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 120L022115

班　　级 2003007

学　　生 王炳轩

指 导 教 师 吴锐

实 验 地 点 线上

实 验 日 期 2022-04-01

计算学部

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

Surface Go 3：x64、Pentium G6500Y @ 1.1GHz、16GB RAM、128GB SSD。

1.2.2 软件环境

Windows 11、Windows Subsystem for Linux、Ubuntu 20.04

1.2.3 开发工具

Code::Blocks、gcc、vim、edb、gdb

1.3 实验预习

上实验课前，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。

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。

第 2 章 实验环境建立

2.1 Ubuntu 下 CodeBlocks 反汇编（10 分）

CodeBlocks 运行 hello.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。

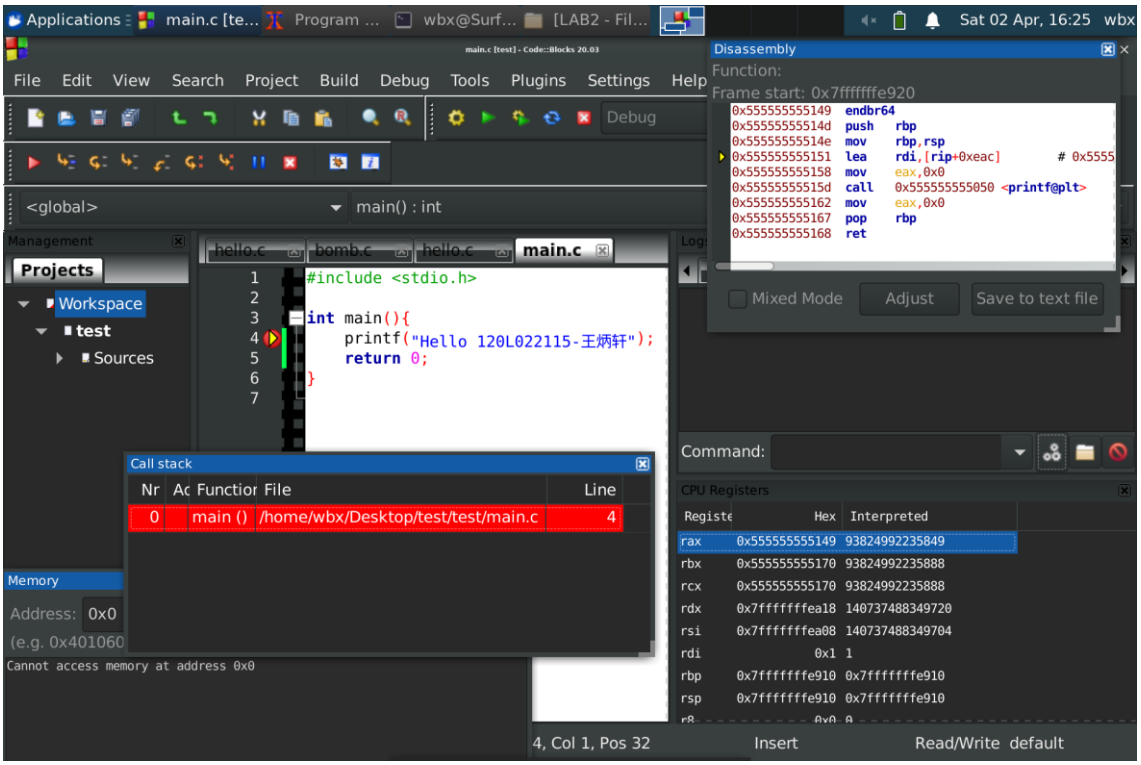


图 2-1 Ubuntu 下 CodeBlocks 反汇编截图

2.2 Ubuntu 下 EDB 运行环境建立（10 分）

用 EDB 调试 hello.c 的执行文件，截图，要求同 2.1

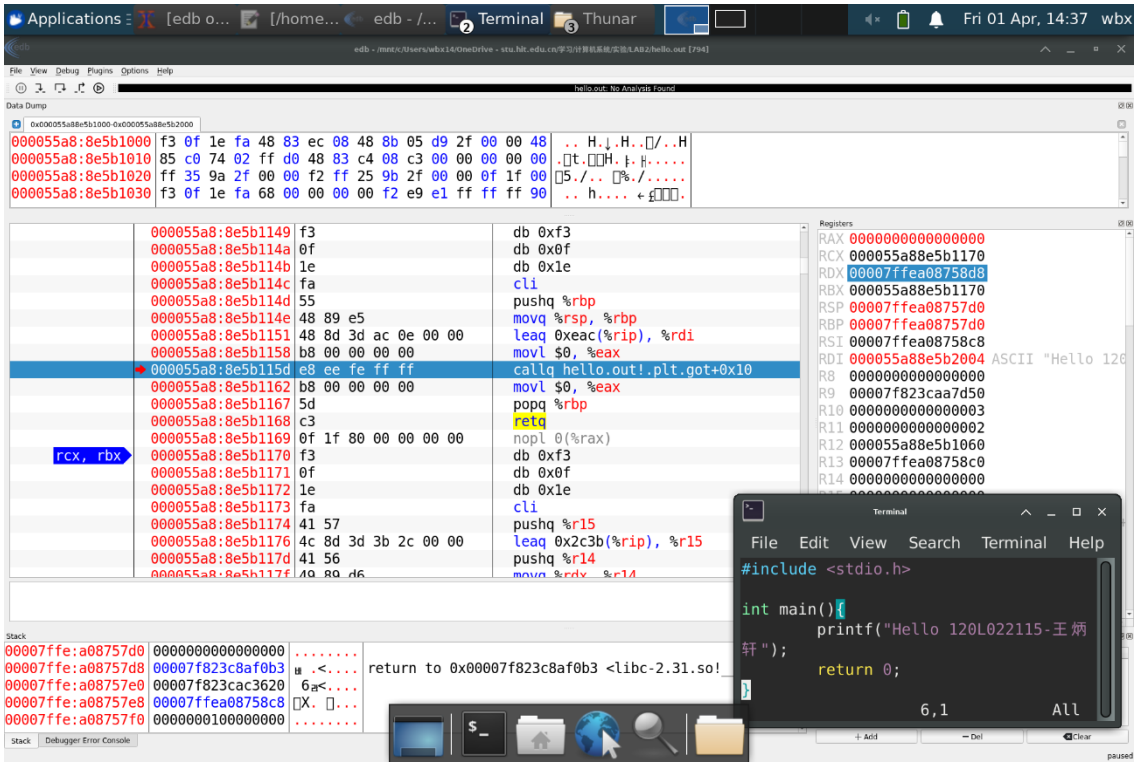


图 2-2　Ubuntu 下 EDB 截图

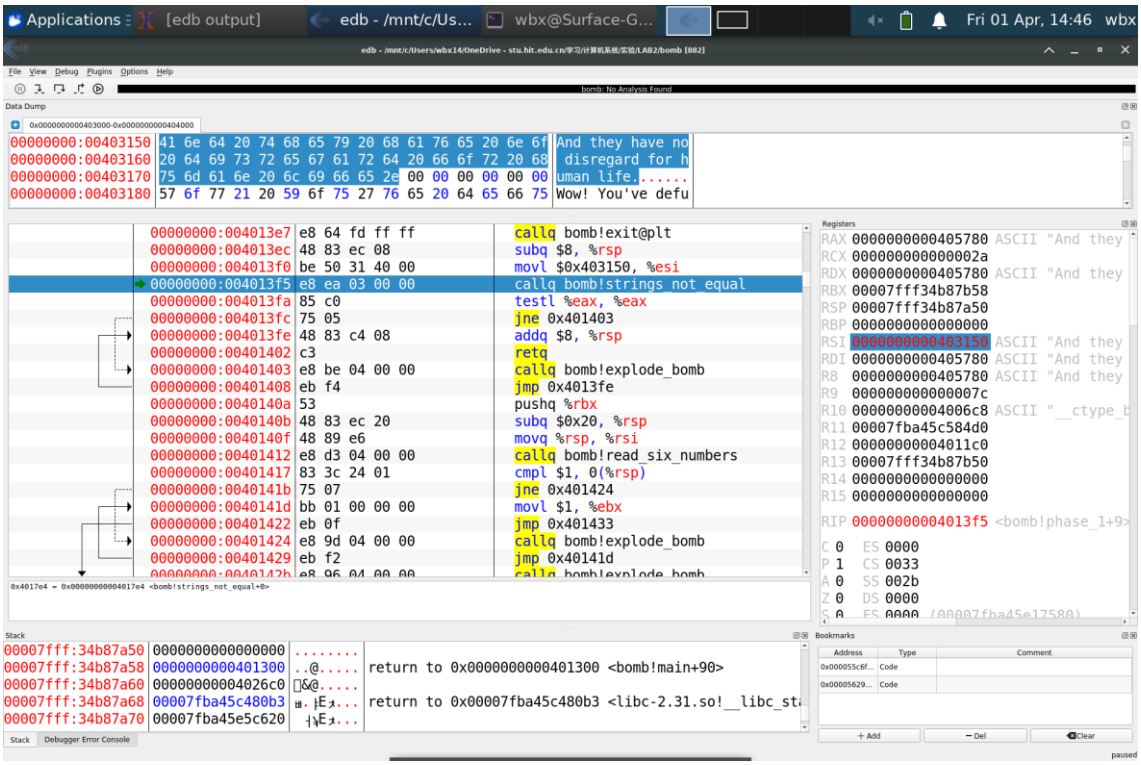
第 3 章 各阶段炸弹破解与分析

每阶段 30 分，密码 10 分，分析 20 分，总分不超过 80 分

3.1 阶段 1 的破解与分析

密码如下：And they have no disregard for human life.

破解过程：进入 Phase_1 函数，发现为直接与 0x403150 地址的字符串进行比较，如果比较正确即破解，因此第一关的密码就是位于该地址的字符串，通过 DataDump 查看，得到密码。



