《计算机组成原理实验》

实验报告

（实验一）

学 院 名 称 ：	数据科学与计算机学院
专业（班级）：	18 计教学 3 班
学 生 姓 名 ：	谢俊杰
学　　号	18340181
时　　间 ：	2019　年　10　月　2　日

成 绩 ：

实验一 ： X86汇编基础二进制炸弹

一. 实验目的

1. 了解并熟悉x86汇编语言，能通过汇编语言反推出高级语言语句。
2. 熟悉gdb调试工具，掌握用gdb反汇编调试代码。

二. 实验内容

程序bomb中有6个关卡和1个隐藏关卡，每个关卡都需要通过对程序反汇编出来的汇编语言进行分析后，输入正确的答案则炸弹解除，否则将会引爆炸弹。该实验要求熟悉x86的汇编语言并进行分析，找到相应使炸弹引爆的地方并避开和成功跳转到解除炸弹的条件，得到正确答案，并鼓励通过汇编语言代码找出隐藏关卡和解除隐藏炸弹。

三. 实验器材

PC机一台，装有Linux操作系统的虚拟机一套。

四. 实验过程与结果

4.1 第一关

经过阅读phase_1的汇编代码后，知道了本关代码是在比较两个字符串是否相同，而由下面代码可知关卡1调用了一个判断两个字符串是否相等的函数，但需要注意的是，若两字符串相等则该函数返回0，反之返回1。

```
push   $0x804a25c
pushl  0x8(%ebp)
call   8049081 <strings_not_equal>
```

而传递进函数有两个参数，一个是在地址0x804a25c中，直觉告诉我答案极有可能就是它，故使用了gdb工具对上地址进行查询，发现为以下字符串，另一个参数则为由用户去输入的。

```
(gdb) x/s 0x804a25c
0x804a25c:      "When a problem comes along, you must zip it!"
```

由下面代码可知，若%eax不为0时会跳转到引爆炸弹的函数，故%eax只能为0。

```
test   %eax,%eax    //test为将两数按位相与
jne    8048b76 <phase_1+0x1c>//即仅%eax = 0时不跳转
```

而函数返回值一般是存储在寄存器%eax中的，即字符串相等。所以，到此可以分析出关卡1的答案为 When a problem comes along, you must zip it! 。

而这关用C++代码表示为

```
void phase_1(string s)
{
    string s0 = "When a problem comes along, you must zip it!";
    if (s == s0)
        return ;
    else
        explode_bomb();
}
```

经验证也为正确答案。虽然这关是有TA在实验课上教过且有实验指导的，但当我解出正确的答案并成功接触炸弹的时候还是很开心的，并极有兴趣地期待着第二关。

```
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
When a problem comes along, you must zip it!
Phase 1 defused. How about the next one?
```

4.2 第二关

来到第二关，经过阅读第二关的代码，发现在开始时便调用了一个read_six_number的函数，据TA及函数名提示，该函数应是一个读取6个数的函数，经查看该函数的具体代码后，发现如下代码，且push的又是一个地址值，使用gdb工具以字符串形式对上地址进行查看，发现是"%d %d %d %d %d %d"，结合下面call的scanf可知确是读取6个数值。

```
push   $0x804a4b9
pushl  0x8(%ebp)
call   8048810 <__isoc99_sscanf@plt>
```

```
(gdb) x/s 0x804a4b9
0x804a4b9:      "%d %d %d %d %d %d"
```

然后，其后的三句代码是确定-0x24(%ebp)即第一个输入的数的范围<=5，对接下来的代码分析，特别是留意跳转指令，可发现这是一个循环。但是在分析这个循环的时候，我被%cl这个寄存器卡了很久，当时我完全不明白这个寄存器是从哪来的，后来经过百度后，

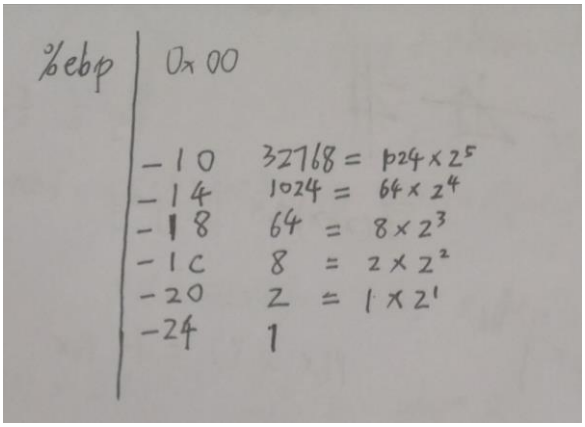
发现%cl是%ecx寄存器的低8位，%ch为其高八位，在明白这个寄存器的含义后，真的是“豁然开朗”，循环的栈帧过程也能明白是什么样的循环过程了。经过循环完成的是第一个数的范围是0到5，而后5个数分别是由前一个数左移1-5位得到，因为左移i位相当于乘以2^i，所以后5个数分别是由其前一个数乘以2^i（i为%ecx的低8位）得到。而循环块最后比较的是%eax的值和相应第i个数的值，不相等则引爆炸弹，故该两数只能相等，即输入的六个数必须符合循环中的条件。

```
add    $0x1,%ebx     //%ebx+=1
cmp    $0x6,%ebx     //循环5次
je     8048bd2 <phase_2+0x55>

mov    -0x28(%ebp,%ebx,4),%eax
mov    %eax,-0x2c(%ebp)
mov    %ebx,%ecx          //%ecx = %ebx
shl    %cl,%eax      //%eax左移%cl位,%cl是%ecx的低8位
cmp    %eax,-0x24(%ebp,%ebx,4)
je     8048bb2 <phase_2+0x35>        //%eax == -0x24(%ebp,%ebx,4)
```

