**深 圳 大 学 实 验 报 告**

**课 程 名 称： 计算机系统(2)**

**实验项目名称： 缓冲区溢出攻击实验**

**学 院： 计算机与软件学院**

**专 业： 计算机科学与技术**

**指 导 教 师： 马晨琳**

**报告人： 郑雨婷 学号： 2021150122 班级： 高性能**

**实 验 时 间： 2023年4月27日~5月7日**

**实验报告提交时间： 2023年5月7日**

**教务处制**

**一、 实验目标：**

1. 理解程序函数调用中参数传递机制；
2. 掌握缓冲区溢出攻击方法；
3. 进一步熟练掌握GDB调试工具和objdump反汇编工具。

**二、实验环境：**

1. 计算机（Intel CPU）
2. Linux 64位操作系统
3. GDB调试工具
4. objdump反汇编工具

**三、实验内容**

本实验设计为一个黑客利用缓冲区溢出技术进行攻击的游戏。我们仅给黑客（同学）提供一个二进制可执行文件bufbomb和部分函数的C代码，不提供每个关卡的源代码。程序运行中有3个关卡，每个关卡需要用户输入正确的缓冲区内容，否则无法通过管卡！

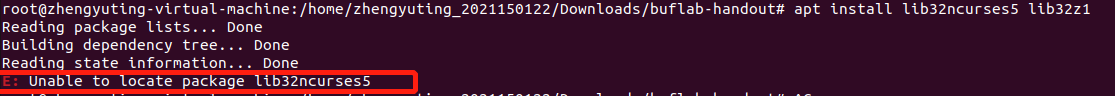
要求同学查看各关卡的要求，运用GDB调试工具和objdump反汇编工具，通过分析汇编代码和相应的栈帧结构，通过缓冲区溢出办法在执行了getbuf()函数返回时作攻击，使之返回到各关卡要求的指定函数中。第一关只需要返回到指定函数，第二关不仅返回到指定函数还需要为该指定函数准备好参数，最后一关要求在返回到指定函数之前执行一段汇编代码完成全局变量的修改。

实验代码bufbomb和相关工具（sendstring/makecookie）的更详细内容请参考“实验四 缓冲区溢出攻击实验.pptx”。

本实验要求解决关卡1、2、3，给出实验思路，通过截图把实验过程和结果写在实验报告上。

**四、实验步骤和结果**

因为本次实验用到的可执行文件是32位，而实验环境是64位的，需要先安装一个32位的库，在root权限下安装如下所示：



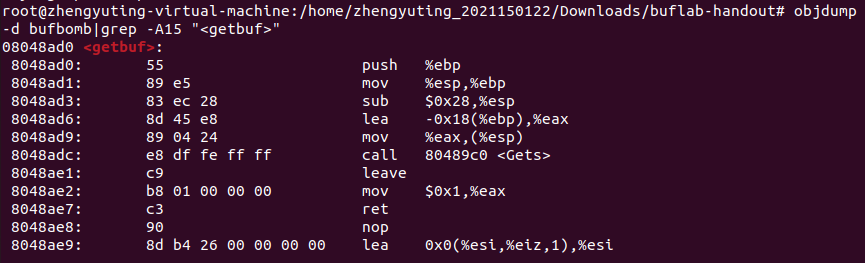
出现错误提示找不到软件包，上网查了发现是通常是软件包名输入错误， lib32ncurses5改为lib32ncurses5-dev即可正常安装。



