ICS-LAB3
BinaryBomb 二进制炸弹

哈尔滨工业大学
计算机科学与技术学院

2021年4月

一、实验基本信息

- **实验类型：验证型实验**
- **实验目的**
  - 熟练掌握计算机系统的ISA指令系统与寻址方式
  - 熟练掌握Linux下调试器的反汇编调试跟踪分析机器语言的方法
  - 增强对程序机器级表示、汇编语言、调试器和逆向工程等的理解
- **实验指导教师**
  - 任课教师：
  - 实验室教师：
  - TA：
- **实验班级、人数与分组**
  - 1903007、1903008、 1903009 一人一组

- **实验学时：3，15:45-18:15**
- **实验学分：20，本次实验按100分计算，折合成总成绩的6分。**
- **实验地点：G712、G710**
- **实验环境与工具：**
  - X64 CPU；2GHz；2G RAM；256GHD Disk 以上
  - Windows7 64位以上；VirtualBox/Vmware 11以上；Ubuntu 16.04 LTS 64位/优麒麟 64位；
  - GDB/OBJDUMP；EDB；KDD等
- **学生实验准备：禁止准备不合格的学生做实验**
  - 个人笔记本电脑
  - 实验环境与工具所列明软件
  - 参考手册: Linux环境下的命令；GCC手册；GDB手册
  - http://docs.huihoo.com/c/linux-c-programming/ C汇编Linux手册
  - http://csapp.cs.cmu.edu/3e/labs.html CMU的实验参考
  - http://www.linuxidc.com/ http://cn.ubuntu.com/ http://forum.ubuntu.org.cn/

二、实验要求

- **学生应穿鞋套进入实验室**
- **进入实验室后在签到簿中签字**
- **实验安全与注意事项**
  - 禁止使用笔记本电脑以外的设备
  - 学行生不得自行开关空调、投影仪
  - 学生不得自打开窗户
  - 不得使用实验室内的其他实验箱、示波器、导线、工具、遥控器等
  - 认真阅读消防安全撤离路线
  - 突发事件处理：第一时间告知教师，同时关闭电源插排开关。
- **遵守学生实验守则，爱护实验设备，遵守操作规程，精心操作，注意安全，严禁乱拆乱动。**
- **实验结束后要及时关掉电源，对所用实验设备进行整理，设备摆放和状态恢复到原始状态。**
- **桌面整洁、椅子归位，经实验指导教师允许后方可离开**

三、实验预习

- 上实验课前，必须认真预习实验指导书（PPT或PDF）
- 了解实验的目的、实验环境与软硬件工具、实验操作步骤，复习与实验有关的理论知识。
- 请写出C语言下包含字符串比较、循环、分支（含switch）、函数调用、递归、指针、结构、链表等的例子程序sample.c。
- 生成执行程序sample.out。
- 用gcc –S或CodeBlocks或GDB或OBJDUMP等，反汇编，比较。
- 列出每一部分的C语言对应的汇编语言。
- 修改编译选项-O (缺省2)、O0、O1、O2、O3，-m32/m64。再次查看生成的汇编语言与原来的区别。
- 注意O1之后无栈帧，EBP做别的用途。-fno-omit-frame-pointer加上栈指针。
- GDB命令详解 –tui模式 ^XA切换 layout改变等等
- 有目的地学习: 看VS的功能GDB命令用什么？

四、实验内容与步骤

- **1.环境建立**
  - Windows下Visual Studio 2010 64位
  - Windows下 OllyDbg（Windows下的破解神器OD）
  - Ubuntu下安装EDB（OD的Linux版---有源程序！）
  - Ubuntu下GDB调试环境、OBjDUMP；EDB
- **2.获得实验包**
  - 从实验教师处获得下 bomb.tar
  - 也可以从课程QQ群下载，也可以从其他同学处获取。
  - 每人的包都不同，一定要注意，
  - HIT与CMU的不同。CMU的网站只有一个炸弹。
- **3.GDB常用命令复习**
  - 设置断点、执行指令、看指令、看调用栈
  - 查看内存（全局变量）、看堆栈（局部变量、返回地址等）

4.sample.c的调试训练

- Windows下用VS调试，注意查看C对应ASM，调试方法

- Windows下用OD调试，注意查看C对应ASM，调试方法
  - VS编译去掉调试信息，包括符号表等，再用OD调试
- Linux下用CB调试，注意查看C对应ASM，调试方法

