**深 圳 大 学 实 验 报 告**

**课程名称： 计算机系统(2)**

**实验项目名称： 逆向工程实验**

**学院： 计算机与软件学院**

**专业： 计算机与软件学院所有专业**

**指导教师： 刘 刚**

**报告人： 学号： 班级：**

**实验时间： 2018年4月18日至5月15日**

**实验报告提交时间： 2018年5月X日**

**教务处制**

|  |
| --- |
| **一、 实验目标与要求：**   1. 理解程序（控制语句、函数、返回值、堆栈结构）是如何运行的 2. 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. Linux64位操作系统（Ubuntu 17） 3. GDB调试工具 4. objdump反汇编工具 |
| **三、实验方法与步骤：**  本实验设计为一个黑客拆解二进制炸弹的游戏。我们仅给黑客（同学）提供一个二进制可执行文件bomb和主函数所在的源程序bomb.c，不提供每个关卡的源代码。程序运行中有6个关卡（6个phase），每个关卡需要用户输入正确的字符串或数字才能通关，否则会引爆炸弹（打印出一条错误信息，并导致评分下降）！  要求同学运用**GDB调试工具和objdump反汇编工具**，通过分析汇编代码**，**找到在每个phase程序段中，引导程序跳转到“explode\_bomb”程序段的地方，并分析其成功跳转的条件，以此为突破口寻找应该在命令行输入何种字符串来通关。  本实验需解决Phase\_1(**15分**)、Phase\_2(**15分**)、Phase\_3(**15分**)、Phase\_4(**15分**)、Phase\_5(**15分**)、Phase\_6(**10分**)。通过**截图+文字**的形式把实验过程写在实验报告上，最后并撰写**实验结论与心得(15分**)。 |
| **四、实验过程及内容：**  **以下是某助教的实验过程以及实验思路，仅供同学们学习参考。请同学们学习后删除以下教程内容（包括此段红字），并基于自己的Linux用户（用户名为学生名称加学号）进行实验，认真记录自己的实验过程以及实验思路，并撰写实验结论与心得。**  注1：  当前用户可能对bomb文件没有执行权限，建议先用ls查看文件权限，若没有执行权限，请用chmod +x bomb命令增加当前用户对bomb文件的执行权限。  注2：  若用命令./bomb执行该文件提示no such file时，请先用 su szu 切换到szu账户，然后输入命令sudo apt-get install lib32z1，等待下载安装完毕即可。   * 第一关  1. 首先对bomb文件进行反汇编，并将结果输出到1.txt。      1. 为方便查阅汇编代码，在图形化界面中将txt结果用文本编辑器打开。 2. 已知本实验共有6个关卡，下面先开始第一关，打开txt，定位到phase\_1函数处。      1. 【主要思路】 通过阅读汇编代码可知，里面有一个strings\_not\_equal函数，故我们推测它进行字符串比较。且比较后的结果，有je跳转指令，可能是相等则通关。故程序自身应有一个字符串，经代码检测，关键点在0x80497c0。故下面进入gdb调试，然后查看0x80497c0该地址对应的数据。      1. 由上图可知，第一关的答案是“Public speaking is very easy.”。 2. 对答案进行测试。成功过关！   上面一段请在最终的实验报告中，请删除，包括此句话。 |
| **五、实验结论：** |
| **六、心得体会：** |

|  |
| --- |
| 指导教师批阅意见：  成绩评定：  指导教师签字：刘刚  2018年 5月15 日 |
| 备注： |

**三、实验内容**

本实验设计为一个黑客拆解二进制炸弹的游戏。我们仅给黑客（同学）提供一个二进制可执行文件bomb和主函数所在的源程序bomb.c，不提供每个关卡的源代码。程序运行中有6个关卡（6个phase），每个关卡需要用户输入正确的字符串或数字才能通关，否则会引爆炸弹（打印出一条错误信息，并导致评分下降）！

要求同学运用**GDB调试工具和objdump反汇编工具**，通过分析汇编代码**，**找到在每个phase程序段中，引导程序跳转到“explode\_bomb”程序段的地方，并分析其成功跳转的条件，以此为突破口寻找应该在命令行输入何种字符串来通关。

本实验需解决Phase\_1(**15分**)、Phase\_2(**15分**)、Phase\_3(**15分**)、Phase\_4(**15分**)、Phase\_5(**15分**)、Phase\_6(**10分**)。通过**截图+文字**的形式把实验过程写在实验报告上，最后并撰写**实验结论与心得(15分**)。