还需要安装sendmail



首先利用反汇编命令查看getbuf函数的汇编代码，以便分析getbuf在调用<Gets>时的栈帧结构，汇编代码如下：



**步骤1 返回到smoke()**

* 1. **解题思路**

本实验中，bufbomb中的test()函数将会调用getbuf()函数，getbuf()函数再调用gets()从标准输入设备读入字符串。

系统函数gets()未进行缓冲区溢出保护。其代码如下：

***int getbuf()***

***{***

***char buf[12];***

***Gets(buf);***

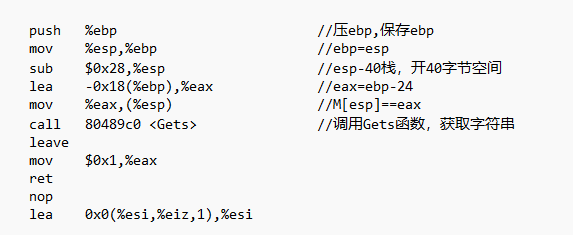
***return 1;***

***}***

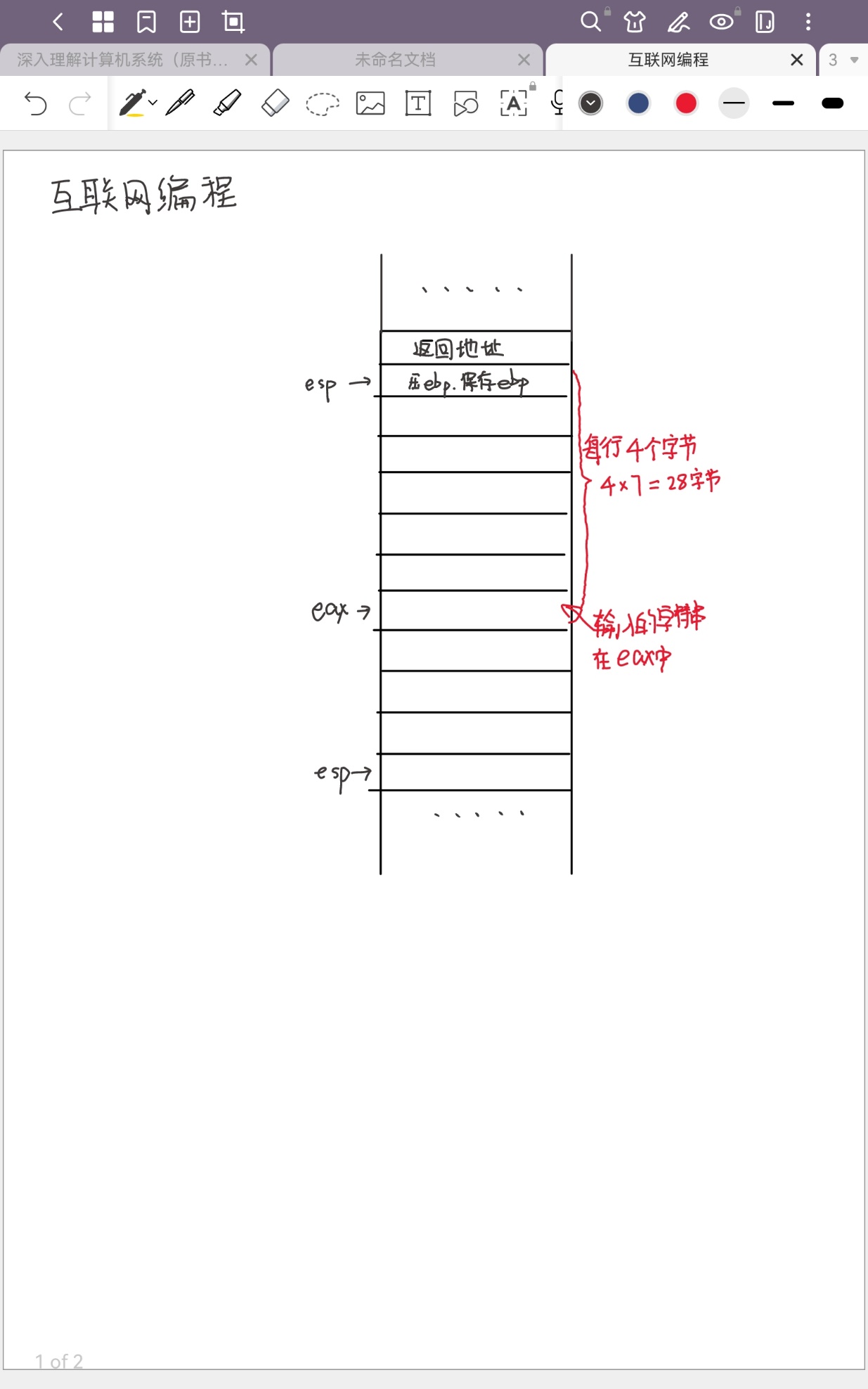
我们的目标是使getbuf()返回时，不返回到test()，而是直接返回到指定的smoke()函数。

