**中国海洋大学计算机科学与技术系**

**实验报告**

**姓名：** 陈扬 **年级：**2017级  **专业：政治学与行政学**

**科目：**汇编语言 **题目：** 熟悉指令系统

**实验时间:** 2018年10月9日 **实验成绩: 实验教师:孙鑫**

**一、实验目的：**

1. 深入了解c语言代码执行的原理

2.能熟练运用GDB,反汇编,idea

**二、实验环境(硬件或软件)：**

C, 汇编；实验环境：Linux i386（32bits）

**三、实验原理：**

在本实验中，你需要使用课程所学知识拆除一个“binary bombs”，从而加强对程序的机器级表示、汇编语言、调试器和逆向工程等方面知识的理解和掌握。 一个“binary bombs”（二进制炸弹，下文将简称为炸弹）是一个Linux可执行程序，包含了6个阶段（或层次、关卡）以及1个隐藏阶段。炸弹运行的每个阶段要求你输入一个特定字符串，你的输入符合程序预期的输入，该阶段的炸弹就被拆除引信即解除了，否则炸弹“爆炸”打印输出 “BOOM!!!”并转到下一阶段等待你的输入。实验的目标是拆除尽可能多阶段的炸弹。

每个炸弹阶段考察了机器级程序语言的一个不同方面：

阶段1：字符串比较

阶段2：循环

阶段3：条件/分支

阶段4：递归调用和栈

阶段5：指针

阶段6：链表/指针/结构

隐藏阶段只有当你在阶段4的拆解字符串后再附加一特定字符串后才会出现（作为最后一个阶段）。

为完成二进制炸弹拆除任务，你需要使用gdb调试器和objdump来反汇编炸弹的可执行文件并单步跟踪每一阶段的机器代码（例如可在每一阶段的开始代码前和引爆炸弹的函数前设置断点），从中理解每一汇编语言代码的行为和作用，进而设法推断拆除炸弹所需的目标字符串。

**四、实验内容(实验步骤或者程序编写)：**

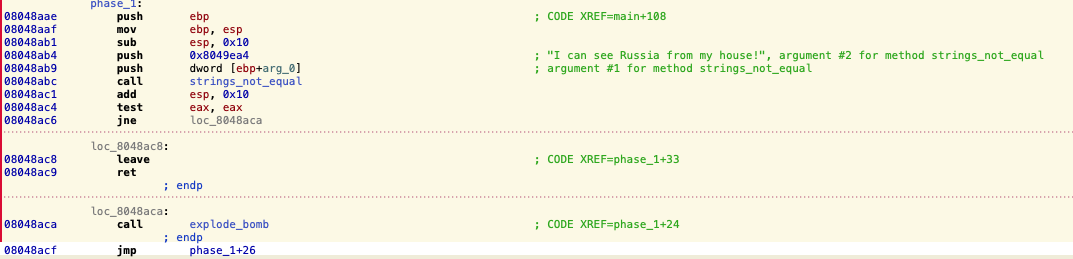
bomb：二进制可执行bomb程序

bomb.c：bomb程序的main函数

运行./bomb可执行程序时可使用0或1个命令行参数（详见bomb.c源文件中的main()函数）。如果运行时不指定参数，则该程序打印出欢迎信息后，期望你按行输入每一阶段用来拆除炸弹的字符串，根据你当前输入的字符串决定你是通过相应阶段还是引爆该阶段的炸弹。

你也可将拆除每一阶段炸弹的字符串按行组织在一个文本文件中（就像你需要提交的实验结果文件，见下说明），然后作为运行程序时的唯一一个命令行参数传给程序，程序依次检查对应每一阶段的字符串来决定炸弹拆除成败。

