二进制炸弹实验

（苏丰，南京大学，修改自 CMU:CSAPP2 实验）

一、实验介绍

本实验要求运用课程所学程序与数据的机器表示方面知识，拆除一个二进制炸弹（“binary bombs”，下文简称为炸弹）程序中设置的多个关卡，在该过程中增强对程序与数据的机器级表示、汇编语言、调试器和逆向工程等方面知识与技能的掌握。

一个二进制炸弹是一个 Linux 可执行程序，包含了多个阶段（又称为层次、关卡）。在炸弹程序运行的每个阶段要求输入一个特定字符串，如果该输入字符串符合程序的要求，该阶段的炸弹就被拆除了，否则炸弹“爆炸——即打印输出“BOOM!!!”的提示。

每个炸弹阶段考察了程序与数据的机器级表示的不同方面，难度逐级递增：

- 阶段 0：字符串比较
- 阶段 1：浮点数表示
- 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作为阶段 7，只有在阶段 4 的拆解字符串后附加一特定字符串后，才能在实验最后进入隐藏阶段。

实验的目标是拆除尽可能多的炸弹关卡——分析获得尽可能多的正确拆解字符串。

- 实验环境：Linux i386
- 实验语言：汇编

二、实验数据

在本实验中，每位学生可从 Lab 实验网站下载包含本实验相关文件的一个 tar 文件。可在 Linux 实验环境中使用命令“**tar xvf <tar 文件名>**”将其中包含的文件提取到当前目录中。该 tar 文件中包含如下实验所需文件：

- bomb：二进制炸弹可执行程序
- bomb.c：包含 bomb 程序中 main 函数的 C 语言程序框架

运行二进制炸弹可执行程序 bomb 需要指定 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，程序根据输入字符串决定是通过相应阶段还是引爆炸弹导致该阶段任务失败。
- 也可将拆除每一炸弹阶段的字符串按行（一行一个字符串）记录在一个文本文件（必须采用 Unix/Linux 换行格式）中（即实验结果提交文件的形式），然后将该文件作为运行二进制炸弹程序时的唯一命令行参数，程序将依次检查对应每一阶段的字符串来决定炸弹拆除成败。

注意：如果拆解字符串（来自命令行输入或文本文件）不正确导致相应炸弹阶段被引爆，程序在输出炸弹爆炸的提示文字“BOOM!!!”后，将进入下一阶段的字符串检查（等待命令行输

入或读取文件下一行）而不会终止程序的运行。因此，如果暂时未能正确获得某阶段的拆解字符串，可用任意非空字符串（即不同于空格、制表、换行的一个以上字符）临时作为拆解字符串，从而在引爆相应炸弹阶段后，跳到以后阶段继续开展实验。

三、实验结果提交

提交文件名：学号.txt

提交文件格式：每个拆解字符串一行，除此之外不要包含任何其它字符

注意：

- ✓ 实验结果提交文件必须采用 Unix/Linux 的换行字符格式（不同于 Windows 上默认使用的换行字符），因此建议在实验所用的 Linux 环境中编写该文件，并在提交前对其进行检查确认（作为运行 bomb 程序时的命令行参数并检查程序输出）。
- ✓ 在最后一个字符串后也要进行换行，否则将导致最后阶段拆解不正确。

四、实验工具

为完成二进制炸弹拆除任务，可使用 objdump 工具程序反汇编可执行炸弹程序，并使用 gdb 工具程序单步跟踪每一阶段的机器指令，从中理解每一指令的行为和作用，进而推断拆除炸弹所需的目标字符串的内容组成。例如，可在每一阶段的起始指令前和引爆炸弹的函数调用指令前设置断点。

下面简要说明完成本实验所需要的一些实验工具：

GDB

为从二进制炸弹可执行程序"bomb"中找出触发炸弹爆炸的条件，可使用 GDB 程序帮助对程序的分析。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 可帮助完成以下几方面的调试工作（更详细描述可参看 GDB 文档和相关资料）：

- 装载、启动被调试的程序
- 使被调试程序在指定的调试断点处中断执行，以方便查看程序变量、寄存器、栈内容等程序运行的现场数据
- 动态改变程序的执行环境，如修改变量的值

objdump

- –t 选项：打印指定二进制程序的符号表，其中包含了程序中的函数、全局变量的名称和存储地址
- –d 选项：对二进制程序中的机器指令代码进行反汇编。通过分析汇编源代码可以发现 bomb 程序是如何运行的