为此，我们可以通过构造并输入大于getbuf()中给出的数据缓冲区的字符串而破坏getbuf()的栈帧，替换其返回地址，将返回地址改成smoke()函数的地址。

* 1. **解题过程**



分析getbuf（）函数的汇编代码，画出栈帧结构（在32位系统下，每行是4个字节）。可以看到，首先在test()函数中调用getbuf()时，先压入初始返回地址。然后将ebp压入栈，保存ebp的初值，将esp存入ebp。之后esp=esp-40,开辟40字节的空间，eax=ebp-24，输入的字符串存在eax中。从画出的栈帧结构图可以看出：eax到返回地址之间有28字节的空间。



因此我们需要先任意输入28字节的字符串，然后确保最后4字节为smoke()函数的地址。利用objdump查看smoke（）函数的地址如下：



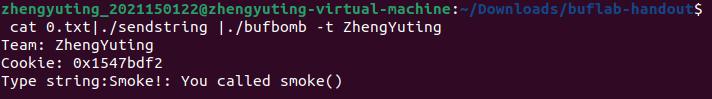
得知smoke()函数的地址为0x08048eb0,需要注意一般操作系统都使用小端法，因此要将地址转换为b08e0408。所以我们输入的字符串可以为：“12345678123456781234567812345678123456781234567812345678**b08e0408**”.

**1.3 最终结果截图**

输入的字符串保存到0.txt，用cat查看0.txt内容：

然后通过sendstring将0.txt转换成二进制格式，在通过管道输入到bufbomb中：

$ cat 0.txt|./sendstring |./bufbomb -t ZhengYuting



结果正确，调用smoke（）函数成功！

**步骤2 返回到fizz()并准备相应参数**

**2.1解题思路**

与第一关的整体思路相同，输入大于缓冲区的字符串来进行缓冲区溢出攻击。第二关需要将返回地址改成fizz()函数的地址,而且还需为fizz()函数准备参数。利用objdump反汇编命令查看fizz()函数的汇编代码，分析得出参数存在哪个位置，将其替换成自定义的cookie值即可。

