计算机系统实验

第一次实验

CONTENTS

目录

本学期实验总体安排

- 实验总分：20分，总共8学时

实验项目	内容	提交	分数	课时
Lab1: Bomblab	通过逆向工程和调试技术，挑战多阶段炸弹破解任务，涵盖字符串处理、数组操作、条件判断、字符串拆解、比较及数学运算	代码及实验报告	10	4
Lab2: perflab	通过循环展开和缓存优化等技术，优化图像旋转与掩膜平滑操作	代码及实验报告	10	4

- **课程主页及指导书**（校内网）：https://comp3052-2.p.cs-lab.top

实验环境介绍

- **实验环境：Linux 64-bit，C/汇编语言**
  - 可以选择使用远程实验平台，也可以利用自己搭建的 **Linux** 环境完成实验。
- **远程实验平台（SSH协议登录，详见：远程实验平台使用指南）**
  - **IP**地址：10.249.12.98，端口号：6666
  - 命令：ssh 你的学号@10.249.12.98 –p 6666
  - 用户名/初始密码：**你的学号**
- **上传实验文件工具：VScode、 Mobaxterm、SCP命令**

安全警告

违规会掉装备！被抓包的同学会将被遣返你自己的本地环境

- **禁用VSCode AI外挂：** ❌ 请卸载 VSCode 在 remote 远端（`10.249.12.98`）的代码自动生成插件，包含但不限于 MarsCode、tabnine、Copilot 等。这类工具会大量占用远程实验平台的有限资源，极易造成服务器过载，导致其他同学无法正常开展实验。
- **平台用途限定：** 本实验平台仅服务于《计算机系统》、《操作系统》、《计算机设计与实践》等课程实验，严禁用于任何其他用途。

Lab1 Bomblab实验概述

实验目的

- 通过对一个二进制可执行程序（称为“二进制炸弹”）的理解和逆向工程，加深对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。

实验内容

- 二进制炸弹“binary bombs”可执行程序包含了多个阶段（或关卡），在每个阶段程序要求你输入一个特定字符串，如果输入满足程序代码所定义的要求，该阶段的炸弹就被拆除了，否则程序输出“炸弹爆炸BOOM!!!”的提示并转到下一阶段再次等待对应的输入 —— 实验的目标是**设法得出解除尽可能多阶段的字符串**。

- 为拆除二进制炸弹，需要**通过反汇编和理解可执行炸弹文件程序**或**使用gdb调试器**跟踪每一阶段的机器代码，从中理解关键机器指令的行为和作用，进而设法推断拆除炸弹所需的目标字符串。

Lab1 Bomblab实验包获取

- **实验文件**下载地址（校内网）： https://file.cs-lab.top/s/KdFWRNRjUaGaQ0v
- **学生实验数据包：**<**学号**>.tar （**每位同学不一样**）
- **解压命令：** tar xf <**学号**>.tar
- 数据包中主要包含下列文件：
  - bomblab：可执行程序，也就是我们需要破解的**炸弹** 💣
  - bomb.c：bomb主程序，帮助拆弹者了解代码框架，没有细节。

Lab1 Bomblab实验bomb文件

二进制炸弹目标程序

- 包含了7个阶段以及1个隐藏阶段，分别集中考察对以下二进制程序表示各方面的理解和掌握：
  - **阶段0**：字符串比较
  - **阶段1**：数值比较
  - **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（**选做**）

Lab1 Bomblab实验bomb运行

二进制炸弹可执行程序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**

方法1：不带参数运行 $./bomb

- 根据提示，逐阶段手工输入拆弹字符串

$ ./bomb

Welcome to my fiendish little bomb. You have 7 phases with
which to blow yourself up. Have a nice day!
（等待输入阶段0的拆解字符串）

方法2：带参数运行 $./bomb 学号.txt（详见"学号.txt"的格式要求）

- 程序会检查每一阶段的拆弹密码字符串。

$ ./bomb 学号.txt

Welcome to my fiendish little bomb. You have 7 phases with
which to blow yourself up. Have a nice day!
Well done! You seem to have warmed up!
……（其余阶段的验证输出）

Lab1 Bomblab实验工具（1）

- **objdump**：反汇编二进制炸弹程序，获得其中汇编指令供分析
  - objdump -d bomb　输出bomb程序的反汇编结果
  - objdump -d bomb > bomb.s　获得bomb程序的反汇编结果并保存于文本文件bomb.s中供分析
  - objdump -t bomb　打印bomb程序的符号表，其中包含bomb中所有函数、全局变量的名称和存储地址
- **strings**: 显示二进制程序中的所有可打印字符串
  - $ strings bomb