**四、实验步骤和结果**

**以下是某助教的实验过程以及实验思路，仅供同学们学习参考。请同学们学习后删除以下教程内容（包括此段红字），并基于自己的Linux用户（用户名为学生名称加学号）进行实验，认真记录自己的实验过程以及实验思路，并撰写实验结论与心得。**

注1：

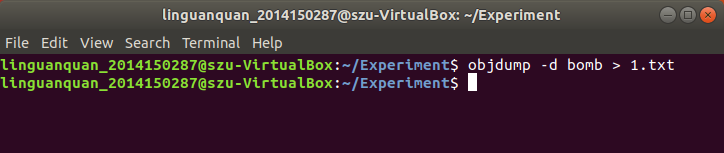
当前用户可能对bomb文件没有执行权限，建议先用ls查看文件权限，若没有执行权限，请用chmod +x bomb命令增加当前用户对bomb文件的执行权限。

注2：

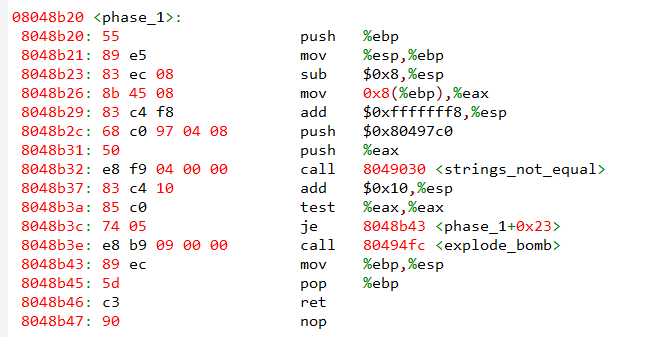
若用命令./bomb执行该文件提示no such file时，请先用 su szu 切换到szu账户，然后输入命令sudo apt-get install lib32z1，等待下载安装完毕即可。

* 第一关

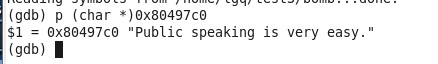
1. 首先对bomb文件进行反汇编，并将结果输出到1.txt。



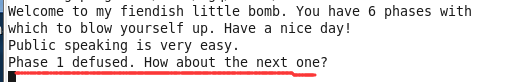
1. 为方便查阅汇编代码，在图形化界面中将txt结果用文本编辑器打开。
2. 已知本实验共有6个关卡，下面先开始第一关，打开txt，定位到phase\_1函数处。



1. 【主要思路】 通过阅读汇编代码可知，里面有一个strings\_not\_equal函数，故我们推测它进行字符串比较。且比较后的结果，有je跳转指令，可能是相等则通关。故程序自身应有一个字符串，经代码检测，关键点在0x80497c0。故下面进入gdb调试，然后查看0x80497c0该地址对应的数据。

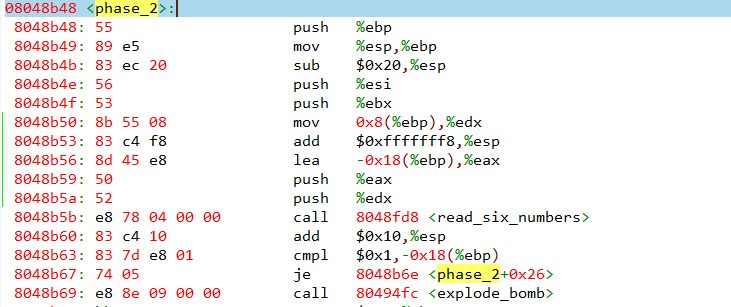


1. 由上图可知，第一关的答案是“Public speaking is very easy.”。
2. 对答案进行测试。成功过关！

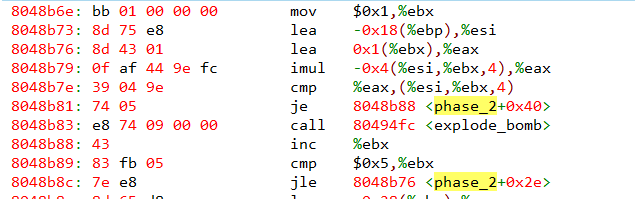


* 第二关

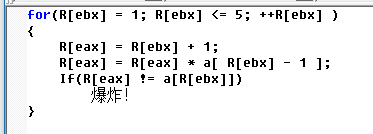
1. 查找phase\_2函数所在处。（部分代码如下）



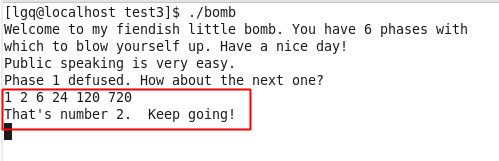
1. 【主要思路①】由PPT的提示可知，本关卡主要考察循环和数组。那么观察代码，我们看到一个函数 read\_six\_numbers， 从命令来看，推测应该是读取6个数，那么我们可以大概确定会对比6个数，且只能是6个。
2. 【主要思路②】然后，下面有一个数字1的比较，说明第一个数只能是1，否则boom。
3. 继续往下分析，抓到以下关键代码。