3.2 阶段 2 的破解与分析

密码如下：1 2 4 8 16 32

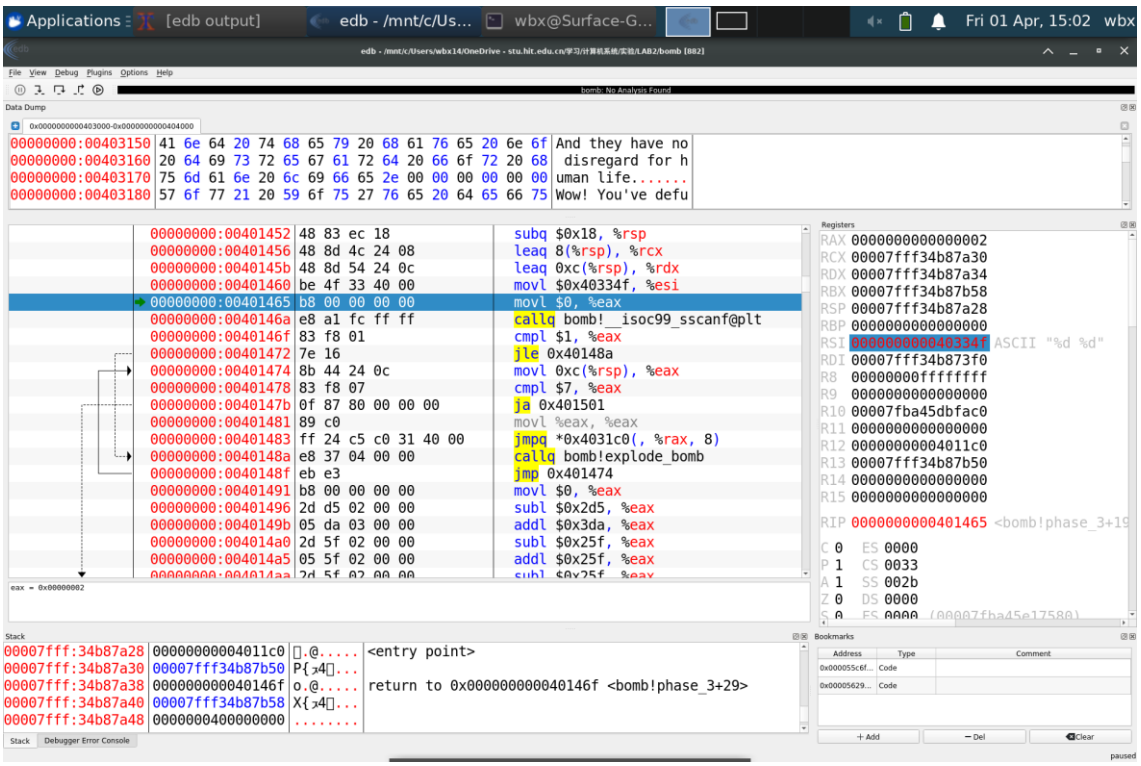
破解过程：进入 Phase_2 函数，发现了一个读入 6 个数的函数，那么密码就一定是 6 个数。jne 给出了第一个数为 1，因为如果不等就会爆炸。之后发现有一个 6 次循环比较（ebx 是循环次数，当为 5 的时候跳转到 ret），6 个数分别与 eax 比较，而 eax 从 1 开始、每次比较都翻倍，所以这 6 个数为 1、2、4、

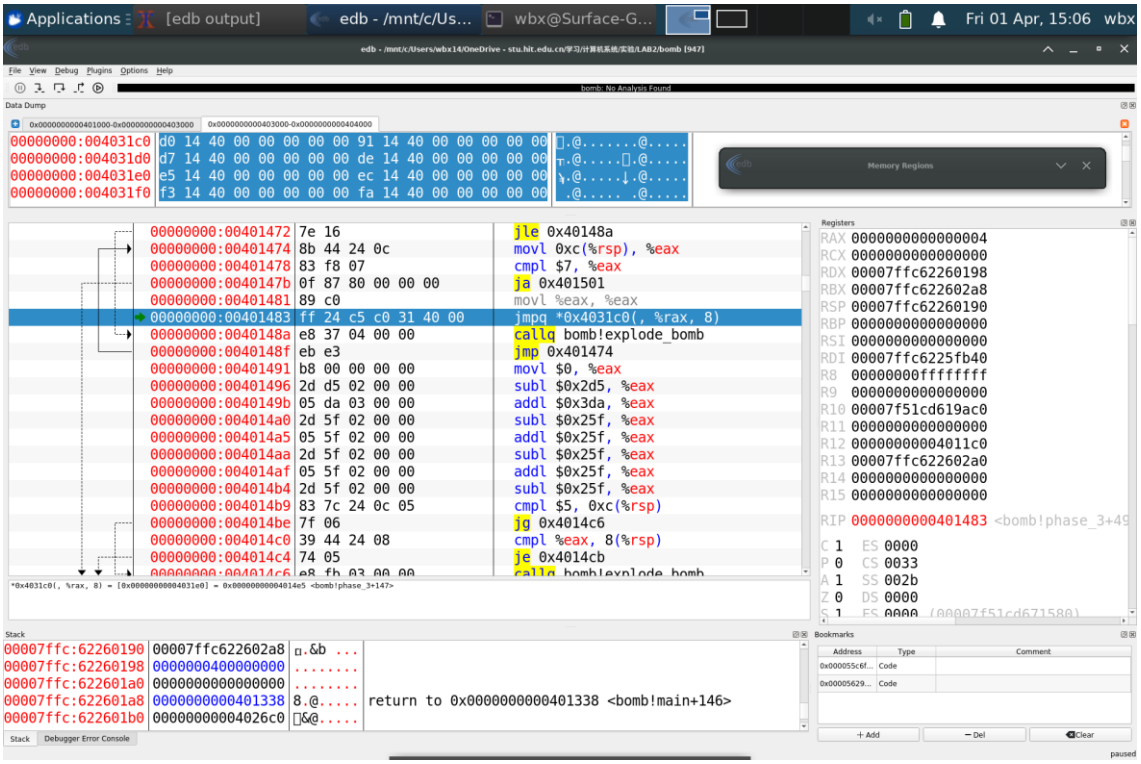
8、16、32.