用C++语言表示则为

```
int a[6];
cin >> a[0];
if (a[0] > 5 || a[0] < 0)
    explode_bomb();
else {
    for (int i = 1; i < 6; i++) {
        a[i] = a[i - 1] * pow(2, i);
    }
}
```

由于第一个数的范围是0到5，故在这假设为1，附上该循环的栈帧图：



故输入的6个数应为1、2、8、64、1024、32768，但这一关的答案并不唯一，第一个数范围为0-5，在确定第一个数后，其余5个数由上述方法可得出，这些组合亦是正确答案，同时为避免重复输入已解除炸弹的关卡，用一个answer.txt的文件储存答案：

```
eagle@ubuntu:~/Desktop/bomb45$ ./bomb answer.txt
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Phase 1 defused. How about the next one?
0 0 0 0 0 0
That's number 2.  Keep going!
^CSo you think you can stop the bomb with ctrl-c, do you?
Well...OK. :-)
eagle@ubuntu:~/Desktop/bomb45$ ./bomb answer.txt
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Phase 1 defused. How about the next one?
1 2 8 64 1024 32768
That's number 2.  Keep going!
^CSo you think you can stop the bomb with ctrl-c, do you?
Well...OK. :-)
```

在这一关中，我明白了在高级语言的循环板块中用汇编语言编写的方法，也就是通过地址间的跳转和跳转的条件去作为循环的依据。

4.3 第三关

来到phase_3，在大致阅读汇编代码后，发现了一个在上一关出现的scanf函数，因为scanf函数是从栈中依次读取的，故从该语句往上寻找可知道要输入的参数的类型和存储的地址，同时在一个push中也发现了一个可疑的地址0x804a2b2，利用gdb以字符串形式对上地址进行查看，发现是"%d %c %d"，即说明要输入的参数相对应的类型。且对应参数的地址依次为-14、-15、-10。

```
push   %ebp
mov    %esp,%ebp
sub    $0x24,%esp   //开24H空间
mov    %gs:0x14,%eax    //eax = 14H
mov    %eax,-0xc(%ebp)
xor    %eax,%eax    //eax = 0
lea    -0x10(%ebp),%eax     //三%d
push   %eax
lea    -0x15(%ebp),%eax     //二%c
push   %eax
lea    -0x14(%ebp),%eax         //第一个参数%d
push   %eax
push   $0x804a2b2
pushl  0x8(%ebp)
call   8048810 <__isoc99_sscanf@plt>
```

```
(gdb) x/s 0x804a2b2
0x804a2b2:      "%d %c %d"
```

接下来%eax是scanf的返回值，这里将其与2比较，根据比较结果和跳转指令可知eax>2即%eax输入的参数应为3个，与上述分析相符。

```
add     $0x20,%esp
cmp     $0x2,%eax         //eax为scanf的返回值
jle     8048c2e <phase_3+0x46>        //eax>2, eax<=2爆炸
```

下面两句代码说明第一个数小于等于7，该处跳转的地址是引爆炸弹的函数。

```
cmpl    $0x7,-0x14(%ebp)
ja      8048cd8 <phase_3+0xf0>    //  (i)-0x14(%ebp)<=7
```

然后将第一个数传给%eax，并跳转到一个地址，而这个地址是由一个指针指向的一个地址值，利用gdb查看该指针所指向的地址，为0x8048c35。并由上述第一个数的范围假设第一个数为0时，跳转到0x8048c35，此时%eax=71H，接下来第三个数根据比较，可得第三个数等于395H，即917。

```
8048c24:   8b 45 ec              mov    -0x14(%ebp),%eax
8048c27:   ff 24 85 c4 a2 04 08  jmp    *0x804a2c4(,%eax,4)      //j 0x8048c35 + 4i
8048c2e:   e8 82 06 00 00        call   80492b5 <explode_bomb>
8048c33:   eb e5                 jmp    8048c1a <phase_3+0x32>

8048c35:   b8 71 00 00 00        mov    $0x71,%eax           //i = 0跳转到此
8048c3a:   81 7d f0 95 03 00 00  cmpl   $0x395,-0x10(%ebp)
8048c41:   0f 84 9b 00 00 00     je     8048ce2 <phase_3+0xfa>   //-0x10(%ebp) = 395H
8048c47:   e8 69 06 00 00        call   80492b5 <explode_bomb>
```

```
(gdb) p/x *0x804a2c4
$1 = 0x8048c35
```

再次跳转后，可判断第二个数与%al相等，%al是%eax的低8位，由上面可知第二个数为71H转换为char类型为字符q，所以该关卡的答案为0 q 917，经验证为正确答案。

```
cmp     %al,-0x15(%ebp)
je      8048cec <phase_3+0x104>       //%al == -0x15(%ebp)
call    80492b5 <explode_bomb>
mov     -0xc(%ebp),%eax
xor     %gs:0x14,%eax
jne     8048cfa <phase_3+0x112>
leave
ret
```

```
eagle@ubuntu:~/Desktop/bomb45$ ./bomb answer.txt
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Phase 1 defused. How about the next one?
That's number 2.  Keep going!
0 q 917
Halfway there!
^CSo you think you can stop the bomb with ctrl-c, do you?
Well...OK. :-)
```