1. 【主要思路③】通过对寄存器的分析，可知esi存放的是数组首地址，ebx是个计数器，后面cmp $0x5, %ebx 可知循环的范围是1-5，综合上面，可大概转换成比较直观的C语言伪算法。



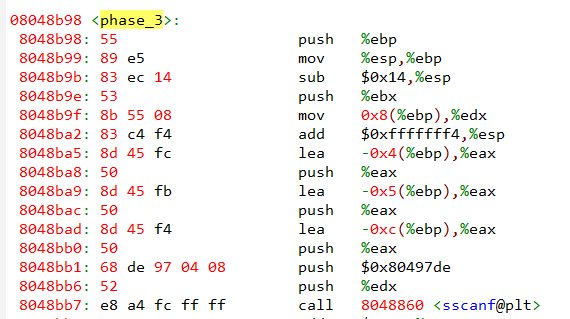
1. 故将开始的1代入，依次可得后面的数 2\*1， 3\*2， 4\*6， 5\*24， 6\*120。
2. 测试结果。1、2、6、24、120、720（输入时用将、换成空格）



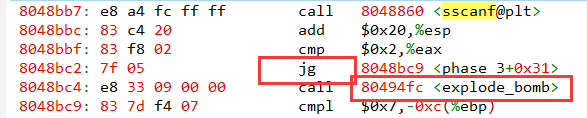
1. Yeah,思路没错，通关！

* 第三关

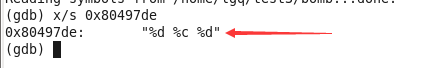
1. 代码跳转到phase\_3，开始分析其代码。部分代码截图如下：



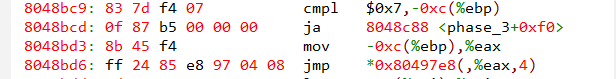
1. 【主要思路①】由上图代码可知，其对eax进行了三次入栈（三个地址），再调用了sscanf函数，则，大概猜测它是有3个参（因为输入函数后面的参数一般是地址）。继续分析下面的代码，可见其拿eax与2比较，如果不是大于2则爆炸。更说明猜测的正确性，确实要3个参。

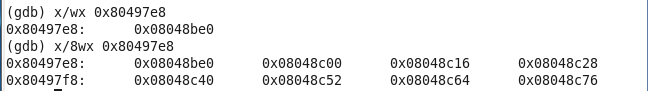


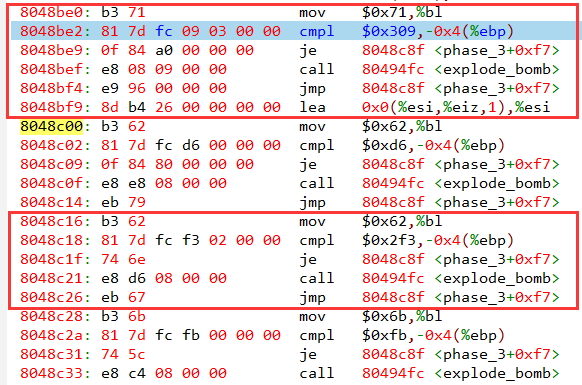
1. 既然估计要三个参，那么参数类型的什么？通过查阅资料知 ，输入函数还会将格式串入栈的，因为函数需要用到。所以查看下对应的内容就知道了输入的参数格式了。查看如下图。可知，第一个是数，第二个是字符，第三个是数。



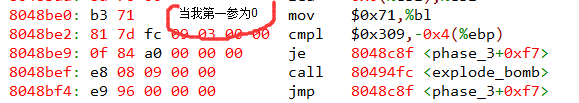
1. 【主要思路②】继续往下。其的第一个参数与7进行比较，若大于7，则爆炸。所以，可得第一个数得小于等于7。然后，从jmp那条语句可知，结果会是根据第一个参数指跳转到不同的地址，具体地址是： M[0x80498e8+ R[eax] \* 4]。 所以，目测0x80498e8是一个地址数组的开端，查看它，发现内容是0x08048be0。再查以此开端的8个地址内容。结合代码可知，不同整数会导致判断条件不一样。那么可猜测这是个If条件判断，或者switch。（不过PPT明确了是switch）