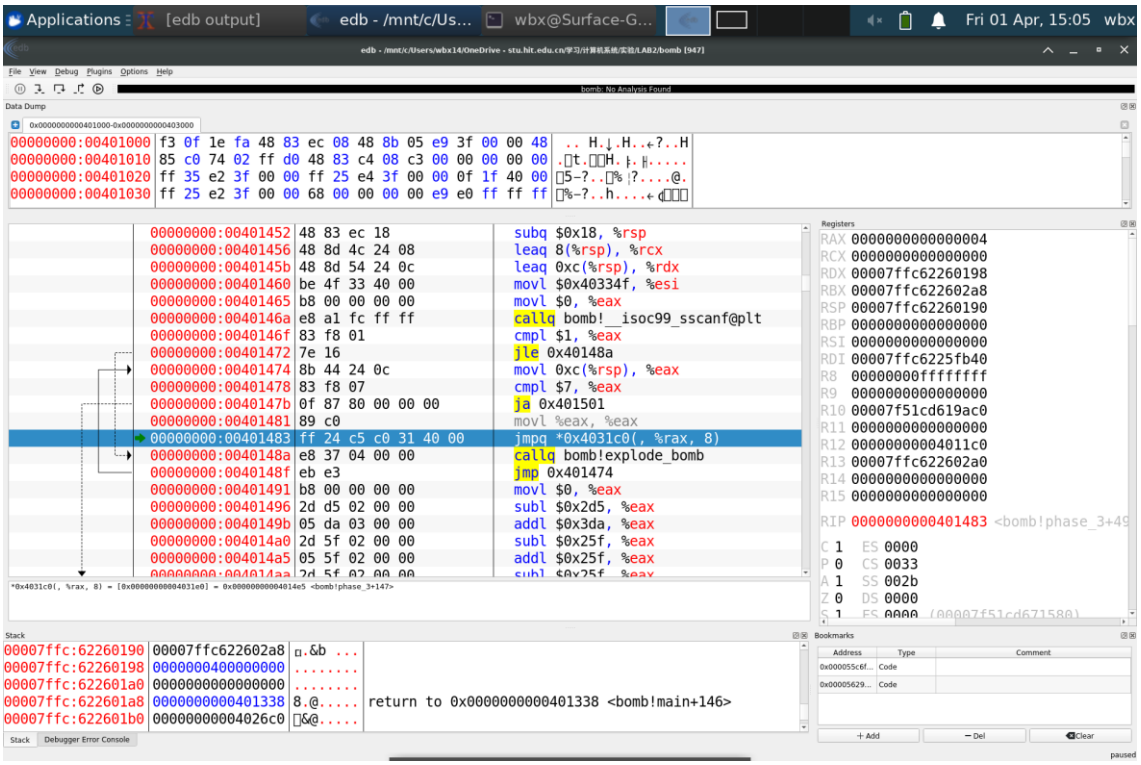
3.3 阶段 3 的破解与分析

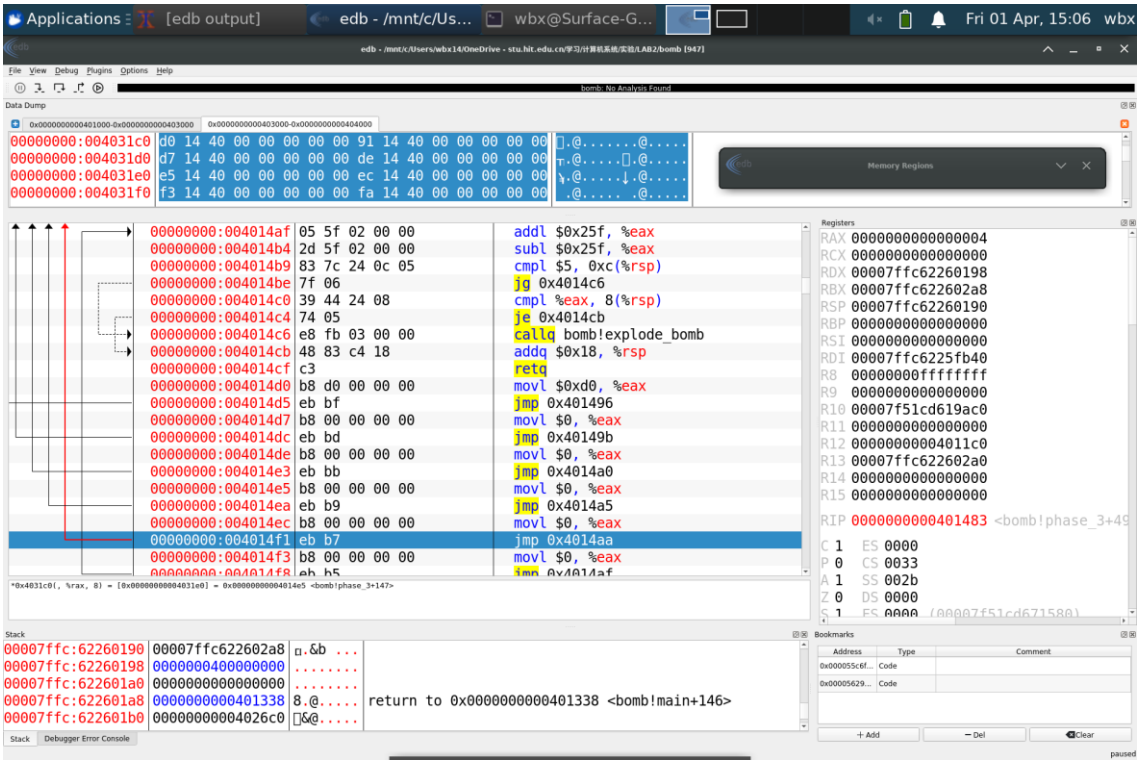
密码如下：4 0（密码不唯一）

破解过程：进入 Phase_3 函数，发现”%d %d”传入了 scanf 函数，那么密码就一定是 2 个数。j 给出了第一个数必须小于 7，之后是一个含有第二个数的地址寻址模式，经过 Memory 查看，发现了跳转列表，显然是一个 switch 语句。经过计算得到了第 2 个数为 4 才能跳转到正确的计算结果，使 eax 不超出范围（大于 5），最后 eax 与第一个数进行比较，相等即可拆除，得到第一个数为 0 的结果。实际上，通过其他 switch 分支也有可能破解，但本次实验未要求（造成密码不唯一）。