用伪代码表示为

```
int eax;
    int a, c;
    char b;
    eax = scanf("%d%c%d", &a, &b, &c);
    if (eax <= 2)
        explode_bomb();
    else {
        if (a > 7)
            explode_bomb();
        else {
            switch (a) {
                case 0: c = 0x395; eax = 0x71; b = eax; break;
                ...
            }
        }
    }
```

第三关是switch的汇编代码，而这里的switch是以数为选择基础，而每个数字在switch的case中是以地址有关的数，利用该数进行地址的跳转实现选择。

4.4 第四关

phase_4的汇编代码比前三关的都要长，而且调用了一个fun4的函数，我感到压力巨大，而且在前三关的分析后也感到有点吃力。但是发现开始也是熟悉的scanf函数和push一个地址值，利用gdb查看发现是"%d %d"，即要输入两个数，第一个参数在地址-10，第二个参数在-14。同时scanf的返回值%eax与2比较说明必须要输入两个数，否则引爆炸弹，这是第一个避开炸弹的条件。

```
push   %ebp
mov    %esp,%ebp
sub    $0x18,%esp
mov    %gs:0x14,%eax
mov    %eax,-0xc(%ebp)
xor    %eax,%eax
lea    -0x14(%ebp),%eax      //第二个参数j
push   %eax
lea    -0x10(%ebp),%eax      //第一个参数i
push   %eax
push   $0x804a4c5         //%d  %d
pushl  0x8(%ebp)          //stdin
call   8048810 <__isoc99_sscanf@plt>         //scanf
add    $0x10,%esp
cmp    $0x2,%eax                  //输入两个数
jne    8048d83 <phase_4+0x39>
```

接下来的代码中，令%eax等于第二个参数设为j，操作并比较，说明第二个参数必须<=4，但是，由于比较是无符号的比较，所以第二个参数必须>=2，否则在这个范围外会爆炸。

```
mov    -0x14(%ebp),%eax
sub    $0x2,%eax          //%eax = j - 2
cmp    $0x2,%eax          //%eax >= 2 无符号比较
jbe    8048d88 <phase_4+0x3e>    //j <= 4
call   80492b5 <explode_bomb>
```

然后就是调用fun4了，同时传递2个参数，第一个是立即数7，第二个是j。

```
sub    $0x8,%esp
pushl  -0x14(%ebp)       //参2 j
push   $0x7              //参1 7
call   8048cff <func4>
```

而且，再往后的代码中注意到，第一个数为fun4的返回值，故接下来就是分析fun4了。

```
add    $0x10,%esp
cmp    %eax,-0x10(%ebp)
je     8048da2 <phase_4+0x58>    //-0x10(%ebp) == %eax
call   80492b5 <explode_bomb>
mov    -0xc(%ebp),%eax
xor    %gs:0x14,%eax
jne    8048db0 <phase_4+0x66>
leave
ret
call   8048790 <__stack_chk_fail@plt>
```

大致阅读fun4的代码后，发现它也调用自己，我也就想到了递归。那肯定是比较复杂的了，所以我首先逐行分析代码，发现先是将参数1赋给%esi，参数2赋给%edi，也发现离

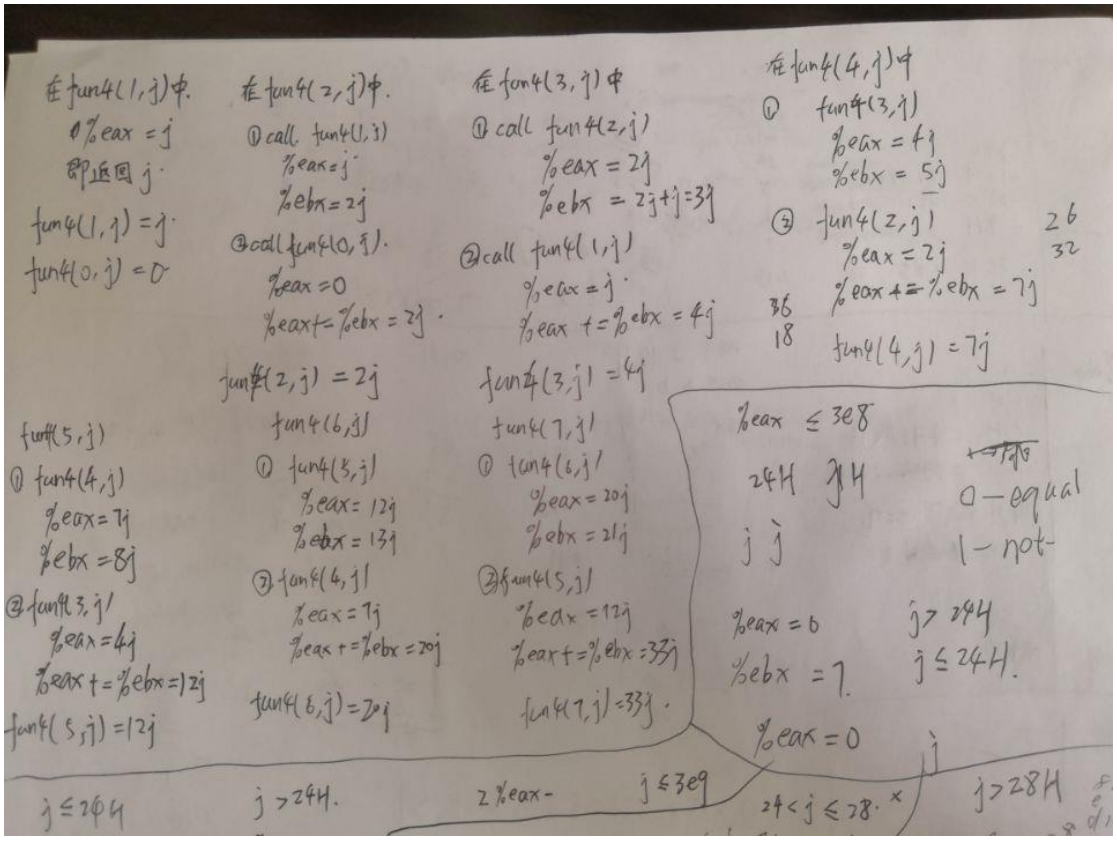
开fun4的条件有二：一是%esi<=0，二是%esi=1。若不符合条件则跳转到下面的代码进行递归。那么我们可以知道该函数的递归基是0和1，在这也可以求出%esi=0时，fun4返回0；%esi=1时，fun4返回j。

```
push   %ebp
mov    %esp,%ebp
push   %edi          //保留现场
push   %esi
push   %ebx

sub    $0xc,%esp
mov    0x8(%ebp),%esi        //参数1 %esi = 7
mov    0xc(%ebp),%edi        //参数2 %edi = j
mov    $0x0,%eax             //%eax = 0
test   %esi,%esi
jle    8048d1e <func4+0x1f>      //%esi<=0则跳转离开fun4
mov    %edi,%eax             //%eax = %edi = j
cmp    $0x1,%esi
jne    8048d26 <func4+0x27>      //%esi = 1离开fun4

lea    -0xc(%ebp),%esp

pop    %ebx          //恢复现场
pop    %esi
pop    %edi
pop    %ebp
ret
```