strings

该命令显示二进制程序中的所有可打印字符串

五、实验步骤演示

下面以 phase0 为例演示一下基本的实验步骤（以下示例中的代码、数据、地址等可能不同于实际获得的程序，仅供参考）：

首先调用"objdump –d bomb > disassemble.txt"对二进制可执行程序 bomb 进行反汇编，

并将汇编代码输出到“disassemble.txt”文本文件中。

查看该汇编代码文件，可以在 main 函数中找到如下指令，从而得知 phase0 的处理程序包含在“main()“函数所调用的函数“phase_0()”中：

```
8048a4c: c7 04 24 01 00 00 00    movl    $0x1,(%esp)
8048a53: e8 2c fd ff ff          call    8048784 <__printf_chk@plt>
8048a58: e8 49 07 00 00          call    80491a6 <read_line>
8048a5d: 89 04 24                mov     %eax,(%esp)
8048a60: e8 a1 04 00 00          call    8048f06 <phase_0>
8048a65: e8 4a 05 00 00          call    8048fb4 <phase_defused>
8048a6a: c7 44 24 04 40 a0 04    movl    $0x804a040,0x4(%esp)
```

接下来在汇编代码文件中继续查找 phase_0 的具体定义，如下所示：

```
08048f06 <phase_0>:
8048f06: 55                      push    %ebp
8048f07: 89 e5                   mov     %esp,%ebp
8048f09: 83 ec 18                sub     $0x18,%esp
8048f0c: c7 44 24 04 fc a0 04    movl    $0x804a0fc,0x4(%esp)
8048f13: 08
8048f14: 8b 45 08                mov     0x8(%ebp),%eax
8048f17: 89 04 24                mov     %eax,(%esp)
8048f1a: e8 2c 00 00 00          call    8048f4b <strings_not_equal>
8048f1f:  85 c0                  test    %eax,%eax
8048f21: 74 05                   je      8048f28 <phase_0+0x22>
8048f23: e8 49 01 00 00          call    8049071 <explode_bomb>
8048f28: c9                      leave
8048f29: c3                      ret
……
```

从上面的汇编代码中可以看出函数 strings_not_equal 所需要的两个参数保存在(%esp)和 0x4(%esp)所指向的栈存储单元里。在前面的 main()函数中可以找到：

```
8048a58: e8 49 07 00 00          call    80491a6 <read_line>
8048a5d: 89 04 24                mov     %eax,(%esp)
```

由于%eax 里存储的是 read_line()函数返回的结果，也就是用户输入字符串的地址，所以很容易推断出和用户输入字符串相比较的字符串的存储地址为 0x804a0fc，因此可以使用 gdb 查看这个地址存储的数据内容，具体过程如下：

```
$gdb bomb
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) b main
Breakpoint 1 at 0x80489a5: file bomb.c, line 45.
(gdb) r
Starting program:./bomb

Breakpoint 1, main (argc=1, argv=0xbffff3f4) at bomb.c:45
45          if (argc == 1) {
(gdb) ni
0x080489a8  45          if (argc == 1) {
(gdb) ni
46          infile = stdin;
(gdb) ni
0x080489af  46          infile = stdin;
(gdb) ni
0x080489b4  46          infile = stdin;
(gdb) ni
67          initialize_bomb();
(gdb) ni
printf (argc=1, argv=0xbffff3f4) at /usr/include/bits/stdio2.h:105
105     return __printf_chk (__USE_FORTIFY_LEVEL - 1, __fmt, __va_arg_pack ());
(gdb) ni
0x08048a38  105     return __printf_chk (__USE_FORTIFY_LEVEL - 1, __fmt, __va_arg_pack ());
(gdb) ni
0x08048a3f  105     return __printf_chk (__USE_FORTIFY_LEVEL - 1, __fmt, __va_arg_pack ());
(gdb) ni
Welcome to my fiendish little bomb. You have 6 phases with
0x08048a44 in printf (argc=1, argv=0xbffff3f4)
    at /usr/include/bits/stdio2.h:105
105     return __printf_chk (__USE_FORTIFY_LEVEL - 1, __fmt, __va_arg_pack ());
(gdb) ni
0x08048a4c  105     return __printf_chk (__USE_FORTIFY_LEVEL - 1, __fmt, __va_arg_pack ());
(gdb) ni
0x08048a53  105     return __printf_chk (__USE_FORTIFY_LEVEL - 1, __fmt, __va_arg_pack ());
(gdb) ni
which to blow yourself up. Have a nice day!
main (argc=1, argv=0xbffff3f4) at bomb.c:73
73          input = read_line();                /* Get input                    */
(gdb) ni
74          phase_0(input);                     /* Run the phase               */
 (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
(gdb)

其中从 0x804a0fc 地址开始到“0x00”字节结束的字节序列就是应输入的拆解字符串的 ASCII 码，根据小端存储规则，可以查 ASCII 表得到该字符串为”I am just a renegade hockey mom.“，从而完成了该阶段拆解字符串的求解。