Applications
[edb output]
edb - /mnt/c/Us...
wbx@Surface-G...
Fri 01 Apr, 15:06 wbx
00000000:004031c0 d0 14 40 00 00 00 00 00 91 14 40 00 00 00 00 00
00000000:004031d0 d7 14 40 00 00 00 00 00 de 14 40 00 00 00 00 00
00000000:004031e0 e5 14 40 00 00 00 00 00 ec 14 40 00 00 00 00 00
00000000:004031f0 f3 14 40 00 00 00 00 00 fa 14 40 00 00 00 00 00
00000000:00401472 7e 16 jle 0x40148a
00000000:00401474 8b 44 24 0c movl 0xc(%rsp), %eax
00000000:00401478 83 f8 07 cmpl $7, %eax
00000000:0040147b 0f 87 80 00 00 00 ja 0x401501
00000000:00401481 89 c0 movl %eax, %eax
00000000:00401483 ff 24 c5 c0 31 40 00 jmpq *0x4031c0(, %rax, 8)
00000000:0040148a e8 37 04 00 00 callq bomb!explode_bomb
00000000:0040148f eb e3 jmp 0x401474
00000000:00401491 b8 00 00 00 00 movl $0, %eax
00000000:00401496 2d d5 02 00 00 subl $0x2d5, %eax
00000000:0040149b 05 da 03 00 00 addl $0x3da, %eax
00000000:004014a0 2d 5f 02 00 00 subl $0x25f, %eax
00000000:004014a5 05 5f 02 00 00 addl $0x25f, %eax
00000000:004014aa 2d 5f 02 00 00 subl $0x25f, %eax
00000000:004014af 05 5f 02 00 00 addl $0x25f, %eax
00000000:004014b4 2d 5f 02 00 00 subl $0x25f, %eax
00000000:004014b9 83 7c 24 0c 05 cmpl $5, 0xc(%rsp)
00000000:004014be 7f 06 jg 0x4014c6
00000000:004014c0 39 44 24 08 cmpl %eax, 8(%rsp)
00000000:004014c4 74 05 je 0x4014cb
Registers
RAX 0000000000000004
RCX 0000000000000000
RDX 00007ffc62260198
RBX 00007ffc622602a8
RSP 00007ffc62260190
RBP 0000000000000000
RSI 0000000000000000
RDI 00007ffc6225fb40
R8 00000000ffffffff
R9 0000000000000000
R10 00007f51cd619ac0
R11 0000000000000000
R12 00000000004011c0
R13 00007ffc622602a0
R14 0000000000000000
R15 0000000000000000
RIP 0000000000401483 <bomb!phase_3+49
C 1 ES 0000
P 0 CS 0033
A 1 SS 002b
Z 0 DS 0000
Stack
00007ffc:62260190 00007ffc622602a8
00007ffc:62260198 0000000400000000
00007ffc:622601a0 0000000000000000
00007ffc:622601a8 0000000000401338 return to 0x0000000000401338 <bomb!main+146>
00007ffc:622601b0 00000000004026c0
Bookmarks
Address
Type
Comment
Stack
Debugger Error Console
paused