再下来就是递归的过程了，且有两次递归，分别是传递第一个参数-1、第二个参数不变和第一个参数-2、第二个参数不变。在此我在这用从2开始迭代的方法，求出到7的fun4的返回值。

```
sub    $0x8,%esp          //开辟2个数的空间
push   %edi               //参数2 %edi = j
lea    -0x1(%esi),%eax
push   %eax               //参数1 %eax -= 1

call   8048cff <func4>
add    $0x8,%esp
lea    (%eax,%edi,1),%ebx
push   %edi               //参数2 %edi = j
sub    $0x2,%esi
push   %esi               //参数1 %esi -= 2
call   8048cff <func4>
add    $0x10,%esp
add    %ebx,%eax
jmp    8048d1e <func4+0x1f>
```

附上迭代分析fun4的图：



由于phase_4主体主要说明参数范围，故在这只写出fun4的递归代码

```
int i, j;
scanf("%d%d", &i, &j);
if (j < 2 || j > 4)
    explode_bomb();
if (i != fun4(7, j))
    explode_bomb();
int fun4(int a, int j) {
```

```
        int esi = a, edi = j, eax = 0;
        if (esi != 0) {
            eax = edi;
            if (esi == 1) {
                return eax;
            }
            eax = esi - 1;
            eax = fun4(esi, j);
            ebx = eax + edi;
            esi -= 2;
            eax = fun4(esi, j);
            eax += ebx;
            return eax;
        }
    }
```

所以可知，要输入的第一个数是第二个数的33倍，而结合第二个数的范围是2到4，所以输入可为66 2、99 3、132 4，经验证为正确答案，成功解除炸弹。

```
eagle@ubuntu:~/Desktop/bomb45$ ./bomb answer.txt
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Phase 1 defused. How about the next one?
That's number 2.  Keep going!
Halfway there!
66 2
So you got that one.  Try this one.
^CSo you think you can stop the bomb with ctrl-c, do you?
Well...OK. :-)
```

```
eagle@ubuntu:~/Desktop/bomb45$ ./bomb answer.txt
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Phase 1 defused. How about the next one?
That's number 2.  Keep going!
Halfway there!
99 3
So you got that one.  Try this one.
^CSo you think you can stop the bomb with ctrl-c, do you?
Well...OK. :-)
```

```
eagle@ubuntu:~/Desktop/bomb45$ ./bomb answer.txt
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Phase 1 defused. How about the next one?
That's number 2.  Keep going!
Halfway there!
132 4
So you got that one.  Try this one.
^CSo you think you can stop the bomb with ctrl-c, do you?
Well...OK. :-)
```

Rhythmbox

第四关是递归的汇编，而这关我了解了一些函数在汇编语言的实现，主要利用栈帧调整指针来实现函数的调用，还有函数参数的传递关系（也是栈）和返回地址。

4.5 第五关

经阅读第五关的代码后，发现没有了前面三关熟悉的scanf函数，首先对这一关要输入的数是什么类型感到困惑，然后当看到push后调用一个名为string_length的函数，显然这个函数是返回字符串长度的，那么可知需要输入的是一个字符串。而通过函数返回值和立即数6进行比较，发现该字符串长度为6，否则将会引爆炸弹。

```
push   %ebp
mov    %esp,%ebp
push   %ebx
sub    $0x20,%esp
mov    0x8(%ebp),%ebx
mov    %gs:0x14,%eax
mov    %eax,-0xc(%ebp)
xor    %eax,%eax
push   %ebx
call   804905f <string_length>
add    $0x10,%esp
cmp    $0x6,%eax              //%eax = 6
jne    8048e24 <phase_5+0x6f>       //%ebx的长度为6
```

