哈爾濱工業大學

实验报告

实 验（三）

题　　目 Binary Bomb

二进制炸弹

专　　业 计算机类

学　　号 1190200523

班　　级 1903002

学　　生 石翔宇

指 导 教 师 郑贵滨

实 验 地 点 G709

实 验 日 期 2021.4.23

计算机科学与技术学院

目 录

第 1 章 实验基本信息

1.1 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

1.2 实验环境与工具

1.2.1 硬件环境

- Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
- 16GB RAM
- 1TB HDD + 512G SSD

1.2.2 软件环境

- Windows 10 21H1
- Ubuntu 20.04 LTS

1.2.3 开发工具

- VSCode，CodeBlocks，gcc+gdb，edb

1.3 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？

第 2 章 实验环境建立

2.1 Ubuntu 下 CodeBlocks 反汇编（10 分）

CodeBlocks 运行 hellolinux.c。反汇编查看 printf 函数的实现。

要求：C、ASM、内存(显示 hello 等内容)、堆栈（call printf 前）、寄存器同时在一个窗口。

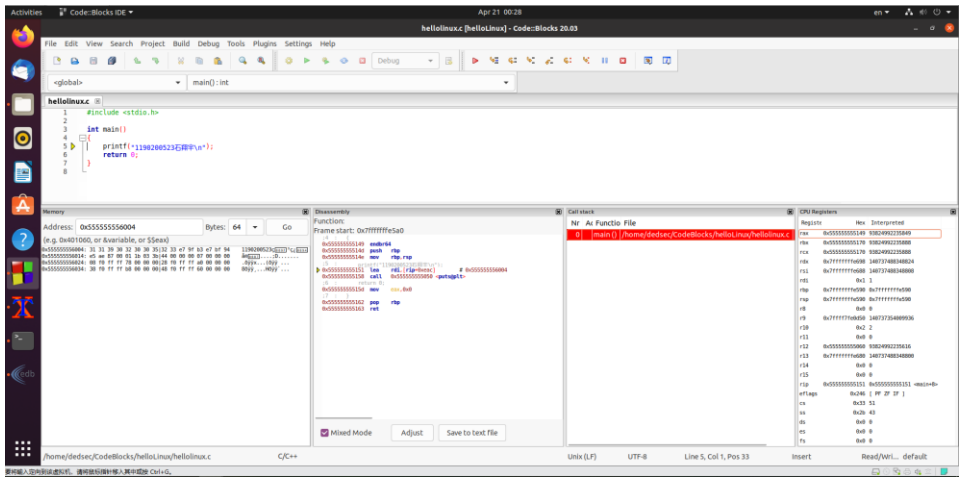


图 2-1 Ubuntu 下 CodeBlocks 反汇编截图

2.2 Ubuntu 下 EDB 运行环境建立（10 分）

用 EDB 调试 hellolinux.c 的执行文件，截图，要求同 2.1

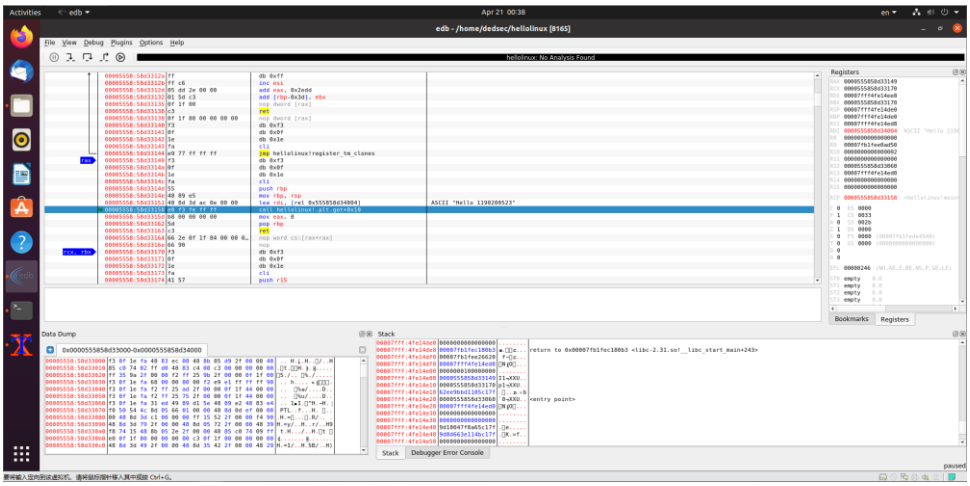


图 2-2 Ubuntu 下 EDB 截图

第 3 章 各阶段炸弹破解与分析

每阶段 15 分（密码 10 分，分析 5 分），总分不超过 80 分

3.1 阶段 1 的破解与分析

密码如下：You can Russia from land here in Alaska.

