二进制拆炸弹bomb实验第一弹

原创

gjzjmmd 于 2015-06-08 16:29:12 发布 14394 收藏 27
文章标签： gdb linux 汇编 bomb拆弹实验 终端

著名的bomb拆炸弹实验：

程序运行在linux环境中。程序运行中有6个关卡（6个phase），每个phase需要用户在终端上输入特定的字符或者数字才能通关，否则会引爆炸弹！那么如何才能知道输入什么内容呢？这需要你使用gdb工具反汇编出汇编代码，结合c语言文件找到每个关卡的入口函数。然后分析汇编代码，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通关。

实验过程很艰辛,三言两语道不尽呐！好吧，那就一关一关来拆。看到两千行汇编代码头都晕啊！！！

首先，bomb.c中的c代码好抽象，根本得不到什么信息，只看到有六个phase的函数。根据提示，要将抽象的c代码反汇编，我先打开了我的Ubuntu，我把整个LAB3文件放在fsj文件下，进入终端，输入objdump -d bomb > fsj.txt将汇编代码输出到与bomb同目录的一个自动生成的叫fsj.txt的文件中。这样就可以看到fsj.txt中有约两千行汇编代码，总的浏览一下，大概的意思是将一个一个的函数分开，明显可以看到六个phase函数。

Bomb第一弹正式开始。分析一下代码吧！

08048f61 <phase_1>://We have to stand with our North Korean allies.

```
 8048f61:  55                   push   %ebp
 8048f62:  89e5                 mov    %esp,%ebp
 8048f64:  83ec 18              sub    $0x18,%esp //开辟空间
 8048f67:  c744 24 04 5c a1 04 movl  $0x804a15c,0x4(%esp)
 8048f6e:  08
 8048f6f:  8b45 08              mov    0x8(%ebp),%eax
 8048f72:  8904 24              mov    %eax,(%esp)
 8048f75:  e831 00 00 00        call  8048fab <strings_not_equal>
 8048f7a:  85c0                 test   %eax,%eax //eax要为0
 8048f7c:  7405                 je     8048f83<phase_1+0x22>
 8048f7e:  e84e 01 00 00        call   80490d1 <explode_bomb>  //很贴心地为我们写好了地址所指的函数
 8048f83:  c9                   leave
 8048f84:  c3                   ret
 8048f85:  90                   nop
 8048f86:  90                   nop
```

8048f87: 90 nop

8048f88: 90 nop

8048f89: 90 nop

8048f8a: 90 nop

8048f8b: 90 nop

8048f8c: 90 nop

8048f8d: 90 nop

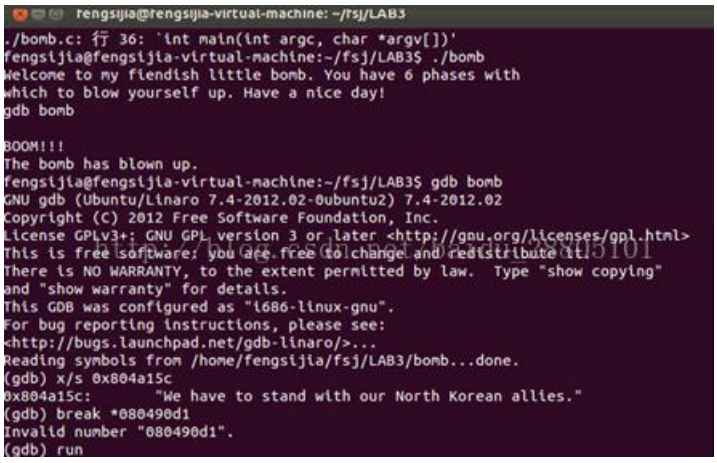
8048f8e: 90 nop

8048f8f: 90 nop

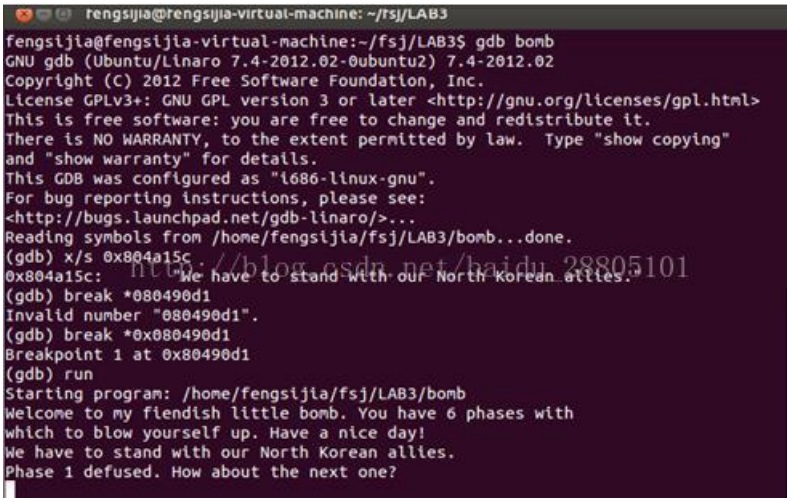
读了一遍代码后，应该有点思路了，从call 8048fab <strings_not_equal>这里在比较字符串应该可以猜到第一关要我们输入的是一串字符，如果输入的字符和它要求 的字符串一样的话，就ok,不然就会bomb。

那么它想要的字符串到底是什么呢，红色标注的代码值得注意，movl $0x804a15c,0x4(%esp)，这里有一串立即数很引人注意啊，难道这里有什么东西，我就尝试性地用gdb调试了一下：

进入bomb的上一层目录就是LAB3后，输入 gdb bomb，就可以调试了，以防万一，我先在进入bomb函数前设置断点，先找到这个玩意儿080490d1<explode_bomb>，输入break *0x080490d1,这样它就永远不会爆炸了，哈哈！（当然了，第一次我还不怎么会用gdb，就直接./bomb，然后很兴奋地乱输入一串字符，就被炸死了，看：）。



查看一下0x080490d1的地方放着什么东西，输入指令：x/s 0x804a15c 看到有一串字符！！！！We have to standwith our North Korean allies.哈哈，这个时候就猜想这就是我要输入的东西。验证一下是不是，我就run了一下，看截图1：



真的是！程序没有到断点那里停下，就说明已经通过了第一关。

第一关相对后面的关卡来说真的很easy了，但是用逆向思维去思考代码的时候，我也是用半蒙半猜的方法去解决的，然后用gdb调试去验证我的猜想正不正确，在当结果被验证是正确的时候，再结合答案去推敲汇编代码，才能更好的理解有些代码的意思。比如这里mov 0x8(%ebp),%eax mov %eax,(%esp)这两条语句是把我输入的参数给eax，再给esp; call 8048fab <strings_not_equal>

test %eax,%eax //eax要为0，这里eax是函数调用的返回值。这些都是我得出答案之后回过头去才理解的汇编代码。

趟着石头过河，形容我做这个实验最形象不过了，正向思维和逆向思维结合，才能更好更高效地得到答案和理解汇编代码。不得不说，gdb的调试功能很强大，这一点在后面的实验中体现的更加淋漓尽致。

程序运行在linux环境中。程序运行中有6个关卡（6个phase），每个phase需要用户在终端上输入特定的字符或者数字才能通关，否则会引爆炸弹！那么如何才能知道输入什么内容呢？这需要你使用gdb工具反汇编出汇编代码，结合c语言文件找到每个关卡的入口函数。然后分析汇编代码，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通关。