以下代码将%eax赋为0，并在一系列操作后%eax+=1和与6比较，可发现这是一个循环，由于%ebx为输入的字符串，故%edx=%ebx[%eax]。将%edx和fH相与是取%edx低四位放进%edx中的意思。而0x804a2e4这个地址是一个可疑的地址，于是拿起gdb先是以字符串形式查看，并没有发现什么有效的信息，但注意到地址的括号里写着array，也就是一个数组，也确是一个字符数组，故以1字节的形式去查看该字符数组，储存信息如下图。

而可知%edx=数组[%edx]，同时取%edx的低八位即字符数组的相应内容存进-0x13(%ebp,%eax,1)中，这个循环就是将%eax从0循环至5，即将字符串的6个字符均遍历了。

```
mov    $0x0,%eax            //%eax = 0

movzbl (%ebx,%eax,1),%edx   //%edx = %ebx[0]
and    $0xf,%edx            //取%edx的低四位
movzbl 0x804a2e4(%edx),%edx //0x804a2e4为数组首地址，%edx为下标，%edx = 数组[%edx]元素
mov    %dl,-0x13(%ebp,%eax,1)   //取%edx的低8位存进-0x13(%ebp,%eax,1)中
add    $0x1,%eax        //%eax += 1
cmp    $0x6,%eax        //循环直到%eax = 6退出   (0-5)
jne    8048ddd <phase_5+0x28>
```

```
(gdb) x/s 0x804a2e4
0x804a2e4 <array.3032>: "maduiersnfotvbylSo you think you can stop the bomb wit
h ctrl-c, do you?"
```

```
(gdb) x/16u 0x804a2e4
0x804a2e4 <array.3032>: 109     97      100     117     105     101     114
115
0x804a2ec <array.3032+8>:       110     102     111     116     118     98
121     108
```

接着跟phase_1基本相同，调用strings_not_equal这个函数将-13为首地址的字符串与0x804a2bb比较，相等即成功解除炸弹。用gdb查看这个地址，发现是一个长度为6的字符串"bruins"。那么该字符串就是上述循环中用字符数组依次选出来组合而成的。由字符数组的内容可知下标分别为7、8、4、3、6、13，用十六进制表示为7、8、4、3、6、D，即输入字符串各字符的低4位，由ASCII码表可得高四位可为6或7，选取任一高四位加上低四位得到得字符组合即为要输入字符串，在这我选择7，输入即为mfcdhg，经验证为正确答案。其实在前四关的基础上，阅读分析汇编代码的能力也逐渐增强，所以这一关很快就通过了。

```
movb   $0x0,-0xd(%ebp)
sub    $0x8,%esp
push   $0x804a2bb       //“bruins”
lea    -0x13(%ebp),%eax
push   %eax
call   8049081 <strings_not_equal>  //equal即-0x13(%ebp)为首地址字符串 = “bruins”
add    $0x10,%esp
test   %eax,%eax
jne    8048e2b <phase_5+0x76>   //%eax = 0即equal
mov    -0xc(%ebp),%eax
xor    %gs:0x14,%eax
jne    8048e32 <phase_5+0x7d>
mov    -0x4(%ebp),%ebx
leave
ret
```

```
(gdb) x/s 0x804a2bb
0x804a2bb:      "bruins"
```

```
eagle@ubuntu:~/Desktop/bomb45$ ./bomb answer.txt
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Phase 1 defused. How about the next one?
That's number 2.  Keep going!
Halfway there!
So you got that one.  Try this one.
mfcdhg
Good work!  On to the next...
```

本关主要伪代码为：

```
char a[6];
char array[17] = "maduiersnfotvbyl";
char* s = "bruins";
for (int i = 0; i < 6; i++) {
    for (int j = 0; j < 16; j++) {
```

```
            if (array[j] == s[i]) {
                a[i] = j;
                a[i] += 7 * 16;
            }
        }
    }
```

第五关是关于数组的汇编代码，与C语言的数组差不多，都是用数组的首地址和下标来确定数组成员的具体内容，同时也更了解C语言数组是一片连续地址的存储空间。

4.6 第六关

来到第六关，意味着离结束也不远了。在这一关的代码中，发现了一个老朋友函数read_six_number，同样，也是要输入六个数。而该6个数储存在-3c、-38、-34、-30、-2c、-28的地址中。

```
lea     -0x3c(%ebp),%eax
push    %eax
pushl   0x8(%ebp)
call    80492f5 <read_six_numbers>    //读取6个数
```

然后是一个嵌套循环，是遍历6个数并判断6个数之间各不相等的，若存在相等的两个数，则引爆炸弹。

```
add    $0x10,%esp
mov    $0x0,%esi                 //%esi = 0

mov    -0x3c(%ebp,%esi,4),%eax   //%eax = %esi * 4 - 3c
sub    $0x1,%eax                 //%eax -= 1
cmp    $0x5,%eax
ja     8048e76 <phase_6+0x3f>    //%eax <= 5 %eax > 5则炸

add    $0x1,%esi                 //%esi += 1
cmp    $0x6,%esi
je     8048ec3 <phase_6+0x8c>    //%esi == 6则跳转
mov    %esi,%ebx                 //%ebx = %esi
jmp    8048e85 <phase_6+0x4e>
call   80492b5 <explode_bomb>
jmp    8048e6a <phase_6+0x33>

add    $0x1,%ebx             //%ebx += 1
cmp    $0x5,%ebx
jg     8048e5e <phase_6+0x27>    //%ebx > 5则跳转即%ebx = 6
mov    -0x3c(%ebp,%ebx,4),%eax   //%eax = %ebx * 4 - 3c
cmp    %eax,-0x40(%ebp,%esi,4)
jne    8048e7d <phase_6+0x46>        //%eax != %esi * 4 - 40
```