**炸弹一:**



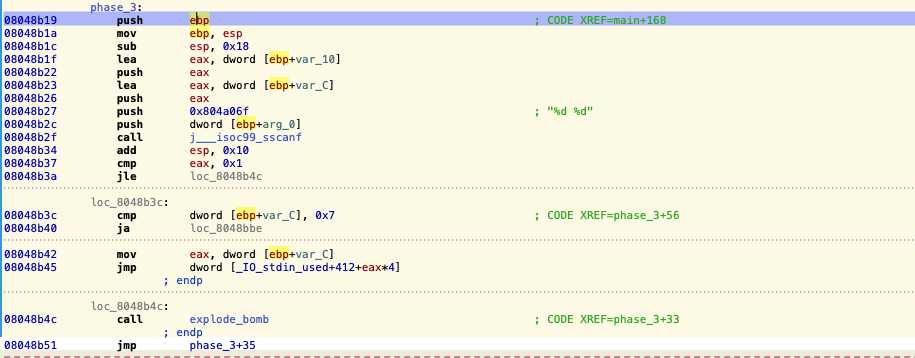
密码:I can see Russia from my house

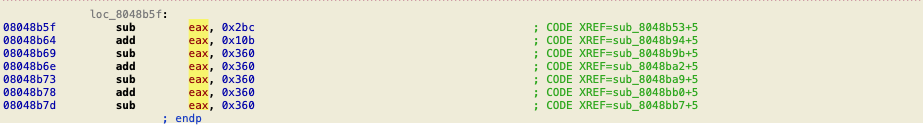
**炸弹2:**



密码是一段等比数列:1,2,4,8,16,32

**炸弹3:**

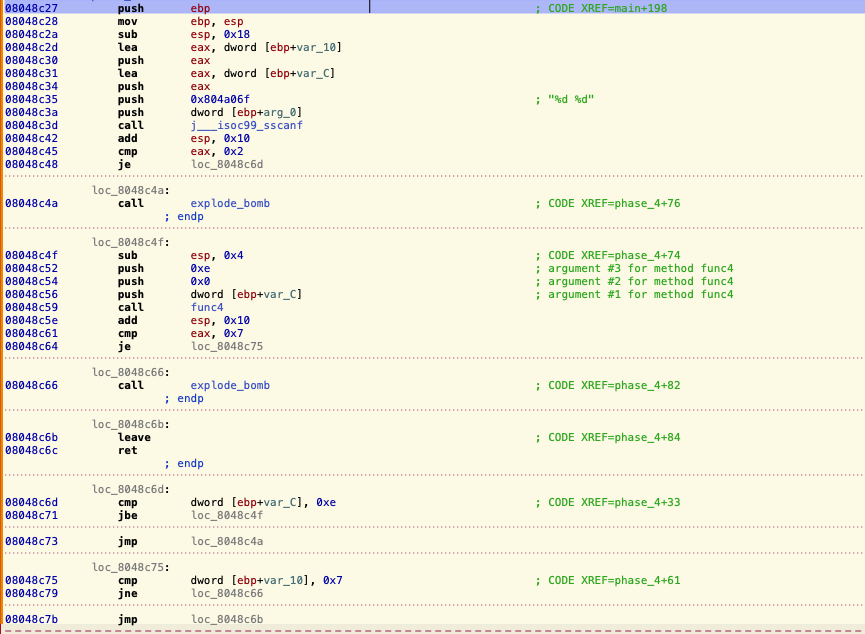




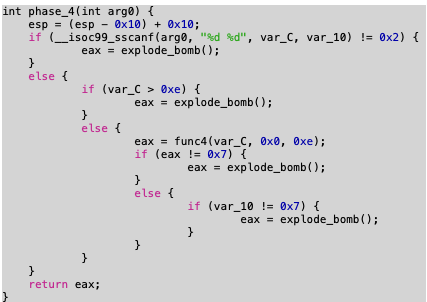
分支结构,我是从0开始的,所以我的密码是for i=0 to 7 do:ans+=a[i];