- 用GDB调试，注意查看C对应ASM，熟练掌握命令
  - CB编译去掉调试信息，包括符号表等，再用GDB调试
- 用EDB调试，注意查看C对应ASM，熟练掌握调试方法
  - CB编译去掉调试信息，包括符号表等，再用GDB调试

- 可以增加Ox与32/64位与栈帧选项后再次查看程序。

VS的使用

- **建解决方案+工程：连接选择子系统平台console**
  - 添加源程序.c或者.cpp
  - 选择64/32位，选择Debug/Release
- **编译链接**
  - 菜单->生成，可选择编译、生成，重新生成等
- **调试：调试菜单**
  - F10 单步调试，不进入子程序
  - F11 单步调试，进入子程序
  - F9 断点 F5 开始/继续 Shift+F5停止 不调试运行Ctrl+F5
  - Ctrl+F10 运行到光标处
  - Shift+F11 跳出子程序
- **调试：查看菜单**
  - 看寄存器 Alt+5 看内存Alt+6 看汇编语言程序 Alt+8
  - 鼠标右键可以增加内容，更改显示格式等等
  - 增加内存窗口：Ctrl+Alt+M+2 3 4

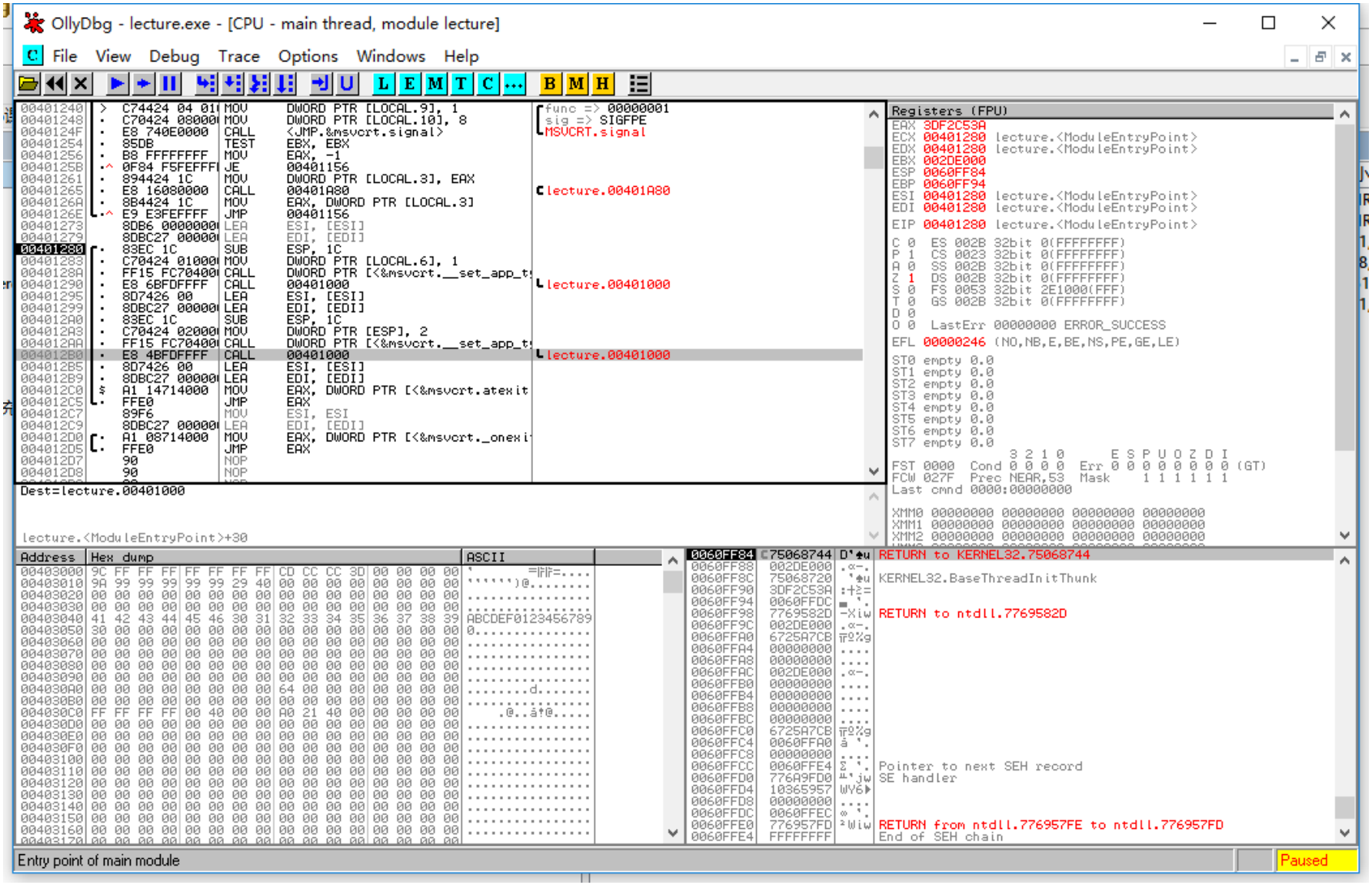
CB的使用

- **建立WorkPlace+Project工程：console/empty**
  - 添加源程序.c或者.cpp
  - 选择64/32位，选择Debug/Release　项目属性GCC编译里 -m32/-m64等
- **编译链接**
  - 菜单->生成，可选择编译^Shift+F9、生成^F9，重新生成^F11等
  - F2 查看log编译等输出的窗口
- **调试：调试菜单：GDB同Linux，可以代替Linux环境的学习**