**2.2解题过程**

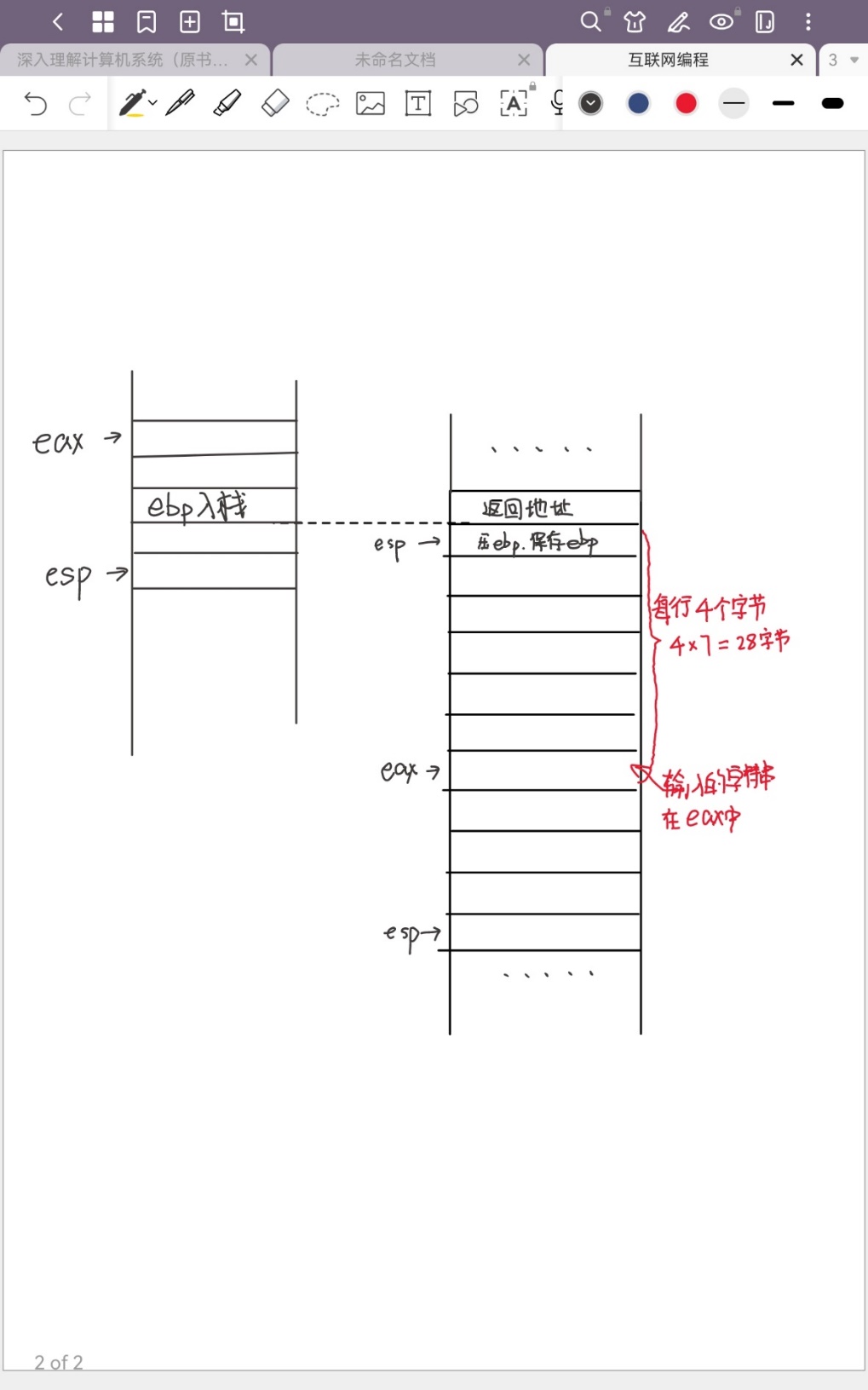
利用objdump反汇编命令查看fizz()函数的汇编代码：



分析fizz()的汇编代码,画出栈帧结构（下图左侧）。可以看到，在调用fizz()时，压入初始返回地址。然后将ebp压入栈，保存ebp的初值，将esp存入ebp。之后esp=esp-8,开辟8字节的空间，eax=ebp+8，参数存在于eax地址中。利用gdb查看0x804a1d4的内容：

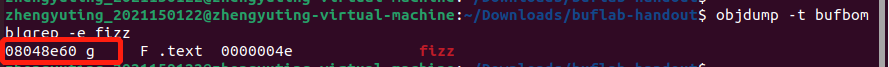


因此，比较exa与0x804a1d4即为比较参数与cookie的内容。



然后看getbuf()函数的栈帧（上图右侧），当执行到返回地址为fizz()的地址后，该地址开始就为fizz()函数的栈帧，随即执行fizz()第一条语句存入ebp。因此上图两橙色框中是同一块栈帧区域，看出参数在返回地址上8个字节的位置，所以输入的字符串应该是1-28个字节任意，29-32个字节为fizz()返回地址，33-36字节任意，37-40字节为参数。