Lab1 Bomblab实验工具（2）

- **gdb**：强大的交互式程序调试工具（详见：GDB调试指南），可帮助从二进制可执行bomb程序中分析、找出触发bomb爆炸的条件。
  - gdb bomb　　启动GDB，加载bomb程序
  - (gdb) b phase_1　　在phase_1函数入口设断点
  - (gdb) r 运行　c 继续
  - (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) x/[数量][格式][单位] <地址>　查看内存地址数据
  - (gdb) quit 退出GDB

格式符	说明	示例
x	十六进制	x/xw 0x402400
d	十进制	x/dw 0x402400
s	字符串	x/s 0x402400
i	汇编指令	x/5i 0x401100

单位符	说明	示例
b	字节（1字节）	x/8xb $rsp
h	半字（2字节）	x/4xh 0x402400
w	字（4字节）	x/3wd $rbp-0x8
g	巨字（8字节）	x/2gx 0x6032d0

实验任务——以“阶段0 字符串比较”为例

- **阶段说明：该阶段要求输入与程序中内置的某一特定字符串相匹配的字节序列**
- **实验步骤：**
  - **Step1**：对二进制炸弹程序进行反汇编

    $ objdump -d bomb > bomb.s

```
00000000004014ec <phase_0>:
  4014ec:       f3 0f 1e fa             endbr64
  4014f0:       55                      push   %rbp
  4014f1:       48 89 e5                mov    %rsp,%rbp
  4014f4:       be 80 21 40 00          mov    $0x402180,%esi
  4014f9:       e8 49 05 00 00          call   401a47 <strings_not_equal>
  4014fe:       85 c0                   test   %eax,%eax
  401500:       75 07                   jne    401509 <phase_0+0x1d>
  401502:       b8 01 00 00 00          mov    $0x1,%eax
  401507:       5d                      pop    %rbp
  401508:       c3                      ret
  401509:       e8 4d 06 00 00          call   401b5b <explode_bomb>
  40150e:       b8 00 00 00 00          mov    $0x0,%eax
  401513:       eb f2                   jmp    401507 <phase_0+0x1b>
```

实验任务——以“阶段0 字符串比较”为例

- **Step2**：本阶段函数汇编指令代码的分析
  - 定位与函数功能相对应的各控制逻辑
  - 获得主要变量的地址

内置字符串基址

输入字符串基址：

%rdi（第1个参数）

字符串匹配比较

```
00000000004014ec <phase_0>:
  4014ec:       f3 0f 1e fa             endbr64
  4014f0:       55                      push   %rbp
  4014f1:       48 89 e5                mov    %rsp,%rbp
  4014f4:       be 80 21 40 00          mov    $0x402180,%esi
  4014f9:       e8 49 05 00 00          call   401a47 <strings_not_equal>
  4014fe:       85 c0                   test   %eax,%eax
  401500:       75 07                   jne    401509 <phase_0+0x1d>
  401502:       b8 01 00 00 00          mov    $0x1,%eax
  401507:       5d                      pop    %rbp
  401508:       c3                      ret
  401509:       e8 4d 06 00 00          call   401b5b <explode_bomb>
  40150e:       b8 00 00 00 00          mov    $0x0,%eax
  401513:       eb f2                   jmp    401507 <phase_0+0x1b>
```

分析红色框中汇编代码可知：
为避免执行0x401509处对引爆炸弹函数的调用指令，jne指令的测试条件应被满足，即0x4014fe处test指令执行之前，寄存器EAX的值应为0。