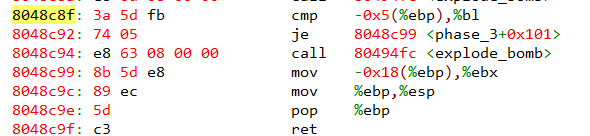




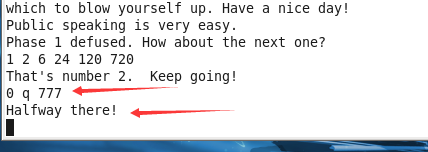
1. 【主要思路③】既然有了多种选择（也就是有多种答案），那么在这则选择其中一种进行向下分析。我选择了第一参数为数字0。因为第一个参为0，所以跳转到0x08048be0， 在这里，它把一个内容保存在bl寄存器。再使0x309（也就是十进制的777）与我输入的第三个参比较，只有相等，才继续跳转。那么可知，若第一个参为0，那么第三个参为777。



1. 【主要思路④】跳到8048C8F，可见其那之前保存到bl寄存器的值与第二个参比较，只有相等，才能跳过爆炸的代码，第三个函数完毕。说明bl寄存器的值就是第二个参答案，此时其为0x71（对应字母为q）。故，在此，已成功得出一组答案，“0 q 777”。



1. 测试这一组答案。



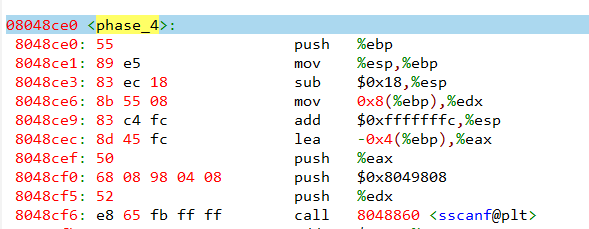
1. 由上图可见，答案正确。故同理，可找出全部答案。

“0 q 777” 、 “1 b 214” 、 “2 b 755” 、 “3 k 251”

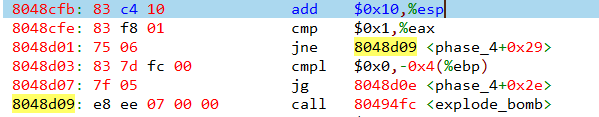
“4 o 160” 、 “5 t 458” 、 “6 v 780” 、 “7 b 524”

* 第四关

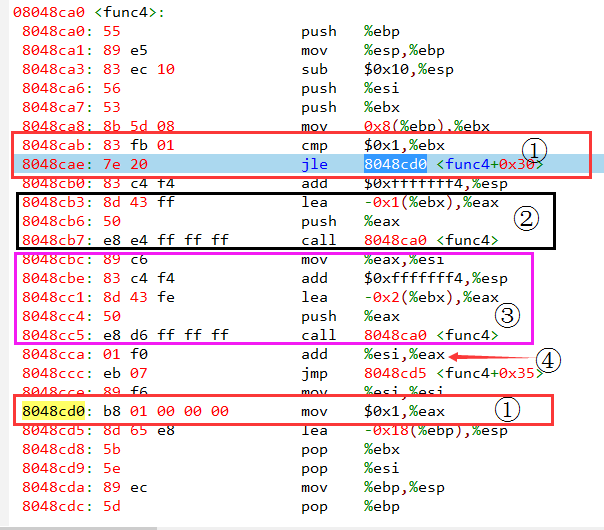
1. 首先还是跳到该关对应的部分。查看其代码，部分代码如下：

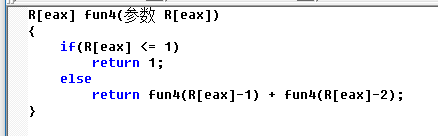


1. 【主要思路①】由上图的部分代码可知，调用sscanf函数前eax进栈，查看一下格式串，可知是输入一个数。后面的edx进栈，发现前面的关卡都存在。但前面关卡都没用到它，所以估计它有其他的作用，所以暂不管它。所以目前得到的信息是：输入的是一个整数。
2. 【主要思路②】继续往下分析，可见下面的代码判断输入的整数是否为1个，而且这个整数是否大于0，两个条件需要同时满足才不会爆炸。