  - 开始/继续 F8(无断点全速) ，Shift+F7 进入编译连接调试并到main。
  - F7下一行　Shift+F7单步入　^F7单步出
  - Alt+F7下一指令　ALT+Shift+F7 下一指令入
  - F5　断点　　F4　运行到光标处　　Shift+F8停止调试
  - Shift+F11 跳出子程序　　settings/debugger可改成Intel汇编
- **调试：查看菜单 Debugging Windows**
  - 寄存器　内存　汇编语言程序 监控变量窗口
  - 汇编语言是Linux的AT&T汇编

GDB的使用

- 可以先用readelf –a bomb 看一下执行文件的大致结构。第一条指令位置不是main
- (gdb)l (ist)　　(gdb) b main 断点　　(gdb) r 运行 c继续
- (gdb) disassemble main /m c和asm一起排列 /r 看16进制代码
- (gdb) x/15i main　用 x/i 可以查看指令
- (gdb) x/15 地址　用 x/ 可以查看数据　s字符串 x十六进制
- (gdb) display/10i main 命令自动显示当前正要执行汇编指令
- (gdb) ni; 单步执行下一条机器指令　　n单步执行一条C语句
  (gdb) si: 单步执行入一条机器指令　　s单步执行一条C语句
- (gdb) print　$eax $RIP　➔反人类！！
- (gdb) info reg　　info proc all看本进程的所有信息
- (gdb) layout prev | next | <layout_name> 改变布局　src/asm/split/regs
- (gdb) set disassembly-flavor　intel 改成Intel格式

