bomblab 实验说明

1. 简介

本实验通过要求你使用课程所学知识拆除一个“binary bombs”来增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原理与技能的掌握。 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）。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!!!"。实验的目标是拆除尽可能多的炸弹层次。

每个炸弹阶段考察了机器级程序语言的一个不同方面，难度逐级递增：

- 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

另外还有一个隐藏阶段，只有当你在第 4 阶段的解后附加一特定字符串后才会出现。

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跟踪调试每一阶段的机器代码，从中理解每一汇编语言代码的行为或作用，进而设法推断拆除炸弹所需的目标字符串。比如在每一阶段的开始代码前和引爆炸弹的函数前设置断点。

实验语言：C；实验环境：Linux

2. 实验步骤

2.1. 第一步：获取 bomb

在浏览器中打开 http://www.yiligong.org:8900，在二进制炸弹请求表格中输入你的学号和邮箱地址，点击 Submit 按钮。服务器会构造属于你的炸弹，并以 tar 文件的形式 bombX.tar 返回给你，其中 X 是一个你的 bomb 的唯一标识。

解压该 tar 文件（tar -xvf bombX.tar）得到一个目录./bombX，其中包含如下文件：

- README：标识该 bomb 和所有者。
- bomb：bomb 的可执行程序。
- bomb.c：bomb 程序的 main 函数。

2.2. 第二步：拆除 bomb

本实验的任务就是拆除炸弹。一定要在指定的虚拟机上完成作业，在其他的环境上运行有可能导致失败。

运行./bomb 可执行程序需要 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炸弹爆炸导致任务失败。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，然后作为运行程序时的唯一一个命令行参数传给程序，程序读入文件中的每一行直到遇到 EOF，再转到从 stdin 等待输入。这样对于你已经拆除的炸弹，就不用每次都重新输入，只用放进文件里即可。

前四个阶段每个 10 分，第五和第六阶段更难一些，每个 15 分，满分 70 分。每输入错误一次，炸弹爆炸，会扣除 0.5 分（最多扣除 20 分）。所以你必须小心！要学会单步跟踪调试汇编代码以及学会设置断点。你还要学会如何检查寄存器和内存状态。很好的使用调试器是你在未来的职业生涯中赚到更多 money 的一项重要技能。

2.3. 实验结果提交

这是一项独立实验，每个人单独完成。不需要单独提交最后的结果，bomb 程序会自动发送结果到服务器（请确保你调试执行的虚拟机能够连接上网！），可以在 http://54.187.73.103:8900/scoreboard 查看所有人的成绩结果。

3. 提示！

下面简要说明完成本实验所需要的一些实验工具：

gdb

为了从二进制可执行程序"./bomb"中找出触发 bomb 爆炸的条件，可使用 gdb 来帮助对程序的分析。GDB 是 GNU 开源组织发布的一个强大的交互式程序调试工具。一般来说，GDB 主要帮忙你完成下面几方面的功能（更详细描述可参看 GDB 文档和相关资料）：

1. 装载、启动被调试的程序。
2. 让被调试的程序在你指定的调试断点处中断执行，方便查看程序变量、寄存器、栈内容等运行现场数据。
3. 动态改变程序的执行环境，如修改变量的值。

gdb 相关资料：

http://beej.us/guide/bggdb/

https://www.gnu.org/software/gdb/

objdump –t

该命令可以打印出 bomb 的符号表。符号表包含了 bomb 中所有函数、全局变量的名称和存储地址。你可以通过查看函数名得到一些目标程序的信息。

objdump –d

该命令可用来对 bomb 中的二进制代码进行反汇编。通过阅读汇编源代码可以发现 bomb 是如何运行的。但是，objdump –d 不能告诉你 bomb 的所有信息，例如一个调用 sscanf 函数的语句可能显示为： 8048c36: e8 99 fc ff ff call 80488d4 <_init+0x1a0> ，你还需要 gdb 来帮助你确定这个语句的具体功能。