破解过程：

1. 查看反汇编代码得知此阶段比较字符串是否相等。
2. 由反汇编代码 4013fd: be 50 31 40 00 mov $0x403150,%esi 得知答案字符串存在 0x403150 处
3. 提取得字符串 16 进制表示 59 6f 75 20 63 61 6e 20 52 75 73 73 69 61 20 66 72 6f 6d 20 6c 61 6e 64 20 68 65 72 65 20 69 6e 20 41 6c 61 73 6b 61 2e
4. 由程序：

```
int main()
{
    freopen("phase1.txt", "r", stdin);
    char a[2];
    while(scanf("%s", a) != EOF)
    {
        int x = 0;
        if(a[0] >= '0' && a[0] <= '9')
            x = a[0] - '0';
        else if(a[0] >= 'a' && a[0] <= 'f')
            x = a[0] - 'a' + 10;
        x = x * 16;
        if(a[1] >= '0' && a[1] <= '9')
            x += a[1] - '0';
        else if(a[1] >= 'a' && a[1] <= 'f')
            x += a[1] - 'a' + 10;
```

```
            printf("%c", x);
        }
        return 0;
    }
```

转换成字符串“You can Russia from land here in Alaska.”，得到答案

3.2 阶段 2 的破解与分析

密码如下：1 2 4 7 11 16（满足第一个数为非负整数，其余依次递增 1,2,3,4,5 的六个整数即可）

破解过程：

1. 查看反汇编代码

   401421: e8 ef 04 00 00 callq 401915 <read_six_numbers>

   及 read_six_numbers 的有关内容得知此阶段首先输入 6 个整数，存储于%rbp-0x30 处。

2. 由反汇编代码

   401426: 83 7d d0 00 cmpl $0x0,-0x30(%rbp)

   40142a:78 07 js 401433 <phase_2+0x1f>

   可知第一个数字不能为负数

3. 40142c:bb 01 00 00 00 mov $0x1,%ebx

   首先设置%ebx 为 1，%ebx 代表当前进行到第几个数字（下标从 0 开始），

   401442: 83 fb 05 cmp $0x5,%ebx

   401445: 7f 17 jg 40145e <phase_2+0x4a>

   当%ebx 达到 6 时跳出，

   401447: 48 63 c3 movslq %ebx,%rax

   令%rax = %ebx，%rax 表示当前第几个数字，

   40144a: 8d 53 ff lea -0x1(%rbx),%edx

令%edx = %rbx(%ebx) – 1，%edx 表示当前数字的前一个，

40144d: 48 63 d2　　movslq %edx,%rdx

401450: 89 d9　　mov　　%ebx,%ecx

令%ecx = %ebx，表示当前第几个数，也即当前数字需要比前一个数字大多少，

401452: 03 4c 95 d0　　add　　-0x30(%rbp,%rdx,4),%ecx

-0x30(%rbp,%rdx,4)表示前一个数字的值，加到%ecx 上，现在的%ecx 就表示当前数字期望的值，

401456:　　39 4c 85 d0　　cmp　　%ecx,-0x30(%rbp,%rax,4)

-0x30(%rbp,%rax,4)表示输入的当前数字的值，%ecx 表示当前数字期望的值，将两个数字进行比较，

40145a: 74 e3　　je　　40143f <phase_2+0x2b>

若两数字相等则回到 0x40143f，继续循环，

40145c: eb dc　　jmp　　40143a <phase_2+0x26>

否则炸弹爆炸。

4. 由此可知，满足第一个数为非负整数，其余依次递增 1,2,3,4,5 的六个整数即为答案。

3.3 阶段 3 的破解与分析

密码如下：0 168 或 1 967 或 2 408 或 3 811 或 4 82 或 5 323 或 6 90 或 7 319

破解过程：

1. 查看反汇编代码

401475:　　be 4f 33 40 00　　mov　　$0x40334f,%esi

及 0x40334f 处的有关内容得知此阶段首先输入 2 个整数，存储于%rbp-0x4 和%rbp-0x8 处。

2. 由反汇编代码

401489: 8b 45 fc mov -0x4(%rbp),%eax

40148c:83 f8 07 cmp $0x7,%eax

40148f: 77 46 ja 4014d7 <phase_3+0x72>

可知第一个输入应小于或者等于 7

3. 由反汇编代码

401493: ff 24 c5 c0 31 40 00 jmpq *0x4031c0(,%rax,8)

由对应位置的内容可知，第一个数分别等于 0，1，2，3，4，5，6，7 时分别跳到 00 00 00 00 00 40 14 a1，00 00 00 00 00 40 14 e3，00 00 00 00 00 40 14 ad，00 00 00 00 00 40 14 b4，00 00 00 00 00 40 14 bb，00 00 00 00 00 40 14 c2，00 00 00 00 00 40 14 c9，00 00 00 00 00 40 14 d0 处