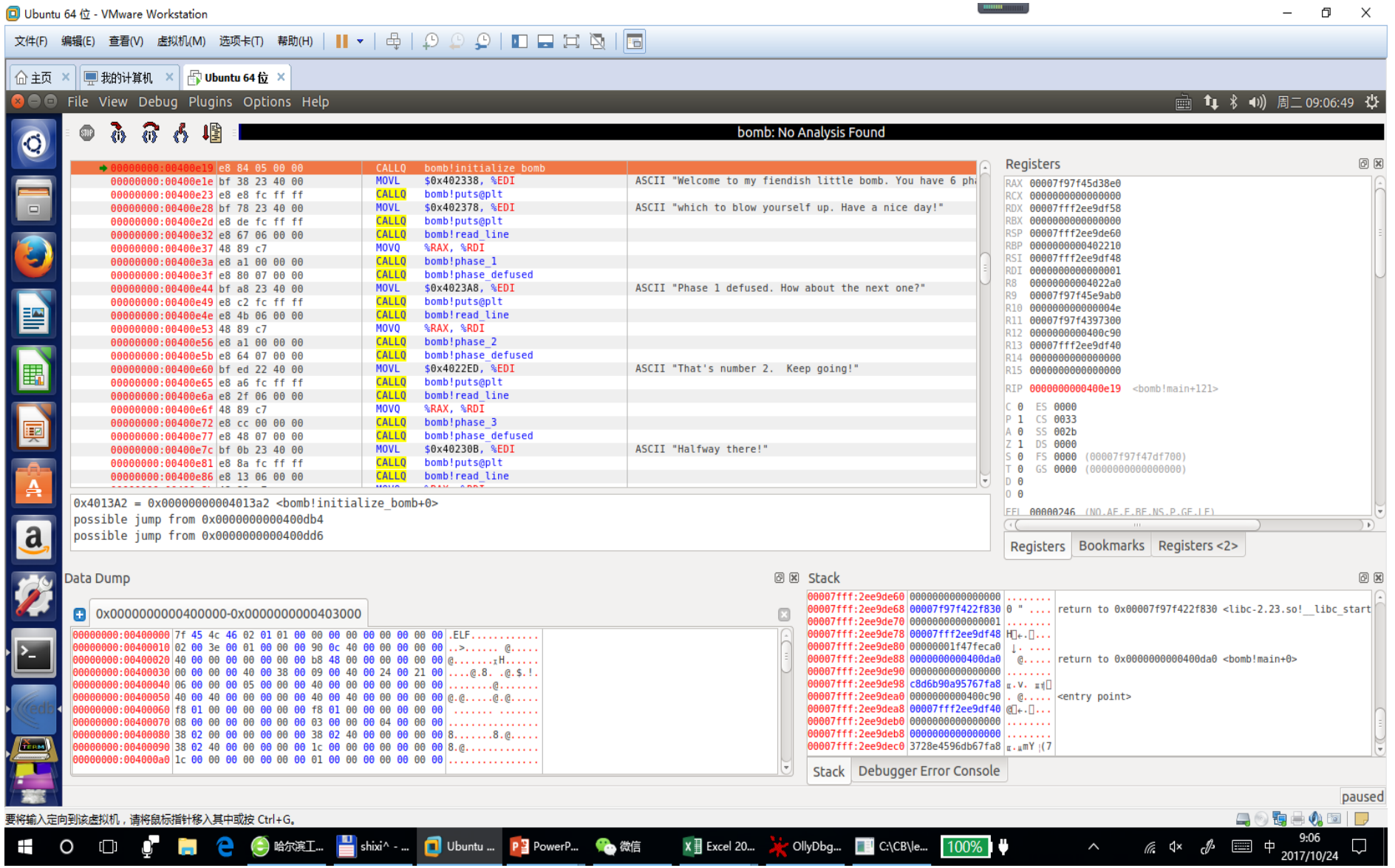
OllyDBG 破解神器（Windows）



EDB破解神器（Linux）

- **安装# install dependencies**
  - sudo apt-get install cmake build-essential libboost-dev \
  - libqt5xmlpatterns5-dev qtbase5-dev qt5-default \
  - libqt5svg5-dev libgraphviz-dev libcapstone-dev
- **安装# build and run edb**
  - sudo apt install git
  - git clone --recursive https://github.com/eteran/edb-debugger.git
  - cd edb-debugger
  - mkdir build
  - cd build
  - cmake .. 如出错 sudo apt-get install --reinstall pkg-config cmake-data
  - make
  - ./edb --run 执行程序

EDB破解神器（Linux）



5. 实验的二进制炸弹“Binary Bombs”程序简介

- 包含phase1~phase6共6个阶段，每个阶段考察机器级语言程序不同方面，难度递增
  - **阶段1：字符串比较**
  - **阶段2：循环**
  - **阶段3：条件/分支：含switch语句**
  - **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，第4阶段之后附加特定字符串后出现**
- 炸弹运行各阶段要求输入一个字符串，若输入符合程序预期，该阶段炸弹被“拆除”，否则“爆炸”。
- 你需要拆除尽可能多的炸弹。

6.分析实验代码框架

- 炸弹文件包：（每位同学不一样）
- $tar vxf bomb.tar
  - bomb： bomb的可执行程序。
  - bomb.c：bomb程序的main函数。
  - README

用文本编辑器打开看看就知道里面有什么了

- bomb：linux下可执行程序，需要0或1个命令行参数
  - 不带参数运行，输出欢迎信息后，期待你按行输入拆弹字符串，错误炸弹引爆退出，正确提示进入下一关。
  - 带参数运行，从拆弹者的密码文件中读取用户密码
- bomb.c：bomb主程序，帮助拆弹者了解代码框架，没有细节

7.拆弹过程

- 方法1：$./bomb
  - 根据提示，逐阶段手工输入拆弹字符串（见演示）
  - 较为繁琐，重复工作多
- 方法2：$./bomb ans.txt （推荐）
  - ans.txt为拆弹密码文本文件，名字可以自定义
    - 文本文件，每个拆弹字符串一行，回车结束，最多7行
    - 除此之外不要包含任何其它字符
  - 程序会检查每一阶段的拆弹密码字符串来决定炸弹拆除成败。

