**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： 计算机科学与技术**

**指导教师： 王 毅**

**报告人： 钟善扬 学号： 2017303031 班级： 02**

**实验时间： 2020年04月22日**

**实验报告提交时间： 2020年06月10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\_64和主函数所在的源程序bomb\_64.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：**   1. 将bomb.txt文件的后缀改为exe，并利用objdump反汇编，将结果输出在1.txt中。      1. GDB调试     <Phase 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0x401af8。故下面进入gdb调试，然后查看0x401af8该地址对应的数据。    由上图可知，0x401af8存储了一个字符串“Science isn’t about why, it’s about why not?”是一句有意义的字符串，很可能就是本题答案，所以尝试该答案：  用chmod +x bomb命令增加当前用户对bomb文件的执行权限    由上图可知本题答案正是“Science isn’t about why, it’s about why not?”  第一关通过。  <Phase 2>    从400ea7中的<read\_six\_numbers>可以看出phase\_2需要读入6个数，然后在400eba-400ed4的循环中比较%rbp和%rbp+0xc，%rbp+0x4和%rbp+0x10，%rbp+0x8和%rbp+0x14，如果出现不相等则爆炸，如果都相等，到400ed6中判断%r12d是否为0，若为0则爆炸，否则通关。所以9 9 6 9 9 6满足以上条件，进行测试，结果如下：    第二关通过。  <Phase 3>    首先看400f07，发现0x401ebe的值被当作参数传入，所以查看其内容：    发现是需要读两个数。从图中后面的分析可知第一个数不能超过7，否则直接爆炸。然后来到jmpq跳转指令，根据输入的第一个数跳到一个地址对%eax赋值。通过gdb调试查看跳转的地址：    建立第一个输入和跳转到的地址想要给%eax赋的值之间的映射，如下图：    最后判断第二个输入和%eax是否相等，相等则通关。  所以，输入0 535就应该可以通关：    第三关通过。  <Phase 4>  首先看400fca，发现0x401ec1的值被当作参数传入，所以查看其内容：    发现是需要读一个数。      由上图分析可知，我们需要输入一个数，将这个数传进func4的话会返回55；而func4(x)=func4(x-1)+func4(x-2)(x>1)，即求斐波那契数列的第n项的值。所以问题转化成：斐波那契数列第几项的值为55？答案是第9项。    第四关通过。  <Phase 5>  首先看401010，发现0x401ebe的值被当作参数传入，所以查看其内容：    发现是需要读两个数。    由上图的分析可知需要知道0x401ba0开始预存的数组，用gdb查看：    从图中可知数组内容为（下标为0开始）：  {a，2，e，7，8，c，f，b，0，4，1，d，3，9，6}  即：  {10，2，14，7，8，12，15，11，0，4，1，13，3，9，6}  从15回溯12次就可以得到第一个数的下标。(次数，值，下标）  （12，15，6）  （11，6，14）  （10，14，2）  （9，2，1）  （8，1，10）  （7，10，0）  （6，0，8）  （5，8，4）  （4，4，9）  （3，9，13）  （2，13，11）  （1，11，7）  所以第一个输入为7，求和得93，所以第二个输入为93.    第五关通过。  <Phase 6>    发现有注释node0，说明可能含有链表结构，所以设置断点查看：    果然是链表  下面再看看fun6做了什么：    观察每个节点的next指针变化，容易发现链表的进行了一次由大到小的排序，（826,782,673,666,600,529,488,374,370,286）我们输入的数是一直保存在第一个结点之中    观察这段代码可以发现%rax指向的是第四个结点，而该结点的值必须等于输入，也就是说我们输入的数必须排在原来的链表中第四大的位置，而从该链表中原有的数字（826,782,673,600,529,488,374,370,286,0）可以看出，输入的数字必须在600-673之间，刚好这样插进第四大的位置被%rax指着,否则%rax的值要么是673（输入太大）要么就是600（输入太小），不会等于输入值。数字666符合分析，结果如下：    第六关通过。  <Secret Phase>   1. 仔细观察发现在每道关卡之后都有一次对于phase\_defused的调用，查看该关卡可以发现有跳转到Secret Phase的代码。     设置断点调用gdb查看内存0x 401ec4中的值    可看出是要输一个整数和一个字符串。查看0x603030中的值，结果如下，值为9，所以可能是在第四关输入‘9’之后再输入另一个字符串就能触发Secret Phase。查看内存0x401eca中的值如下图所示，所以尝试在第四关输入正确答案之后再输入该字符串            3、107？ |
| 1. **实验结论：**   第一关答案是：Science isn't about why, it's about why not?  第二关答案是满足前三位和后三位对应相等且和不为0的6个数，如：9 9 9 9 9 9  第三关答案通过gdb建立映射，选取合适“键值对”，如：2 214  第四关答案是：9  第五关答案是：7 93  第六关答案是在600~673范围的整数  Secret Phase答案是107 |
| **六、心得体会：**  这次实验没有想象中那么难而且充满乐趣，其实前面几道题目一看就知道要干嘛了，就是第五关最难了，因为第五关的操作太冷门而且平时我们也不那么干吧。通过本次实验，我更加熟练地掌握了gdb调试，学会了使用断点和查看内存或寄存器中的值来进行反汇编，对第三章的各种常见汇编语言比如跳转指令、函数调用等内容有了更加深入的掌握，对计算机系统如何从高级语言到可执行程序的过程有了更深刻的认识。 |

|  |
| --- |
| 指导教师批阅意见：  成绩评定：  指导教师签字：王毅  2020年 6月5 日 |
| 备注： |