此处伪代码为：

```
int a[6];
for (int i = 0; i < 6; i++) {
    scanf("%d", &a[i]);
}
for (int i = 0; i < 5; i++) {
    for (int j = i + 1; j < 6; j++) {
        if (a[i] == a[j])
            explode_bomb();
    }
}
```

接下来的代码中发现一个地址值0x804c154，依然是用gdb查看，发现是一个数值，且在括号内写有node，我猜想是链表，立即查看该地址值+4i所表示的值，发现有数值，有表示第几个节点的标号，有指向下一个节点的地址，那么可确定这一关是关于链表的。

```
mov    $0x804c154,%edx           //$0x804c154 = 1be
```

```
(gdb) x 0x804c154
0x804c154 <node1>:      0x000001be
```

```
(gdb) x 0x804c154
0x804c154 <node1>:      0x000001be
(gdb) x 0x804c158
0x804c158 <node1+4>:    0x00000001
(gdb) x 0x804c15c
0x804c15c <node1+8>:    0x0804c160
(gdb) x 0x804c160
0x804c160 <node2>:      0x00000278
(gdb) x 0x804c164
0x804c164 <node2+4>:    0x00000002
(gdb) x 0x804c168
0x804c168 <node2+8>:    0x0804c16c
(gdb) x 0x804c16c
0x804c16c <node3>:      0x00000371
0x804c170 <node3+4>:    0x00000003
(gdb) x 0x804c174
0x804c174 <node3+8>:    0x0804c178
(gdb) x 0x804c178
0x804c178 <node4>:      0x000001f3
(gdb) x 0x804c17c
0x804c17c <node4+4>:    0x00000004
(gdb) x 0x804c180
0x804c180 <node4+8>:    0x0804c184
(gdb) x 0x804c184
0x804c184 <node5>:      0x00000088
(gdb) x 0x804c188
0x804c188 <node5+4>:    0x00000005
(gdb) x 0x804c18c
0x804c18c <node5+8>:    0x0804c190
(gdb) x 0x804c190
0x804c190 <node6>:      0x00000160
(gdb) x 0x804c194
0x804c194 <node6+4>:    0x00000006
(gdb) x 0x804c198
0x804c198 <node6+8>:    0x00000000
```

继续阅读代码，发现这段代码也是一个嵌套循环，根据跳转指令的跳转条件，可判断出这个循环是使输入的6个数匹配链表的序号并使地址-0x24 - -0x10表示相应链表节点所表示的数。

```
mov    0x8(%edx),%edx        //%edx = (%edx) + 8
add    $0x1,%eax             //%eax += 1
cmp    %ecx,%eax
jne    8048e96 <phase_6+0x5f>   //%eax != %ecx则跳转

mov    %edx,-0x24(%ebp,%esi,4)  //%esi * 4 - 24 = %edx
add    $0x1,%ebx             //%ebx += 1
cmp    $0x6,%ebx
je     8048eca <phase_6+0x93>   //%ebx = 6则跳转

mov    %ebx,%esi                //%esi = %ebx
mov    -0x3c(%ebp,%ebx,4),%ecx  //%ecx = %ebx * 4 - 3c
mov    $0x1,%eax                //%eax = 1
mov    $0x804c154,%edx          //$0x804c154 = 1be
cmp    $0x1,%ecx
jg     8048e96 <phase_6+0x5f>   //%ecx > 1则跳转
jmp    8048ea0 <phase_6+0x69>
```

下面一堆的mov指令都是链表地址间的赋值，这段代码不用细看，细看感觉还会把自己绕晕，我们需要知道的，最主要的是跳转指令和跳转条件，我们注意到，%esi=5，并且在%esi=0时结束循环，也就是这个循环依旧在围绕链表做操作，其次就是%eax和(%ebx)作比较，且%eax必须大于等于(%ebx)，这时我们往前看，发现这是链表地址间的比较，由上面gdb得出的链表内容可知，也是链表储存的数之间的比较，也就是这个循环就是说明这些在-24 - -10间地址的排列是按照数值降序排列的，那么链表节点的标号也是按此排序。

```
mov    -0x24(%ebp),%ebx      //%ebx = -24
mov    -0x20(%ebp),%eax      //%eax = -20
mov    %eax,0x8(%ebx)        //%ebx + 8 = %eax
mov    -0x1c(%ebp),%edx      //%edx = -1c
mov    %edx,0x8(%eax)        //%eax + 8 = %edx
mov    -0x18(%ebp),%eax      //%eax = -18
mov    %eax,0x8(%edx)        //%edx + 8 = %eax
mov    -0x14(%ebp),%edx      //%edx = -14
mov    %edx,0x8(%eax)        //%eax + 8 = %edx
mov    -0x10(%ebp),%eax      //%eax = -10
mov    %eax,0x8(%edx)        //%edx + 8 = %eax
movl   $0x0,0x8(%eax)        //%eax + 8 = 0
mov    $0x5,%esi             //%esi = 5
jmp    8048f01 <phase_6+0xca>
```

```
mov    0x8(%ebx),%ebx        //%ebx = %ebx + 8
sub    $0x1,%esi             //%esi -= 1
je     8048f11 <phase_6+0xda>   //%esi = 0结束

mov    0x8(%ebx),%eax        //%eax = %ebx + 8
mov    (%eax),%eax           //%eax = (%eax)
cmp    %eax,(%ebx)
jge    8048ef9 <phase_6+0xc2>   //%eax >= (%ebx)
call   80492b5 <explode_bomb>
jmp    8048ef9 <phase_6+0xc2>
```