8.实验成果提交

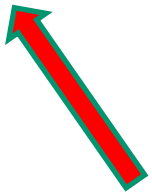
- **本次实验需要提交：实验报告和结果文件**
  - 结果文件：即上述的ans.txt，重新命名如下：
  - 实验报告：Word文档。在实验报告中，对你拆除了炸弹的每一道题，用文字详细描述分析求解过程。
  - 班为单位集中打包发送至指导教师
- **注意：及时记录每一步的地址、变量、函数、参数、数据结构、算法等等。以方便实验报告的撰写。**
- **最好截图：展示当前的代码、数据，并可手绘标注**

9.熟练掌握实验流程

实验步骤提示

- 直接运行bomb

```
bomb: command not found
acd@ubuntu:~/Lab1-3/bomblab/CS201401/U201414557$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
```



在这个位置初入阶段1的拆弹密码，如：This is a nice day.

- 你的工作：猜这个密码？
  - 下面以phase1为例介绍一下基本的实验步骤：

实验步骤演示

第一步：objdump –d bomb > asm.txt　　“>”:重定向

对bomb进行反汇编并将汇编代码输出到asm.txt中。

第二步：查看汇编源代码asm.txt文件，在main函数中找到如下语句

这里为phase1函数在main()函数中被调用的位置）：

```
/* Hmm...  Six phases must be more secure than one phase! */
input = read_line();             /* Get input                   */
phase_1(input);                  /* Run the phase               */
phase_defused();                 /* Drat!  They figured it out!
                                  * Let me know how they did it. */
printf("Phase 1 defused. How about the next one?\n");
```

```
106d:    e8 b7 06 00 00          callq  1729 <read_line>
1072:    48 89 c7                mov    %rax,%rdi
1075:    e8 fa 00 00 00          callq  1174 <phase_1>
107a:    e8 ee 07 00 00          callq  186d <phase_defused>
107f:    48 8d 3d 72 15 00 00    lea    0x1572(%rip),%rdi   # 25f8
1086:    e8 25 fd ff ff          callq  db0 <puts@plt>
```

- 第三步：在反汇编文件中继续查找phase_1的位置，如：

```
0000000000001174 <phase_1>:                    #注意%rdi没变，继承main
   1174:  48 83 ec 08              sub    $0x8,%rsp
   1178:  48 8d 35 d1 14 00 00     lea    0x14d1(%rip), %rsi      # 2650
   117f:  e8 32 04 00 00           callq  15b6 <strings_not_equal>
   1184:  85 c0                    test   %eax,%eax
   1186:  75 05                    jne    118d <phase_1+0x19>
   1188:  48 83 c4 08              add    $0x8,%rsp
   118c:  c3                       retq
   118d:  e8 30 05 00 00           callq  16c2 <explode_bomb>
   1192:  eb f4                    jmp    1188 <phase_1+0x14>
```

参数1

参数2

判断是否成功

- 很明显，这是64位程序，采用寄存器传递参数。
- strings_not_equal函数有两个参数：rdi是主程序main传来的唯一的参数---键盘输入或文件读取的字符串；rsi是一个全局变量。其地址为2650。
- 在GDB或EDB跟踪时可以直接到内存看这个变量的内容。
- 编译选项不同，看到的内容可能不一样 ===我们这儿应该是都一样的。

第四步：在main()函数的汇编代码中，可以进一步找到：

```
106d:        e8 b7 06 00 00          callq  1729 <read_line>
1072:        48 89 c7                mov    %rax,%rdi
```

%rax里存储的是调用read_line()函数返回值，也是用户输入的字符串首地址，推测拆弹密码字符串的存储地址为0x14d1(%rip)，2650 因为调用strings_not_equal前有语句：

```
1178:   48 8d 35 d1 14 00 00    lea    0x14d1(%rip), %rsi      # 2650
```

注意objdump看到的地址不是真正装载的内存地址，所以应进入GDB或EDB进行实际跟踪查看。

Gdb调试

0x2650　　%rsi里存放是是什么呢？

gdb查看这个地址存储的数据内容。具体过程如下：

第五步：执行：gdb bomb，显示如下：

GNU gdb (GDB) 7.2-ubuntu

Copyright (C) 2010 Free Software Foundation, Inc.

License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software: you are free to change and redistribute it.

There is NO WARRANTY, to the extent permitted by law. Type "show copying"

and "show warranty" for details.

This GDB was configured as "i686-linux-gnu".

For bug reporting instructions, please see:

<http://www.gnu.org/software/gdb/bugs/>...

./bomb/bomblab/src/bomb...done.