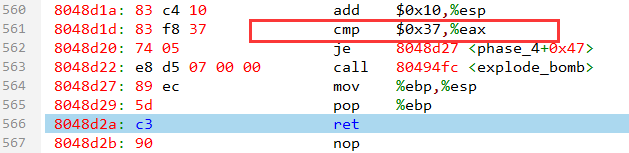
1. 【主要思路③】继续往下，进入分析fun4函数。纵观整体，可见它自己调用自己，所以目测是递归。故第一部分估计就是它的递归出口。只有参数小于等于1，就不再调用自己。而且，小于等于1的结果都是1。从第二部分可知，是把原参数减去1作为新参数。从第三部分可知是把原参数减去2作为新参数。而且最后把他们的结果相加，作为最终结果。所以，若换成C语言的形式，则是如下：





1. 【主要思路④】继续回到执行fun4()后的Phase\_4代码。可知，只有eax等于0x37才不会爆炸。故现在的问题是输入什么值，才可以让fun4的返回值为55(十进制)。 其实我们通过看上面的fun4()的C表达形式，发现这应该是一个斐波拉契数列。我们可以将R[eax]从0开始，一直列出R[eax]增加时的结果。可得：

1 1 2 3 5 8 13 21 34 55（R[eax] = 0,1, ….., 9）故答案为9



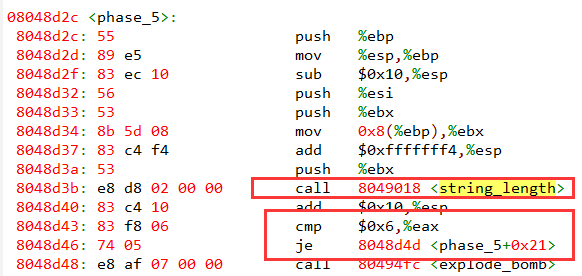
1. 测试答案。



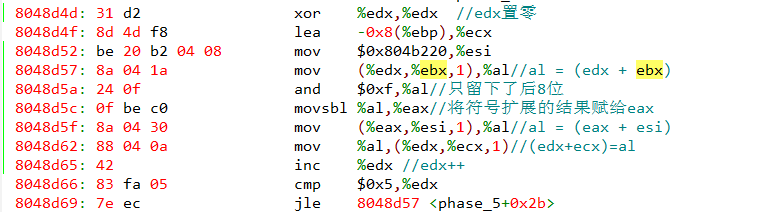
1. 第四关，成功通过！

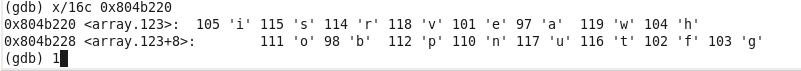
* 第五关

1. 先跳到phase\_5的代码（部分代码如下）



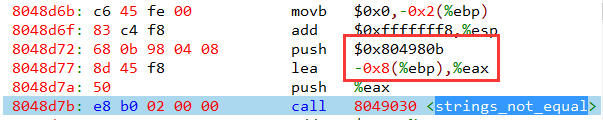
1. 【主要思路①】先分析上图。其先保存了R[ebx]的原值（压栈），再修改R[ebx]的值（这里应是输入的字符串的首地址），然后其调用了字符串长度统计函数，后面是6的比较，所以是字符串的长度必须为6。
2. 【主要思路②】继续分析，如下图。其先把ebx的内容拿出来，取后4位，再根据这个数字，来确定要copy那个内存单元的内容, 相对于这个数字就是数组下标。总表达式为M(R[ebp]-8+R[edx])=M[R[esi]+M(R[ebx]+edx)&0xf]。所以现在可以看看R[esi]对应的内容是什么（见下图）。





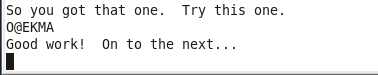
1. 【主要思路③】既然上面知道了要根据用户输入的字符串，生成新的字符串。那么要什么样的字符串?答案估计就在下面的字符串比较了。我们可见两个参数入栈了。故查看下程序要哪什么字符串进行对比。通过命令查看内存可知，0x804980b是“giants”的首地址。所以我们要生成“giants”字符串。由步骤三的内存图可知，我们输入的字符串每个字符的后4位（ASCII码）应依次是（十进制）：15 、0 、5 、11 、13 、 1

所以答案有很多。我**个人的答案是：O@EKMA**





1. 测试，成功过关！



* 第六关

（同学们自己好好思考。此行红字请删除）

**五、实验总结与体会**

（请同学们认真记录你在实验中的疑点难点以及解决方案，记录你在本实验收获的内容以及实验心得。此行红字请删除）

|  |
| --- |
| **指导教师批阅意见：**  **成绩评定：**  指导教师签字：刘刚    2016年 4月21日 |
| 备注： |