strings

该命令可以显示二进制程序中的所有可打印字符串。

实验步骤提示

下面以第一阶段(第一关)为例介绍实验步骤:首先调用”objdump -d bomb > bomb_disas.txt”对 bomb 进行反汇编并将汇编源代码输出到“boomb_disas.txt”文本文件中。查看该汇编源代码文件，我们可以在 main 函数中找到如下语句，从而得知第一关的处理程序包含在“main()”函数所调用的函数“phase_1()”中,判断的过程可以参照 bomb.c 文件源码。汇编代码中地址 400ca7 处调用了 phase_1 函数，

```
400ca0:        48 8b 45 f8              mov    -0x8(%rbp),%rax
400ca4:        48 89 c7                 mov    %rax,%rdi
400ca7:        e8 a8 00 00 00           callq  400d54 <phase_1>
400cac:        bf e8 19 40 00           mov    $0x4019e8,%edi
400cb1:        e8 ca fc ff ff           callq  400980 <puts@plt>
400cb6:        e8 8d 09 00 00           callq  401648 <read_line>
400cbb:        48 89 45 f8              mov    %rax,-0x8(%rbp)
400cbf:        48 8b 45 f8              mov    -0x8(%rbp),%rax
400cc3:        48 89 c7                 mov    %rax,%rdi
400cc6:        e8 b1 00 00 00           callq  400d7c <phase_2>
400ccb:        bf 11 1a 40 00           mov    $0x401a11,%edi
400cd0:        e8 ab fc ff ff           callq  400980 <puts@plt>
400cd5:        e8 6e 09 00 00           callq  401648 <read_line>
400cda:        48 89 45 f8              mov    %rax,-0x8(%rbp)
400cde:        48 8b 45 f8              mov    -0x8(%rbp),%rax
400ce2:        48 89 c7                 mov    %rax,%rdi
```

我们在反汇编代码中寻找这个子函数 phase_1：

```
0000000000400f57 <phase_1>:
  400f57:       48 83 ec 08              sub    $0x8,%rsp
  400f5b:       be 2d 19 40 00           mov    $0x40192d,%esi
  400f60:       e8 b7 01 00 00           callq  40111c <strings_not_equal>
  400f65:       85 c0                    test   %eax,%eax
  400f67:       74 05                    je     400f6e <phase_1+0x17>
  400f69:       e8 74 02 00 00           callq  4011e2 <explode_bomb>
  400f6e:       48 83 c4 08              add    $0x8,%rsp
  400f72:       c3                       retq

0000000000400f73 <phase_4>:
```

可以看到这个子函数比较小，只有几行汇编代码，可以进行简单阅读（如果汇编代码较多，不建议逐句阅读,而是借用 gdb 调试工具进行辅助):我们看到(教科书中已经提到过调用函数的过程),……,还调用了 string_not_equal 函数，接着测试%eax 是否为零，如果是就跳转到+0x26 出否者就调用 explode_boPRINT mb，可以判定这是一个判断两个字符串是否相等的过程，使用 gdb 调试 bomb 二进制文件：gdb bomb 后，输入(gdb)print (char *)ox40192d 输出是

```
Reading symbols from /home/allen/work/bomb...done.
(gdb) print (char *)0x40192d
$1 = 0x40192d "Public speaking is very easy."
(gdb)
```

于是去设置断点去监测这个答案是否正确，我们在 explod_bomb 处设置断点：(gdb)break *0x400f69 然后(gdb)run 运行按照提示输入这个字符串

```
(gdb) print (char *)0x40192d
$1 = 0x40192d "Public speaking is very easy."
(gdb) break *0x400f69
Breakpoint 1 at 0x400f69: file phases.c, line 26.
(gdb) r
Starting program: /home/allen/work/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Public speaking is very easy.
Phase 1 defused. How about the next one?
```

第一关解除。