此处链表的降序排列的高级语言代码为：

```
struct node{
    int num;
    int seq;
    node* next;
};
node* head;
node* p = head->next;
node* pstart, *pend = head, *temp, *pre;
pstart->next = head;
while (p != NULL) {//降序排列
    temp = pstart->next, pre = pstart;
    while (temp != p && p->num <= temp->num) {
        temp = temp->next;
        pre = pre->next;
    }
    if (temp == p)
        pend = p;
    else {
        pend->next = p->next;
        p->next = temp;
        pre->next = p;
    }
    p = pend->next;
}
head = pstart->next;
```

根据链表内容降序排序后，得出序号的排列为３２４１６５，经验证为正确答案。

```
eagle@ubuntu:~/Desktop/bomb45$ ./bomb answer.txt
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Phase 1 defused. How about the next one?
That's number 2.  Keep going!
Halfway there!
So you got that one.  Try this one.
Good work!  On to the next...
3 2 4 1 6 5
Congratulations! You've defused the bomb!
Your instructor has been notified and will verify your solution.
```

第六关中包含的循环由于有前面关卡的经验，所以能更快地转换成高级语言去理清楚各数间的逻辑关系，而在其后的链表的具体汇编代码和指针间的赋值等操作，感觉汇编上的操作比大一学习的链表排序的操作的理解更为清晰，指针间的转移也更有逻辑。

4.7 隐藏关

我刚开始以为隐藏关卡是在过了6关后自动出现的单纯的第七关，但当我完成第六关后，程序就此结束了，我就想那隐藏关在哪呢。难道隐藏关真的是隐藏起来的？然后我就再次回看整个程序的汇编代码，搜索关键字phase，发现有一个叫secret_phase的函数，仅仅在phase_defused的函数中被调用。于是我在那句调用secret_phase函数的语句开始往前寻找，发现这是个每个关卡成功后输出提示你成功的句子的函数，然后又有一个scanf函数，在此我就感觉到其可疑，于是还是老套路，用gdb查看其前面的两个地址值是何方神圣，发现还是提示数据类型的。难道是需要我另外填写3个输入？但也没有可以输入的地方了。往下看还看到了熟悉的strings_not_equal函数，是将输入的%s与一个地址值所指向的字符串比较，即要求它们相等才能跳转到调用隐藏关的语句前面。继续用gdb查看那个字符串，是"SecretSYSU"，这肯定就是通向隐藏关的钥匙了，根据前面两个是数字的提示，发现6关中只有第4关是需要输入2个数字的，所以我就在第4关输入的答案后加上"SecretSYSU"，再次运行后，发现在第6关的后面出现了隐藏关的提示。

```
push   %eax
push   $0x804a51f       //"%d %d %s"
push   $0x804c8f0
call   8048810 <__isoc99_sscanf@plt>
```

```
push   $0x804a528
lea    -0x5c(%ebp),%eax
push   %eax
call   8049081 <strings_not_equal>
```

```
call   8048f7d <secret_phase>
```

```
(gdb) x/s 0x804a51f
0x804a51f:      "%d %d %s"
(gdb) x/s 0x804c8f0
0x804c8f0 <input_strings+240>:  ""
```

```
(gdb) x/s 0x804a528
0x804a528:      "SecretSYSU"
```

```
eagle@ubuntu:~/Desktop/bomb45$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
When a problem comes along, you must zip it!
Phase 1 defused. How about the next one?
0 0 0 0 0 0
That's number 2.  Keep going!
0 q 917
Halfway there!
66 2 SecretSYSU
So you got that one.  Try this one.
mfcdhg
Good work!  On to the next...
3 2 4 1 6 5
Curses, you've found the secret phase!
But finding it and solving it are quite different...
```

接下来就是隐藏关的攻关了。在隐藏关中，先是调用了read_line的函数，由函数名可知该函数是getline的功能，随后又调用strtol的函数，该函数是将输入的字符串即要输入的答案转化为相应的数并返回，在此把它设为j。下面几句代码说明j<=3e9H，且将一个地址和j作为参数传递进fun7，用gdb查看上地址为24H。

```
call   804932f <read_line>
```

```
call   8048880 <strtol@plt>
mov    %eax,%ebx            //%eax = j, %ebx = j
lea    -0x1(%eax),%eax      //%eax = j - 1
add    $0x10,%esp
cmp    $0x3e8,%eax
ja     8048fda <secret_phase+0x5d>  //%eax(j - 1) <= 3e8
sub    $0x8,%esp
push   %ebx             //%ebx = j
push   $0x804c0a0       //0x804c0a0 = 24H
call   8048f29 <fun7>
```