利用objdump查看fizz（）函数的地址为0x08048e60,小端法表示为608e0408



参数要与cookie相等才能验证成功，通过makecookie工具根据姓名设置cookie值为0x1547bdf2,小端法表示为f2bd4715。



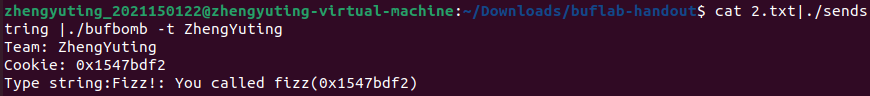
所以我们输入的字符串可以为：“12345678123456781234567812345678123456781234567812345678**608e0408**12345678**f2bd4715**”。

**2.3最终结果截图**

输入的字符串保存到2.txt，用cat查看2.txt内容：

然后通过sendstring将2.txt转换成二进制格式，在通过管道输入到bufbomb中：

$ cat 2.txt|./sendstring |./bufbomb -t ZhengYuting



结果正确，调用fizz（）函数成功！

**步骤3 返回到bang()且修改global\_value**

**3.1解题思路**

最后一关与前两关的不同之处在于，要求在返回到指定函数之前执行一段汇编代码完成全局变量global\_value的修改。需要在输入数据中包含黑客构造的代码，因此需要通过编写C或汇编代码并通过GCC产生目标文件，从而得到所需的代码。

**3.2解题过程**

首先，在 root 权限下关闭 Linux 的内存地址随机化，以确保能够精确低指定跳转地址：

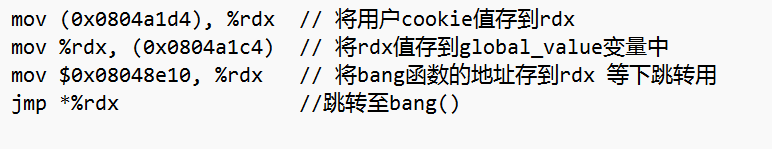


利用objdump反汇编命令查看bang()函数的汇编代码：



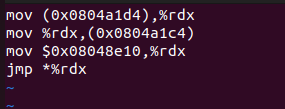
分析bang（）函数代码，可知该函数比较了0x804a1c4与0x804a1d4，若相同则验证成功。根据第二关我们已知0x804a1d4地址为cookie。利用gdb查看0x804a1c4地址为global\_value.

因此我们需要编写汇编代码，先将global\_value的值改为cookie的值，即为第二关设置的0x1547bdf2，然后再跳转到bang()函数。写出的汇编代码如下：