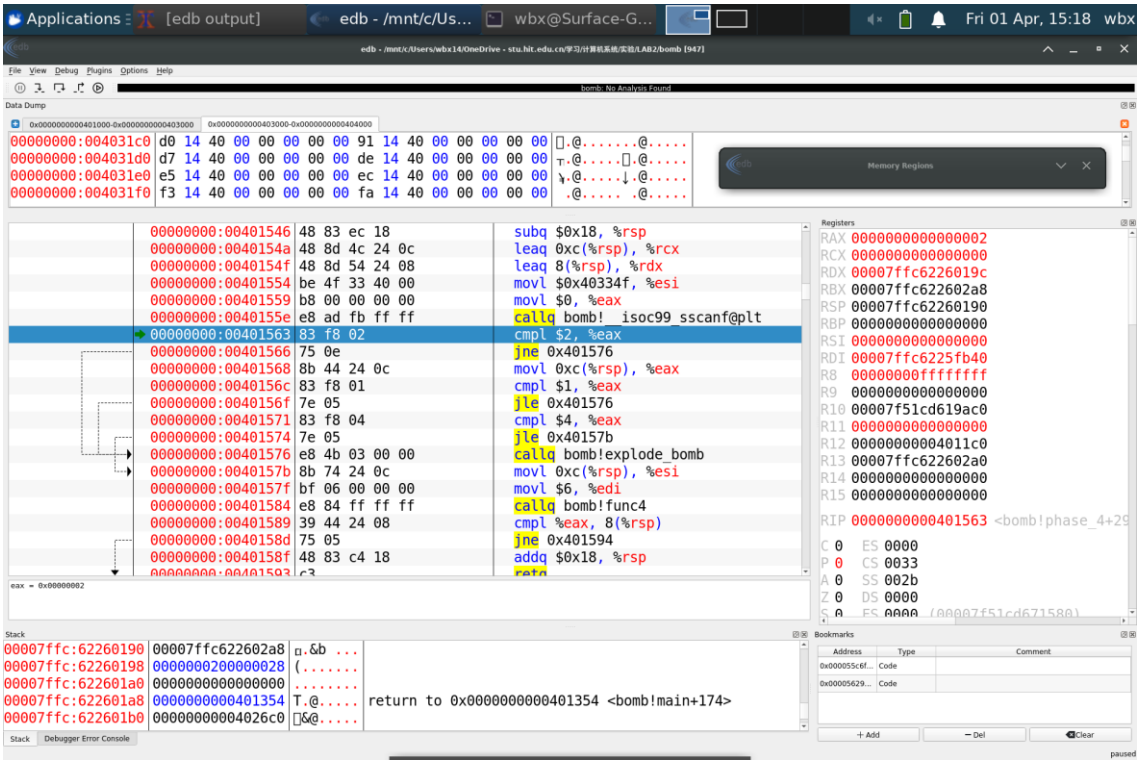

Applications
[edb output]
edb - /mnt/c/Us...
wbx@Surface-G...
Fri 01 Apr, 15:05 wbx
00000000:00401000 f3 0f 1e fa 48 83 ec 08 48 8b 05 e9 3f 00 00 48
00000000:00401010 85 c0 74 02 ff d0 48 83 c4 08 c3 00 00 00 00 00
00000000:00401020 ff 35 e2 3f 00 00 ff 25 e4 3f 00 00 0f 1f 40 00
00000000:00401030 ff 25 e2 3f 00 00 68 00 00 00 00 e9 e0 ff ff ff
00000000:00401452 48 83 ec 18 subq $0x18, %rsp
00000000:00401456 48 8d 4c 24 08 leaq 8(%rsp), %rcx
00000000:0040145b 48 8d 54 24 0c leaq 0xc(%rsp), %rdx
00000000:00401460 be 4f 33 40 00 movl $0x40334f, %esi
00000000:00401465 b8 00 00 00 00 movl $0, %eax
00000000:0040146a e8 a1 fc ff ff callq bomb!__isoc99_sscanf@plt
00000000:0040146f 83 f8 01 cmpl $1, %eax
00000000:00401472 7e 16 jle 0x40148a
00000000:00401474 8b 44 24 0c movl 0xc(%rsp), %eax
00000000:00401478 83 f8 07 cmpl $7, %eax
00000000:0040147b 0f 87 80 00 00 00 ja 0x401501
00000000:00401481 89 c0 movl %eax, %eax
00000000:00401483 ff 24 c5 c0 31 40 00 jmpq *0x4031c0(, %rax, 8)
00000000:0040148a e8 37 04 00 00 callq bomb!explode_bomb
00000000:0040148f eb e3 jmp 0x401474
00000000:00401491 b8 00 00 00 00 movl $0, %eax
00000000:00401496 2d d5 02 00 00 subl $0x2d5, %eax
00000000:0040149b 05 da 03 00 00 addl $0x3da, %eax
00000000:004014a0 2d 5f 02 00 00 subl $0x25f, %eax
00000000:004014a5 05 5f 02 00 00 addl $0x25f, %eax
Registers
RAX 0000000000000004
RCX 0000000000000000
RDX 00007ffc62260198
RBX 00007ffc622602a8
RSP 00007ffc62260190
RBP 0000000000000000
RSI 0000000000000000
RDI 00007ffc6225fb40
R8 00000000ffffffff
R9 0000000000000000
R10 00007f51cd619ac0
R11 0000000000000000
R12 00000000004011c0
R13 00007ffc622602a0
R14 0000000000000000
R15 0000000000000000
RIP 0000000000401483 <bomb!phase_3+49
C 1 ES 0000
P 0 CS 0033
A 1 SS 002b
Z 0 DS 0000
Stack
00007ffc:62260190 00007ffc622602a8
00007ffc:62260198 0000000400000000
00007ffc:622601a0 0000000000000000
00007ffc:622601a8 0000000000401338 return to 0x0000000000401338 <bomb!main+146>
00007ffc:622601b0 00000000004026c0
Bookmarks
Address
Type
Comment
Stack
Debugger Error Console
paused

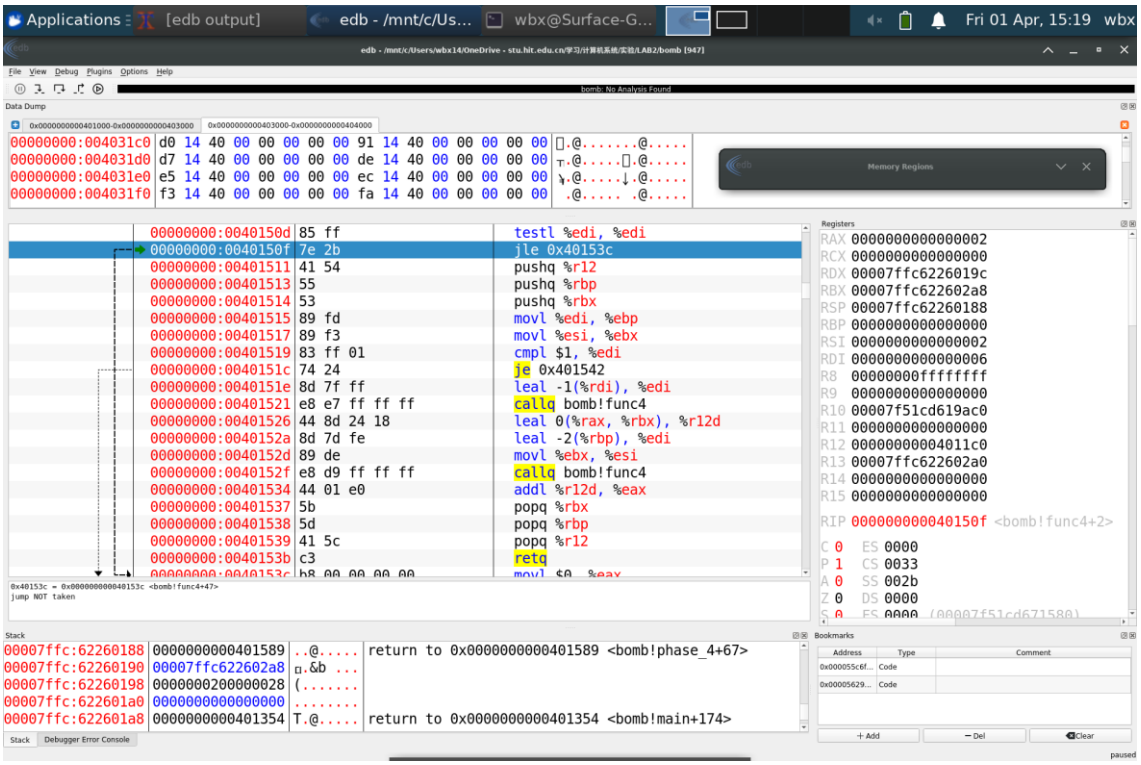


3.4 阶段 4 的破解与分析

密码如下：40 2（密码不唯一）

破解过程：进入 Phase_4 函数，发现”%d %d”传入了 scanf 函数，即密码是两个数。并且发现第二个数介于 2~4 之间（造成密码不唯一），否则就会爆炸。紧接着发现 func4 的递归函数，经过对汇编代码的抽象处理得到了 f(a,b)=f(a-1,b)+f(a-2,b)+b，f(1,b)=b，f(0,b)=b 的数学表达式模型，其中 a 为固定值 6，通过给出数 2（即 b）作为第 2 个参数，得到 f(a,b)的结果返回值 eax 为 0x28，此时有 je 判断数 1 等于 eax，给数 1 为 0x28，即可破解。