4. 由各个对应位置内容可知，第一个数分别等于 0，1，2，3，4，5，6，7 时，且第二个数分别等于 0xa8，0x3c7，0x198，0x32b，0x52，0x143，0x5a，0x13f，即为答案。

5. 将对应的十六进制转为十进制得到答案：0 168 或 1 967 或 2 408 或 3 811 或 4 82 或 5 323 或 6 90 或 7 319

3.4 阶段 4 的破解与分析

密码如下：176 2 或 264 3 或 352 4

破解过程：

1. 查看反汇编代码

40154c:be 4f 33 40 00 mov $0x40334f,%esi

及 0x40334f 处的有关内容得知此阶段首先输入 2 个整数，存储于%rbp-0x8 和%rbp-0x4 处。

2. 查看反汇编代码

401563: 83 f8 01 cmp $0x1,%eax

401566: 7e 05 jle 40156d <phase_4+0x31>

401568: 83 f8 04 cmp $0x4,%eax

40156b: 7e 05 jle 401572 <phase_4+0x36>

可知第二个整数必须大于 1，小于等于 4，即只能为 2，3，4

3. 查看反汇编代码

401572: 8b 75 fc mov -0x4(%rbp),%esi

401575: bf 09 00 00 00 mov $0x9,%edi

40157a:e8 72 ff ff ff callq 4014f1 <func4>

向 func4 中传入两个参数，一个是第二个整数，一个是 0x9

4. 查看 func4 的反汇编代码易写出其 C 代码

```
int func4(int x, int y)
{
    if(x == 1)
        return y;
    else if(x == 0)
        return 0;
    return func4(x - 1, y) + y + func4(x - 2, y);
}
```

5. 可根据其 c 代码得出要求的第一个参数，当第二个参数分别为 2，3，4 时，第一个参数分别为 176，264，352

3.5 阶段 5 的破解与分析

密码如下：012345(123450，234501 等，包含 0~5 这 6 个数字的任意排列组合成的字符串均可)

破解过程：

1. 查看反汇编代码

401599: e8 45 02 00 00 callq 4017e3 <string_length>

40159e:83 f8 06 cmp $0x6,%eax

4015a1:75 25 jne 4015c8 <phase_5+0x3b>

可知此阶段首先输入一个长度为 6 的字符串

2. 由 0x4015a3 到 0x4015c6 的反汇编代码可知，程序中有一个累加器 x（%eax），一个总和 y（%ecx）。按照输入的字符串，依次计算地址 0x403200(,%rdx,4)，到相应的地址取数字。累加器 x 记录字符串的位置，总和 y 记录当前的和。
3. 其中各地址储存的数字分别为：

   403200：0x02

   403204：0x0a

   403208：0x06

   40320c：0x01

   403210：0x0c

   403214：0x10

   403218：0x09

   40321c：0x03

   403220：0x04

   403224：0x07

   403228：0x0e

   40322c：0x05

   403230：0x0b

   403234：0x08

   403238：0x0f

   40323c：0x0d
4. 由反汇编代码

   4015cf: 83 f9 2f　　cmp　　$0x2f,%ecx

   4015d2:　75 07　　jne　　4015db <phase_5+0x4e>

   可知程序要求总和 y 在累加完 6 个数字后必须等于 0x2f

5. 由此我们构造一个长度为 6 的字符串：由数字组成，各个数字对应的地址储存的数字总和为 0x2f。

6. 我们得到答案 012345。

3.6 阶段 6 的破解与分析

密码如下：2 3 5 1 6 4

破解过程：

1. 查看反汇编代码

4015ef: 48 8d 75 c0 lea -0x40(%rbp),%rsi

4015f3: e8 1d 03 00 00 callq 401915 <read_six_numbers>

易知此阶段首先读入 6 个整数，存储于-0x40(%rbp)处

2. 阅读 0x4015f8 到 0x401646 的代码可大概写出原始的 C 代码：

```
while (r12d <= 5)
{
    rax = *(a_40 + r12d);
    rax--;
    if(rax > 5)
        explode();
    ebx = r12d + 1;
    while(ebx <= 5)
    {
        rax = r12d;
        if(*(a_40 + ebx) == *(a_40 + rax))
            explode();
        ebx++;
    }
}
```

此段代码的功能主要为判断输入 6 个数字两两互不相等。

3. 阅读 0x401648 到 0x401676 的代码可大概写出原始的 C 代码：

```
for (int esi = 0; esi <= 5; esi++)
{
    rax = 1;
    rdx = 0x4052d0;
    while(a_40[esi] > rax)
    {
        rdx = *(rdx + 1);
        rax++;
    }
    a_70[esi] = rdx;
}
```

此段代码功能主要为按照输入的数字，找到以 0x4052d0 开始的链表相应位置的数字的地址（例如输入为 3，则找到以 0x4052d0 开始的链表的第 3 个数字的地址），存储在-0x70(%rbp,%rcx,8)处（代码中为 a_70）