(gdb)

```
(gdb) b main      #在main函数的开始处设置断点
Breakpoint 1 at 0x101a: file bomb.c, line 45.
(gdb) r           #从gdb里运行bomb程序
Starting program:./bomb/bomblab/src/bomb
                  # 运行后，暂停在断点1处
Breakpoint 1, main (argc=1, argv=0xbffff3f4) at bomb.c:45
45          if (argc == 1) {
(gdb) n           #单步执行C指令
0x080489a8      45          if (argc == 1) {
(gdb) n
46                  infile = stdin;   #这里可以看到执行到哪一条C语句
(gdb) n
```

```
73          input = read_line();             /* Get input                */
(gdb) n          /*如果是命令行输入，这里输入你的拆弹字符串*/
74          phase_1(input);                  /* Run the phase           */
(gdb) x/10i  $rip
  0x555555555072 <main+88>:     mov    %rax,%rdi
  0x555555555075 <main+91>:     callq  0x555555555174 <phase_1>
(gdb) si
(gdb) si
(gdb) x/10i  $rip
=> 0x555555555174 <phase_1>:    sub    $0x8,%rsp
  0x555555555178 <phase_1+4>:   lea    0x14d1(%rip),%rsi    #0x555555556650
  0x55555555517f <phase_1+11>:  callq  0x5555555555b6 <strings_not_equal>
 (gdb) x/s 0x555555556650     #查看地址0x555555556650处字符串：
   0x555555556650:        "And they have no disregard for human life."
(gdb) q                    #退出gdb


#方法2
   EDB  用F8/F7进入调试即可，直接可以看到指针的内容。
```

拆弹现场演示

正确拆弹的另一个实例的显示（阶段1）：

```
acd@ubuntu:~/Lab1-3/bomblab/src$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
You can Russia from land here in Alaska.
Phase 1 defused. How about the next one?
```

拆弹失败的显示（阶段1）：

```
acd@ubuntu:~/Lab1-3/bomblab/src$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
You can russia from land here in Alaska.

BOOM!!!
The bomb has blown up.
```

拆弹现场演示

1）gdb的使用

$ gdb bomb

2）gdb常用指令

l：　（list）显式当前行的上、下若干行C语句的内容

b：　（breakpoint）设置断点

- 在main函数前设置断点：b main
- 在第5行程序前设置断点：b 5

r：　(run)执行，直到第一个断点处，若没有断点，就一直执行下去直至结束。

ni/stepi：　（next/step instructor）单步执行机器指令

n/step：　（next/step）单步执行C语句

x：显示内存内容

基本用法：以十六进制的形式显式0x804a0fc处开始的20个字节的内容：

(gdb) x/20x 0x804a0fc

0x804a0fc:	0x6d612049	0x73756a20	0x20612074	0x656e6572
0x804a10c:	0x65646167	0x636f6820	0x2079656b	0x2e6d6f6d
0x804a11c:	0x00000000	0x08048eb3	0x08048eac	0x08048eba
0x804a12c:	0x08048ec2	0x08048ec9	0x08048ed2	0x08048ed9
0x804a13c:	0x08048ee2	0x0000000a	0x00000002	0x0000000e

q：退出gdb，返回linux

gdb其他命令的用法详见使用手册，或联机help

10.结果提交

- **最终提交文件名**
  - 结果文件：即上述的ans.txt，重新命名如下：

    班级_学号.txt，如CS1807_1180307101.txt

    计算机 CS1807-CS1809　软工SE1801-SE1802　英才班YC1801
  - 实验报告：Word文档。在实验报告中，对你拆除了炸弹的每一道题，用文字详细描述分析求解过程。
- **班为单位集中打包发送至助教**

- **英才班：**
  - 可网上提交，比赛每个阶段的提交时间与完成的阶段数，看评分。
  - 破解，修改bomb，不用输入密码，直接都ok

五、实验报告格式

- 按照实验报告模板所要求的格式与内容提交。
- 实验后 2 周内 提交至课代表并打包给助教。
- 本次实验成绩按100分计
  - 按时上课，签到5分
  - 按时下课，不早退5分
  - 课堂表现：10分，不按操作规程、非法活动扣分。
  - 实验报告：80分。具体参见实验报告各环节的分值
- 提交CS1907_H190307099.txt （ans.txt）
- 提交实验报告 CS1907_H190307099_学霸.docx
- 提交实验报告 CS1907_H190307099_学霸.pdf

- 学生提交3个文件，课代表提交3个包