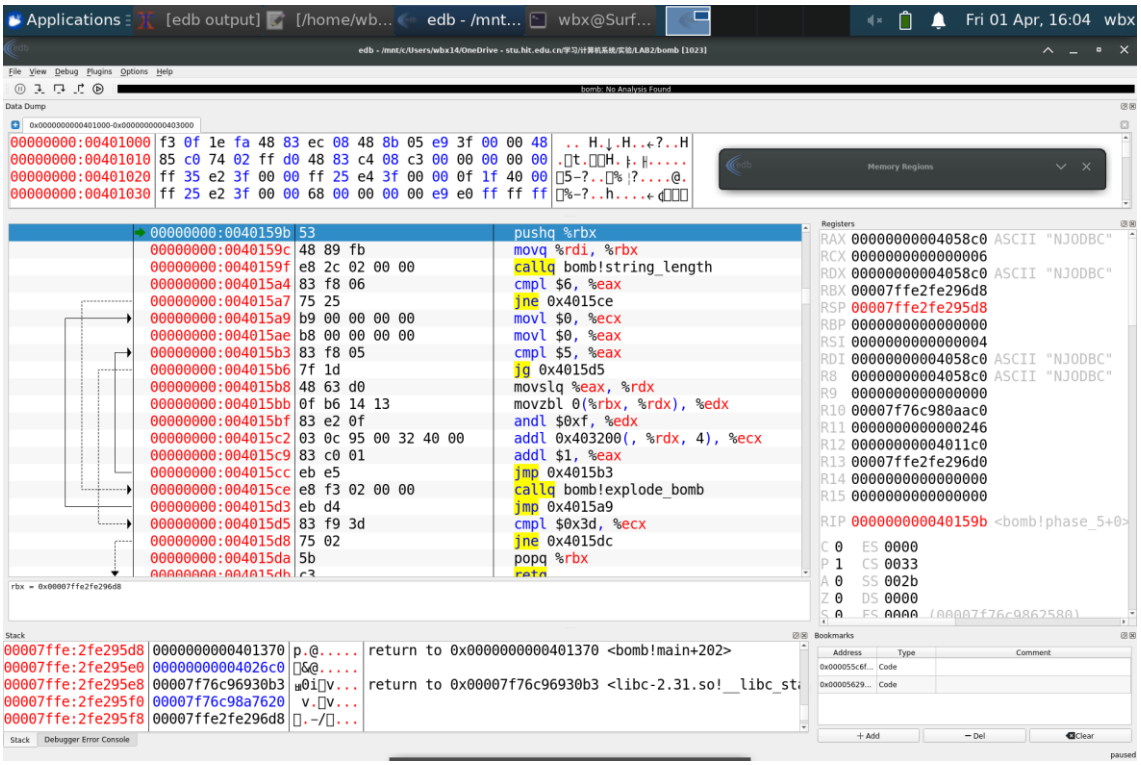


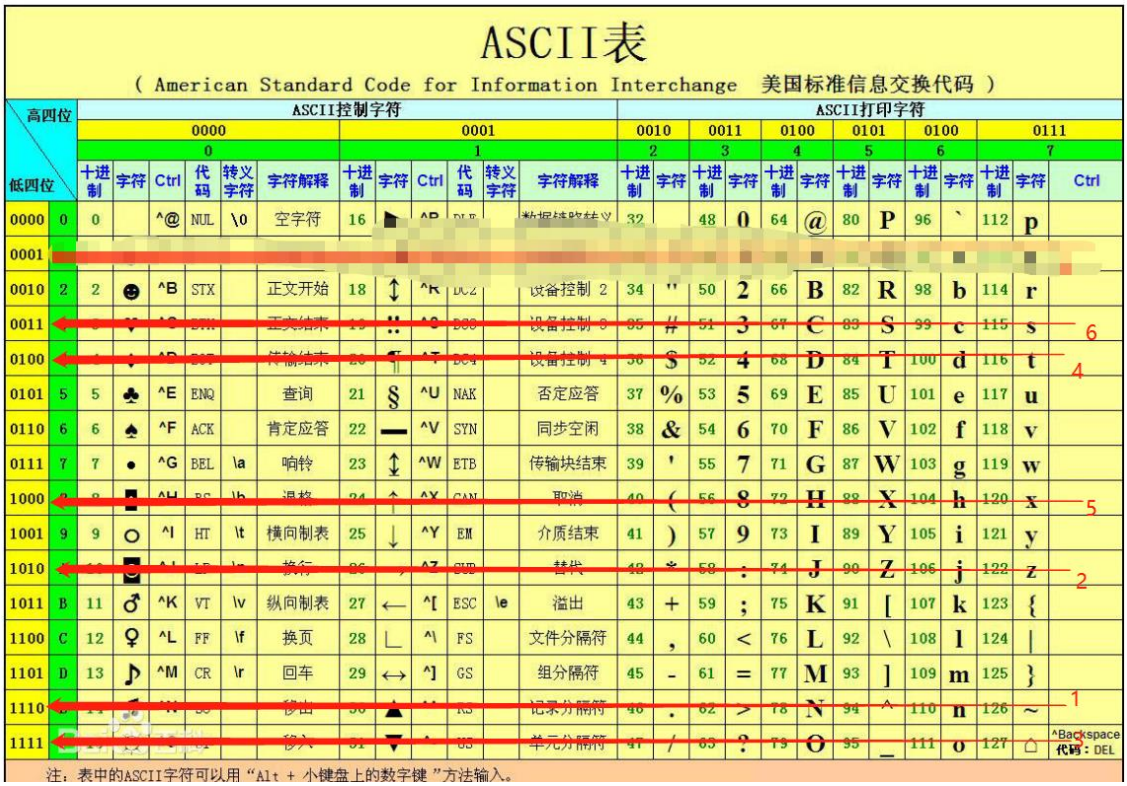


3.5 阶段 5 的破解与分析

密码如下：NJODBC（密码不唯一）

破解过程：进入 Pharse_5 函数，发现了一个判断读入字符串长度为 6 的指令，那么密码就一定是 6 个字符。经过对后续代码的分析，得出最终要让 ecx=0x3d 的结果，而 eax 作为循环计数，总共循环 6 次，初始化 ecx=0，读入的字符串放在 rbx 指定的地址中，rdx 为变址计数，指向字符串索引为 0-5 的字符，edx 作为结果，再与 0xf 相与，保留低 4 位，再寻址 0x403200+rdx*4，得到值加到 ecx 中。此时发现，0x403200 所在空间为 16 个 32 位整数，从地址低位开始依次是：2、A、6、1、C、10、9、3、4、7、E、5、B、8、F、D。总结得到，读入的 6 个字符的低四位进行这样的一个函数映射（f(0)=2，f(1)=A，f(2)=6，……），得到结果相加最终得到 0x3d 即可，因此将 0x3d 分解得到 F+E+D+C+6+1 得到其对应的低四位为 14、10、15、4、2、3。再经过查阅 ASCII 码表得到低四位对应的可打印字符（造成密码不唯一），得到密码。





ASCII表

(American Standard Code for Information Interchange 美国标准信息交换代码)

高四位 / 低四位		ASCII控制字符 0000						ASCII控制字符 0001						ASCII打印字符 0010		ASCII打印字符 0011		ASCII打印字符 0100		ASCII打印字符 0101		ASCII打印字符 0110		ASCII打印字符 0111		
		0						1						2		3		4		5		6		7		
		十进制	字符	Ctrl	代码	转义字符	字符解释	十进制	字符	Ctrl	代码	转义字符	字符解释	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	Ctrl
0000	0	0		^@	NUL	\0	空字符	16						32		48	0	64	@	80	P	96	`	112	p	
0001	1																									
0010	2	2		^B	STX		正文开始	18		^R	DC2		设备控制 2	34	"	50	2	66	B	82	R	98	b	114	r	
0011	3																									
0100	4																									
0101	5	5	♣	^E	ENQ		查询	21	§	^U	NAK		否定应答	37	%	53	5	69	E	85	U	101	e	117	u	
0110	6	6	♠	^F	ACK		肯定应答	22	▬	^V	SYN		同步空闲	38	&	54	6	70	F	86	V	102	f	118	v	
0111	7	7	•	^G	BEL	\a	响铃	23	↨	^W	ETB		传输块结束	39	'	55	7	71	G	87	W	103	g	119	w	
1000	8																									
1001	9	9	○	^I	HT	\t	横向制表	25	↓	^Y	EM		介质结束	41	)	57	9	73	I	89	Y	105	i	121	y	
1010	A																									
1011	B	11	♂	^K	VT	\v	纵向制表	27	←	^[	ESC	\e	溢出	43	+	59	;	75	K	91	[	107	k	123	{	
1100	C	12	♀	^L	FF	\f	换页	28	∟	^\	FS		文件分隔符	44	,	60	<	76	L	92	\	108	l	124	|	