也就是将原始的链表重新排序组合，存储在-0x70(%rbp,%rcx,8)处

4. 阅读 0x 401678 到 0x 4016ab 的代码可大概写出原始的 C 代码：

```
int *rcx = a_70[0];
rax = 1;
while(rax <= 5)
{
    rdx = a_70[rax];
    *(rcx + 1) = rdx;
    rax++;
    rcx = rdx;
}
```

此代码的主要功能是为新的链表重新建立从前到后的链接

5. 阅读 0x4016ad 到 0x4016d6 的代码可大概写出原始的 C 代码：

```
int *rbx = a_70[0];
r12d = 0;
while(r12d <= 4)
```

```
{
    if(*rbx > *(*(rbx + 1)))
        explode();
    else
    {
        rbx = *(rbx + 1);
        r12d++;
    }
}
```

此代码的主要功能是在新的链表上从前到后比较相邻的两个数，需满足前面的数不大于后面的数，即新生成的数字序列是单调不降的。

6. 查看原始链表中的数字依次为：

   1：00 00 01 f8

   2：00 00 00 92

   3：00 00 01 00

   4：00 00 02 82

   5：00 00 01 0e

   6：00 00 02 7f

7. 欲将其从小到大排列，则对应下标为 2 3 5 1 6 4，得到答案。

3.7 阶段 7 的破解与分析(隐藏阶段)

密码如下：进入隐藏阶段的字符串：DrEvil，解开需要的密码：47

破解过程：

1. 观察到反汇编代码段 phase_defused 中 0x401aea 出现了对于 secret_phase 的引用，向上查找到

   401abf: be a2 33 40 00　　mov　　$0x4033a2,%esi

   401ac4:48 8d 7d b0　　lea　　-0x50(%rbp),%rdi

   401ac8:　　e8 2a fd ff ff　　callq　　4017f7 <strings_not_equal>

401acd: 85 c0 test %eax,%eax

401acf: 75 e2 jne 401ab3 <phase_defused+0x37>

表明地址 0x4033a2 存储着进入隐藏阶段需要输入的字符串，查找得 44 72 45 76 69 6c，变成字符串的形式为 DrEvil

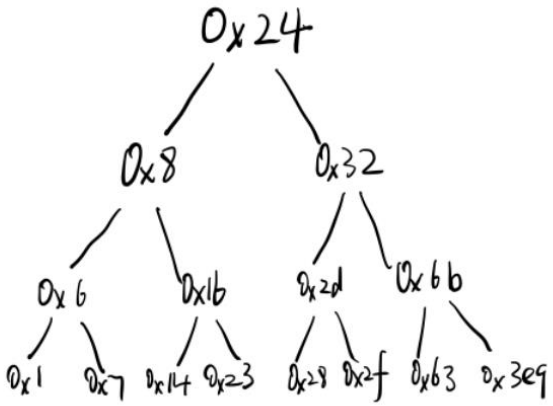
2. 观察 secret_phase 的反汇编代码，由 0x401716 到 0x401731 可知，本阶段首先输入一个整数，要求这个整数小于等于 0x3e9

3. 0x401733 和 0x401735 的代码负责传递 fun7 的参数，%edi 被赋为地址 0x4050f0，%esi 是输入的整数。

4. 阅读 fun7 的代码可大概写出原始的 C 代码：

```
int fun7(int *edi)
{
    if(*edi > esi)
    {
        edi = *(edi + 1);
        return fun7(edi) * 2;
    }
    else if(*edi != esi)
    {
        edi = *(edi + 2);
        return fun7(edi) * 2 + 1;
    }
    return 0;
}
```

由代码可知，数据在以地址 0x4050f0 开头的结构中是以链表的形式存在。Fun7 中，*edi 以位于 0x4050f0 的数据为开始，再以树形结构依次跳转到相应位置的数据，最后返回值。由相应的数据得出的树形结构如下：



5. 由反汇编代码

40173f: 83 f8 05　　cmp　　$0x5,%eax

401742: 75 1d　　jne　　401761 <secret_phase+0x50>

可知 fun7 的返回值需要为 5，构造答案为 0x2f，即 47

第 4 章 总结

4.1 请总结本次实验的收获

1. 学会了 gdb、edb、objdump 等工具的使用，更加熟练地利用这些工具来调试程序
2. 对汇编语言更加熟悉，对于各种汇编指令有了更深的理解

4.2 请给出对本次实验内容的建议

1. 使用 CMU 原版实验收获很大，建议以后章节也能继续使用原版实验

注：本章为酌情加分项。

参考文献

[1] 大卫 R.奥哈拉伦，兰德尔 E。布莱恩特. 深入理解计算机系统[M]. 机械工业出版社.2017.7