实验任务——以“阶段0 字符串比较”为例

- **Step2**：本阶段函数汇编指令代码的分析
  - 确定strings_not_equal函数的控制逻辑

(1) 在程序的反汇编结果中搜索、找到strings_not_equal函数的汇编代码

(2) 分析汇编代码可知：该函数在输入的两个字符串参数的长度和内容均相同时将返回0，否则返回1，并且返回值保存于EAX寄存器中

```
> 0x401a47 <strings_not_equal>     endbr64
  0x401a4b <strings_not_equal+4>   push   %rbp
  0x401a4c <strings_not_equal+5>   mov    %rsp,%rbp
  0x401a4f <strings_not_equal+8>   push   %r13
  0x401a51 <strings_not_equal+10>  push   %r12
  0x401a53 <strings_not_equal+12>  push   %rbx
  0x401a54 <strings_not_equal+13>  sub    $0x8,%rsp
  0x401a58 <strings_not_equal+17>  mov    %rdi,%rbx
  0x401a5b <strings_not_equal+20>  mov    %rsi,%r12
  0x401a5e <strings_not_equal+23>  call   0x401a2f <string_length>
  0x401a63 <strings_not_equal+28>  mov    %eax,%r13d
  0x401a66 <strings_not_equal+31>  mov    %r12,%rdi
  0x401a69 <strings_not_equal+34>  call   0x401a2f <string_length>
  0x401a6e <strings_not_equal+39>  cmp    %eax,%r13d
  0x401a71 <strings_not_equal+42>  je     0x401a8b <strings_not_equal+68>
  0x401a73 <strings_not_equal+44>  mov    $0x1,%eax
  0x401a78 <strings_not_equal+49>  add    $0x8,%rsp
  0x401a7c <strings_not_equal+53>  pop    %rbx
  0x401a7d <strings_not_equal+54>  pop    %r12
  0x401a7f <strings_not_equal+56>  pop    %r13
  0x401a81 <strings_not_equal+58>  pop    %rbp
```

实验任务——以“阶段0 字符串比较”为例

- **Step3**：定位并获得内置字符串的值，并相应构造输入字符串
  - **前面已推断出：**和用户输入字符串相比较的程序内置字符串的存储地址为0x402180，并且两个字符串的内容应相同，因此可使用gdb查看地址0x402180中存储的内置字符串的内容：

```
$ gdb bomb
……
(gdb) break phase_0
(gdb) r
Starting program: /home/ics/Course/bomblab/bomb
Welcome to my fiendish little bomb. You have 7 phases with
which to blow yourself up. Have a nice day!
sgjsogjsoigjosjs（此处暂时随意输入一些字符）

(gdb) x/s 0x402180
0x402180:      “Multiple strong symbols are not allowed."
```

  - 从中可看出，内置字符串（到标志其结束的0x00字节为止）为“Multiple strong symbols are not allowed.“ —— 此即本阶段期望的输入拆解字节串

实验结果的提交 （详见：实验结果文件提交说明）

- **将对应二进制炸弹每一阶段的拆解字符串写入一纯文本文件，命名为“学号.txt”，其中每个拆解字符串独立一行，例如：**

```
# 文件命名：学号.txt
# 格式样例：
Phase0_Answer
I_Love_CSAPP
3.1415926
...
(最后一行必须空行！)
```

提交须知

- 所有提交将重新跑分验证，为确保得分有效，请务必：
  - **1.确认实验包学号匹配**
  - **2.严格遵循文件格式要求**

未达标者将直接判定为无效提交

- **该文本文件必须采用Unix格式（换行字符不同于Windows格式）。另外，注意最后一个字符串后也要进行换行（即所有字符串必须以换行结尾）。**
- **如果未完成其中某阶段，应用任一非空白字符串（即不全是空格、制表、换行字符）置于对应该阶段的文本文件相应行中（不能放置一空行或直接省略该行）。**

附加题（实验课程总分不超过20分）

挑战 1：隐藏阶段 （+0.5分）

- 解锁条件：在阶段 4 正确输入拆解字符串后，追加特定隐藏字符串即可解锁终极挑战
- 考察要点：综合运用动态调试、反汇编分析及字符串处理技巧
- 提交要求：
  - 实验报告中需包含完整的逆向分析过程（含关键汇编代码解析）
  - 提供触发隐藏阶段的完整输入序列（含阶段**0~6**以及隐藏阶段），以“学号**.txt**”格式提交，具体要求见实验结果文件提交说明

挑战 2：二进制免验证破解 （最高+2分，部分完成酌情加分）

- 破解目标：通过逆向工程修改 bomb 可执行文件，实现无需输入密码直接通关
- 验收标准：
  - 提交修改后的 **bomb** 可执行文件（需通过所有测试用例）
  - 实验报告包含完整的逆向分析流程图及关键代码修改说明

作业提交

提交内容：实验报告（有模板）+ <学号>.tar

截止时间：

实验课后两周内提交至HITsz Grader 作业提交平台，具体截止日期参考平台发布。

- 登录网址：： http://grader.tery.top:8000/#/login
- 推荐浏览器： Chrome
- 初始用户名、密码均为学号，登录后请修改

注意

上传后可自行下载以确认是否正确提交

同学们
请开始实验吧！