1101	D	13	♪	^M	CR	\r	回车	29	↔	^]	GS		组分隔符	45	-	61	=	77	M	93	]	109	m	125	}	
1110	E																									
1111	F																									^Backspace 代码: DEL

注：表中的ASCII字符可以用“Alt + 小键盘上的数字键”方法输入。

3.6 阶段 6 的破解与分析

密码如下：1 5 4 2 3 6

破解过程：经过对汇编代码的逐步分析，抽象成以下等价的 C 语言代码。实际上就是对一个链表的重新排序，使其值按从大到小排列（给出 6 个索引值）。

```
#include <stdio.h>
typedef struct Table TB;
struct Table {
    int val;
    int id;
    TB *next;
};
int explode(){
    printf("bomb\n");
    exit(8);
    return 0;
}

int main(){
    TB T[6];
    int num[] = {0x3cb,0x25a,0x1c6,0x2ec,0x3b0,0x154};
```

```
        int i=0,j=0;
        for(i=0;i<6;i++){
            T[i].val = num[i];
            T[i].id = i;
            if(i<5)    T[i].next = &T[i+1];
            else T[i].next = NULL;
        }
        int a[6]={0};
        for(i=0;i<6;i++){
            scanf("%d",&a[i]);
        }
        for(i=0;i<6;i++){
            if(a[i]>6) explode();
            for(j=i+1;j<6;j++){
                if(a[i]==a[j]) explode();
             }
        }
        TB *t,*nt[6]={0},*t1;

        for(i=0;i<6;i++){
            t = &T[0];
            for(j=0;j<a[i]-1;j++) t=t->next;
            nt[i] = t;
        }
        for(i=0;i<5;i++){
            t = nt[i];
            t1 = t->next;
            t->next = nt[i+1];
        }
        t = nt[5];
        t->next = NULL;
        t = nt[0];
        for(i=0;i<5;i++){
            if((t->val)<(t->next->val)) explode();
            t = t->next;
        }
        return 0;
}
```

3.7 阶段 7 的破解与分析(隐藏阶段)

密码如下：未分析

破解过程：未分析

第 4 章 总结

4.1 请总结本次实验的收获

学会并了解了许多 C 语言语句在汇编语言中的形式。
学会使用 EDB\GDB 等调试工具对可执行文件进行调试或分析。
了解一些逆向工程的概念与知识。

4.2 请给出对本次实验内容的建议

无

注：本章为酌情加分项。

参考文献

[1] ASCII_百度百科 (baidu.com)