# 实验 - pwn基础实验

## 实验简介

**本次课程讲解了在 linux 上动态调试器 gdb 的使用，对二进制基础的内容做了实操**

**此外，为了学习 stack overflow 漏洞和利用，课堂上快速讲解了栈的原理，以及栈上缓冲区溢出能够达成的破坏效果，本次实验将对这些内容进行巩固**

## 基础部分

### gdb 调试 30 分

请调试hello world程序并完成如下任务

- 通过b下断点到main并执行c后触发断点
- 断在printf/puts的 call 汇编位置并查看参数
- 通过单步跟踪找到 main 函数 ret 后返回到的函数

### 栈原理 30 分

给定如下源码与汇编代码

```
#include <stdio.h>

int func(int a, int b, int c)
{
    int d;
    long e;
    char buf[32];

    d = a + b + c;
    e = a - b - c;
    buf[0] = a + c;
    buf[16] = a - c;

    return d << b;
}

int main(int argc, char* argv[])
{
    func(1, 5, 9);
    return 0;
```

```
}
```

```
000000000000066a <func>:
 66a:   55                      push   rbp
 66b:   48 89 e5                mov    rbp,rsp
 66e:   48 83 ec 50             sub    rsp,0x50
 672:   89 7d bc                mov    DWORD PTR [rbp-0x44],edi
 675:   89 75 b8                mov    DWORD PTR [rbp-0x48],esi
 678:   89 55 b4                mov    DWORD PTR [rbp-0x4c],edx
 67b:   64 48 8b 04 25 28 00    mov    rax,QWORD PTR fs:0x28
 682:   00 00
 684:   48 89 45 f8             mov    QWORD PTR [rbp-0x8],rax
 688:   31 c0                   xor    eax,eax
 68a:   8b 55 bc                mov    edx,DWORD PTR [rbp-0x44]
 68d:   8b 45 b8                mov    eax,DWORD PTR [rbp-0x48]
 690:   01 c2                   add    edx,eax
 692:   8b 45 b4                mov    eax,DWORD PTR [rbp-0x4c]
 695:   01 d0                   add    eax,edx
 697:   89 45 c4                mov    DWORD PTR [rbp-0x3c],eax
 69a:   8b 45 bc                mov    eax,DWORD PTR [rbp-0x44]
 69d:   2b 45 b8                sub    eax,DWORD PTR [rbp-0x48]
 6a0:   2b 45 b4                sub    eax,DWORD PTR [rbp-0x4c]
 6a3:   48 98                   cdqe
 6a5:   48 89 45 c8             mov    QWORD PTR [rbp-0x38],rax
 6a9:   8b 45 bc                mov    eax,DWORD PTR [rbp-0x44]
 6ac:   89 c2                   mov    edx,eax
 6ae:   8b 45 b4                mov    eax,DWORD PTR [rbp-0x4c]
 6b1:   01 d0                   add    eax,edx
 6b3:   88 45 d0                mov    BYTE PTR [rbp-0x30],al
 6b6:   8b 45 bc                mov    eax,DWORD PTR [rbp-0x44]
 6b9:   89 c2                   mov    edx,eax
 6bb:   8b 45 b4                mov    eax,DWORD PTR [rbp-0x4c]
 6be:   29 c2                   sub    edx,eax
 6c0:   89 d0                   mov    eax,edx
 6c2:   88 45 e0                mov    BYTE PTR [rbp-0x20],al
 6c5:   8b 45 b8                mov    eax,DWORD PTR [rbp-0x48]
 6c8:   8b 55 c4                mov    edx,DWORD PTR [rbp-0x3c]
 6cb:   89 c1                   mov    ecx,eax
 6cd:   d3 e2                   shl    edx,cl
 6cf:   89 d0                   mov    eax,edx
 6d1:   48 8b 75 f8             mov    rsi,QWORD PTR [rbp-0x8]
 6d5:   64 48 33 34 25 28 00    xor    rsi,QWORD PTR fs:0x28
 6dc:   00 00
 6de:   74 05                   je     6e5 <func+0x7b>
 6e0:   e8 5b fe ff ff          call   540 <__stack_chk_fail@plt>
 6e5:   c9                      leave
 6e6:   c3                      ret
```

回答如下问题

- stack frame 给临时变量预留的空间大小为?
- 变量 d 相对于 frame pointer 偏移?
- 变量 e 相对于 frame pointer 偏移?
- 数组 buf 相对于 frame pointer 偏移?

## stack overflow 的破坏能力 40分

复现课上给定的 overflow example（要求使用 pwntools 完成）

- example1.c 完成对于局部变量 `rootuser` 的覆盖（要求程序不会段错误退出
- example2.c 完成对于返回地址的破坏，触发程序段错误
- example3.c 完成对于局部指针变量 `ptr` 的破坏从而更改控制流
- example4.c 完成对于frame pointer的破坏，并调试查看栈变化的效果

# 挑战部分

来看看大二的课程部分的overflow作业吧

完成软件安全课程的实验1-3

- https://gitee.com/zjuicsr/ssec22spring-stu/wikis/hw-01%20%E7%BC%93%E5%86%B2%E5%8C%BA%E6%BA%A2%E5%87%BA%E6%94%BB%E5%87%BB

# 拓展问题和阅读

拓展主要在保护机制上

- 理解 stack canary 的实现细节
- 理解 ASLR 随机会影响的部分