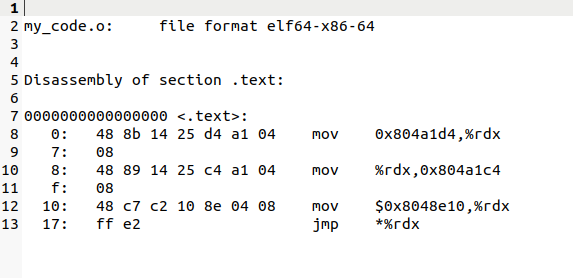
用vi编辑器创建并书写my\_code.s文件。



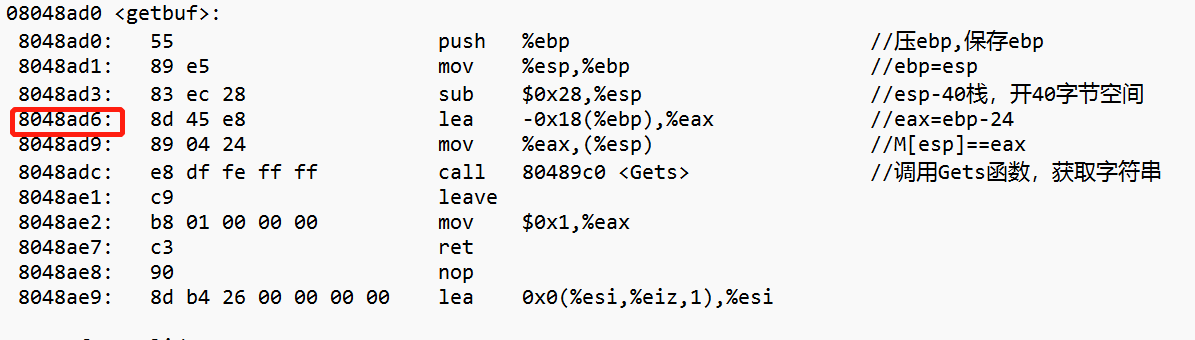


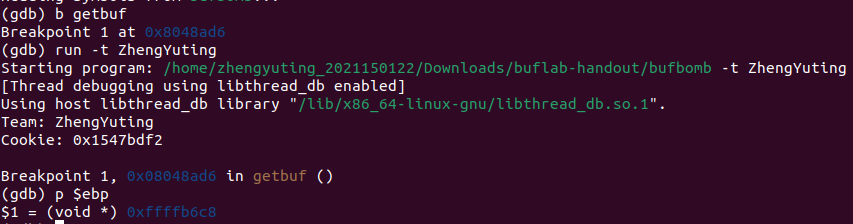
通过 gcc 编译该汇编代码，再反编译输出到my\_code.txt 中，查看my\_code.txt文件可以得到写出的汇编指令对应的机器代码。





所以，输入的字符串前几个字节为这一段机器代码，存入buf数组，然后将getbuf()的返回地址改为buf数组的地址。为此，用gdb断点调试查看执行getbuf()时ebp的值为0xffffb6c8。





从第一关中对 getbuf()函数栈帧结构的分析可知 buf 数组的首地址为ebp-0x18，即 0xffffb6b0,小端法表示为b0b6ffff。

所以输入的字符串应该是1-25个字节为机器码，26-28个字节为任意字符，29-32字节为参buf 数组的首地址。所以我们输入的字符串可以为：

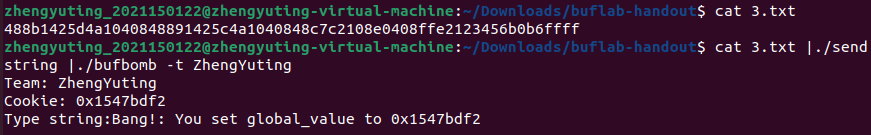
**488b1425d4a10408848891425c4a1040848c7c2108e0408ffe2**123456**b0b6ffff**;

**3.3最终结果截图**

输入的字符串保存到3.txt，用cat查看3.txt内容：

然后通过sendstring将3.txt转换成二进制格式，在通过管道输入到bufbomb中：

$ cat 3.txt|./sendstring |./bufbomb -t ZhengYuting



结果正确，调用bang（）函数成功！

**五、实验总结与体会**

本次实验与实验三类似，都要分析汇编代码输入正确的字符串来通过关卡。但这次实验主要侧重点是对程序函数调用中参数传递机制考察和对缓冲区溢出攻击的理解。需要对帧栈结构了解的清楚才能够进行实验。

本次实验，用到了gdb调试工具、ojbdump反汇编工具vi编辑器等，复习了之前实验的内容。除了分析代码，并且在第三关需要自己编写一小段代码，对理论课上的学习的各项指令、寻址方式、寄存器等知识都进行了复习。还学习到了sendstring的用法。是一次综合性强、收获多的实验。

|  |
| --- |
| **指导教师批阅意见：**  **成绩评定：**  指导教师签字： 马晨琳    2023年 5月 日 |
| 备注： |