chỉ trả về`?
- **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!**

# Bài 1 – easy\_buffer

Ví dụ: truyền chuỗi gồm 44 ký tự '0' và 4 byte 0x12, 0xAB, 0xCD và 0xEF

## ■ Dùng python

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

```
input.py
exploit_str = "0"*44
exploit_str += "\x12\xAB\xCD\xEF"
print exploit_str
```

python 2.x.x

```
input3.py
import sys
exploit_str = '0'*44
sys.stdout.buffer.write(exploit_str.encode())
x = bytes.fromhex('12 AB CD EF')
sys.stdout.buffer.write(x)
```

python 3.x.x

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

## ■ Dùng file hex2raw (Lab 6)

Chuẩn bị 1 file text chứa các byte hexan cần truyền

```
input.txt
30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30
30 30 30 30 12 AB CD EF
```

**./hex2raw** < <byte\_code\_file> | **./simple-buffer** <MSSV> <name>

# Bài 1 – easy\_buffer

- **Mục tiêu 3: Thực hiện đồng thời**
  - Thay đổi biến temp thành giá trị cụ thể 0x4050
  - Điều hướng chương trình đến secret\_func
- **Kết hợp chuỗi khai thác ở mục tiêu 1 và 2**

```
ubuntu@ubuntu: ~/LTHT/LT-demo/buffer/Exercise/Ex1
ubuntu@ubuntu:~/LTHT/LT-demo/buffer/Exercise/Ex1$ python code.py | ./easy_buffer
Oh wow! You've changed temp to 0x4050
You've called secret_func. Nice work
ubuntu@ubuntu:~/LTHT/LT-demo/buffer/Exercise/Ex1$
```

# Bài 1 – easy\_buffer

- **Bonus: Nếu secret\_func không có exit(0)**
  - Thay đổi biến temp thành giá trị cụ thể 0x4050
  - Điều hướng chương trình đến secret\_func

- **Tại sao lỗi Segmentation fault?**

```
ubuntu@ubuntu: ~/LTHT/LT-demo/buffer/Exercise/Ex1
ubuntu@ubuntu:~/LTHT/LT-demo/buffer/Exercise/Ex1$ python code.py | ./easy_buffer_noexit
Oh wow! You've changed temp to 0x4050
You've called secret_func. Nice work
Segmentation fault (core dumped)
ubuntu@ubuntu:~/LTHT/LT-demo/buffer/Exercise/Ex1$
```

# Bài tập 2 – buffer1

```
#include <stdio.h>
#include <stdlib.h>
void get_shell(){
    printf("Yay! You've called get_shell function\n");
    system("/bin/sh");
    exit(0);
}
void buf_overflow(){
    int a = 0;
    char buf[25];
    gets(buf);
    puts(buf);
    if (a == 0xdeadbeef){
        printf("Return to my caller...\n");
        return;
    }
    else exit(0);
}
int main(){
    buf_overflow();
}
```

Yêu cầu: khai thác hàm **buf\_overflow()** để gọi được hàm **get\_shell()** và truyền lệnh