密码是:0 -298

**炸弹4:**

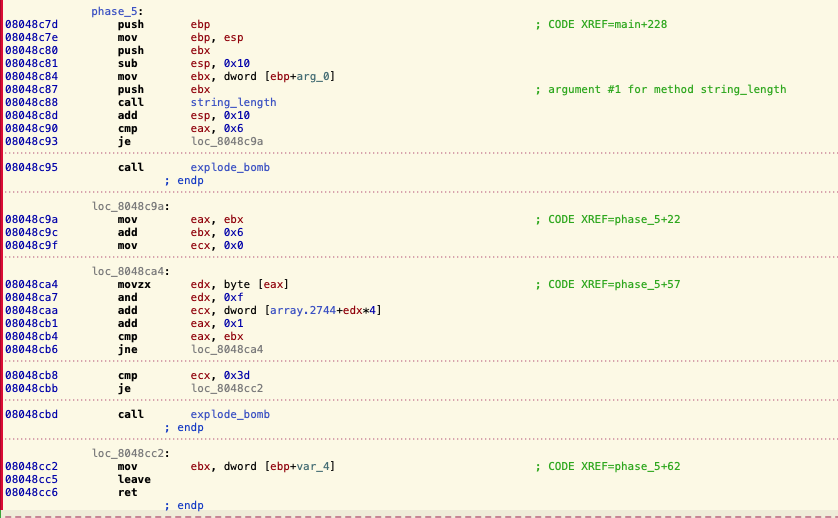


因为有一个递归:



反编译成c,我们发现第一个输入要小于14,第二个输入等于7,会根据第一个输入算出一个值,判断是否等于7,因为只有14种可能,暴力得出密码:14 7

炸弹5:



同理第二题:

密码:DrEvil

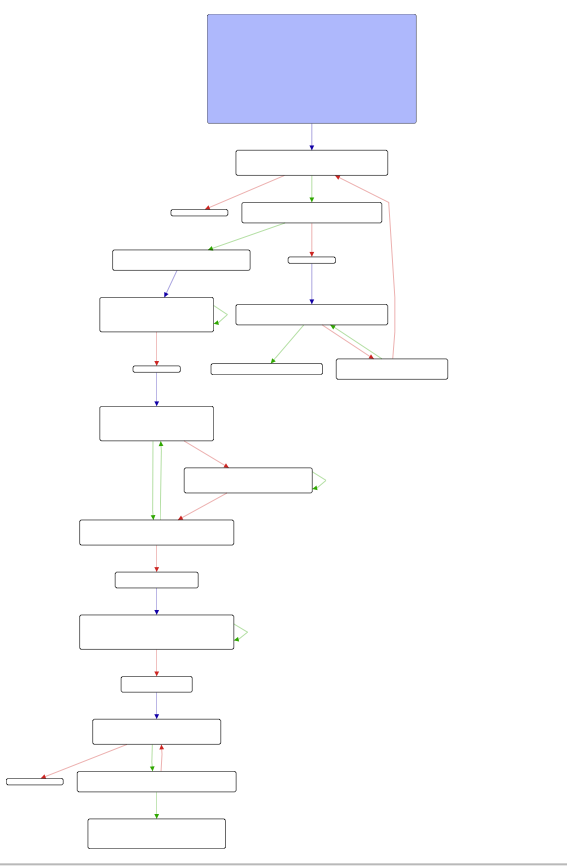
隐藏关卡:

DrEvil:



1001标准答案

1. 炸弹6:这个炸弹爆炸难,调了一个通宵08048cc7         push       ebp                                                 ; CODE XREF=main+258
2. 08048cc8         mov        ebp, esp
3. 08048cca         push       esi
4. 08048ccb         push       ebx
5. 08048ccc         sub        esp, 0x38
6. 08048ccf         lea        eax, dword [ebp+var\_20]
7. 08048cd2         push       eax                                                 ; argument #2 **for** method read\_six\_numbers
8. 08048cd3         push       dword [ebp+arg\_0]                                   ; argument #1 **for** method read\_six\_numbers
9. 08048cd6         call       read\_six\_numbers
10. 08048cdb         add        esp, 0x10
11. 08048cde         mov        esi, 0x0
12. 08048ce3         jmp        loc\_8048d01
14. loc\_8048ce5:
15. 08048ce5         add        esi, 0x1                                            ; CODE XREF=phase\_6+68, phase\_6+75
16. 08048ce8         cmp        esi, 0x6
17. 08048ceb         je         loc\_8048d1b
19. 08048ced         mov        ebx, esi
21. loc\_8048cef:
22. 08048cef         mov        eax, dword [ebp+ebx\*4+var\_20]                       ; CODE XREF=phase\_6+56
23. 08048cf3         cmp        dword [ebp+esi\*4+var\_24], eax
24. 08048cf7         je         loc\_8048d14
26. loc\_8048cf9:
27. 08048cf9         add        ebx, 0x1                                            ; CODE XREF=phase\_6+82
28. 08048cfc         cmp        ebx, 0x5
29. 08048cff         jle        loc\_8048cef
31. loc\_8048d01:
32. 08048d01         mov        eax, dword [ebp+esi\*4+var\_20]                       ; CODE XREF=phase\_6+28
33. 08048d05         sub        eax, 0x1
34. 08048d08         cmp        eax, 0x5
35. 08048d0b         jbe        loc\_8048ce5
37. 08048d0d         call       explode\_bomb
38. ; endp
39. 08048d12         jmp        phase\_6+30
41. loc\_8048d14:
42. 08048d14         call       explode\_bomb                                        ; CODE XREF=phase\_6+48
43. ; endp
44. 08048d19         jmp        phase\_6+50
46. loc\_8048d1b:
47. 08048d1b         lea        eax, dword [ebp+var\_20]                             ; CODE XREF=phase\_6+36
48. 08048d1e         lea        ebx, dword [ebp+var\_8]
49. 08048d21         mov        ecx, 0x7
51. loc\_8048d26:
52. 08048d26         mov        edx, ecx                                            ; CODE XREF=phase\_6+106
53. 08048d28         sub        edx, dword [eax]
54. 08048d2a         mov        dword [eax], edx
55. 08048d2c         add        eax, 0x4
56. 08048d2f         cmp        ebx, eax
57. 08048d31         jne        loc\_8048d26
59. 08048d33         mov        ebx, 0x0
61. loc\_8048d38:
62. 08048d38         mov        esi, ebx                                            ; CODE XREF=phase\_6+154
63. 08048d3a         mov        ecx, dword [ebp+ebx\*4+var\_20]
64. 08048d3e         mov        eax, 0x1
65. 08048d43         mov        edx, 0x804c13c
66. 08048d48         cmp        ecx, 0x1
67. 08048d4b         jle        loc\_8048d57
69. loc\_8048d4d:
70. 08048d4d         mov        edx, dword [edx+8]                                  ; CODE XREF=phase\_6+142
71. 08048d50         add        eax, 0x1
72. 08048d53         cmp        eax, ecx
73. 08048d55         jne        loc\_8048d4d
75. loc\_8048d57:
76. 08048d57         mov        dword [ebp+esi\*4+var\_38], edx                       ; CODE XREF=phase\_6+132
77. 08048d5b         add        ebx, 0x1
78. 08048d5e         cmp        ebx, 0x6
79. 08048d61         jne        loc\_8048d38
81. 08048d63         mov        ebx, dword [ebp+var\_38]
82. 08048d66         mov        ecx, ebx
83. 08048d68         mov        eax, 0x1
85. loc\_8048d6d:
86. 08048d6d         mov        edx, dword [ebp+eax\*4+var\_38]                       ; CODE XREF=phase\_6+181
87. 08048d71         mov        dword [ecx+8], edx
88. 08048d74         add        eax, 0x1
89. 08048d77         mov        ecx, edx
90. 08048d79         cmp        eax, 0x6
91. 08048d7c         jne        loc\_8048d6d
93. 08048d7e         mov        dword [edx+8], 0x0
94. 08048d85         mov        esi, 0x5
95. 08048d8a         jmp        loc\_8048d94
97. loc\_8048d8c:
98. 08048d8c         mov        ebx, dword [ebx+8]                                  ; CODE XREF=phase\_6+212, phase\_6+219
99. 08048d8f         sub        esi, 0x1
100. 08048d92         je         loc\_8048da4
102. loc\_8048d94:
103. 08048d94         mov        eax, dword [ebx+8]                                  ; CODE XREF=phase\_6+195
104. 08048d97         mov        eax, dword [eax]
105. 08048d99         cmp        dword [ebx], eax
106. 08048d9b         jge        loc\_8048d8c
108. 08048d9d         call       explode\_bomb
109. ; endp
110. 08048da2         jmp        phase\_6+197
112. loc\_8048da4:
113. 08048da4         lea        esp, dword [ebp+var\_8]                              ; CODE XREF=phase\_6+203
114. 08048da7         pop        ebx
115. 08048da8         pop        esi
116. 08048da9         pop        ebp
117. 08048daa         ret
118. ; endp



我参考了:http://zpalexander.com/binary-bomb-lab-phase-6/

就是先得出一个序列,再根据序列排序

我的密码是:1 3 4 6 2 5

总结:累的很,学的很开心.