我们先不看fun7函数的具体内容，再紧接看下去，fun7的返回值%eax与立即数4比较，说明返回值必须是4，否则引爆炸弹。

```
call   8048f29 <fun7>
add    $0x10,%esp
cmp    $0x4,%eax
je     8048fc0 <secret_phase+0x43>  //%eax = 4
call   80492b5 <explode_bomb>
```

现在再来阅读fun7的函数，发现这个函数与第4关的fun4非常类似，同样都有递归调

用自己，本来我还打算用第四关的迭代方法去推出j的值，但是由于j的范围过大，这种方法我尝试了假设j的小范围时依然是没有得出答案，并且这是关于地址间的递归，并不清楚地址对应的数是什么。然后我就从函数返回值为4入手，我就在想到底是什么情况下才会返回，并且返回的是什么值？由下面代码可知递归最深处的返回值是0。

```
mov    0x8(%ebp),%edx        //%edx = 24H
mov    0xc(%ebp),%ecx        //%ecx = j
```

```
mov    $0x0,%eax         //%eax = 0
cmp    %ecx,%ebx
jne    8048f61 <fun7+0x38>  //若相等则返回
```

```
8048f4e:   83 ec 08              sub    $0x8,%esp
8048f51:   51                    push   %ecx
8048f52:   ff 72 04              pushl  0x4(%edx)
8048f55:   e8 cf ff ff ff        call   8048f29 <fun7>     //fun7((%edx)+4, j)

8048f5a:   83 c4 10              add    $0x10,%esp
8048f5d:   01 c0                 add    %eax,%eax
8048f5f:   eb e8                 jmp    8048f49 <fun7+0x20>  //%eax=2*%eax

8048f61:   83 ec 08              sub    $0x8,%esp
8048f64:   51                    push   %ecx
8048f65:   ff 72 08              pushl  0x8(%edx)
8048f68:   e8 bc ff ff ff        call   8048f29 <fun7>      //fun7((%edx)+8, j)
8048f6d:   83 c4 10              add    $0x10,%esp
8048f70:   8d 44 00 01           lea    0x1(%eax,%eax,1),%eax    //%eax=2*%eax+1
8048f74:   eb d3                 jmp    8048f49 <fun7+0x20>
```

由于最外层返回值为4，则可得出如下反递归过程：

%eax*2=4->%eax=2，所以有*%ebx>%ecx，%ebx+=4

%eax*2=2->%eax=1，所以有*%ebx>%ecx，%ebx+=4

%eax*2+1=2->%eax=0，所以有*%ebx<%ecx，%ebx+=8

*%ebx=%ecx即返回0

所以我们需要刚开始传进fun7的第一个参数的地址即上面的0x804c0a0，接下来就是连续使用gdb查看操作后的地址对应的地址值，最后一次地址得到的数字7就是我们需要输入的j，满足j<=3e9H的范围。

```
(gdb) x *(0x804c0a0+4)
0x804c0ac <n21>:        0x00000008
(gdb) x *(0x804c0ac+4)
0x804c0dc <n31>:        0x00000006
(gdb) x *(0x804c0dc+8)
0x804c124 <n42>:        0x00000007
```

此处在拆弹完成后经与同学讨论后，发现隐藏关的汇编代码写的是数据结构中的二叉树，其结构为：

```
struct node{
    int num;
    node* left;
    node* right;
};
```

则此处的fun7中地址的+4可解释为使树的指针指向其左子树，+8为使树的指针指向右子树，其实也与链表的结构有所类似，只是在递归遍历的过程中地址和指针的交替使用比非递归的实现要难明白，但从高级语言角度来看，递归往往比非递归要更简单明了。

fun7的大致代码为：

```
int fun7(node* t, int j) {
    node* edx = t;
    int ecx = j;
    node* ebx = ebx;
    if (ecx > ebx) {
        eax = fun7(edx->left, j);//edx + 4 = edx->left
        eax += eax;
        return eax;
    }
    else {
        eax = 0;
        if (ecx == ebx)
            return eax;
        eax = fun7(edx->right, j);//edx + 8 = edx->right
        return eax;
    }
}
```

经验证为正确答案。（完结撒花）

```
eagle@ubuntu:~/Desktop/bomb45$ ./bomb answer.txt
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Phase 1 defused. How about the next one?
That's number 2.  Keep going!
Halfway there!
So you got that one.  Try this one.
Good work!  On to the next...
Curses, you've found the secret phase!
But finding it and solving it are quite different...
7
Wow! You've defused the secret stage!
Congratulations! You've defused the bomb!
Your instructor has been notified and will verify your solution.
```

五. 实验心得

这个二进制炸弹实验的确像老师所说的，痛并快乐着。刚开始对于一大片汇编代码只能一个一个指令去找它是什么意思，到后来过了3、4关后基本能够读懂一关汇编代码，并能够由此分析对应某些高级语言现象，这些每一关的成功解除炸弹确实令人兴奋。在这个实验过程中，我碰到了函数参数如何传递、传递后在栈中的存放位置和汇编代码中函数看不懂等问题，经过在网上博客的学习，我知道了每一个函数调用都需要保护现场和返回时的恢复现场，被调用函数的栈帧是在调用函数的栈帧的基础上开辟的，而被调用函数的ESP根据参数的个数加上相应的数使ESP指向原来函数的栈帧的栈顶（一个参数加4，两个加8，以此类推）。当然这些有关函数的汇编语言在这里还是比较简单的，在有关一些指针、递归的问题上，我依然不能很快顺利地转换为高级语言去先思考。同时我也在想函数刚开始所开的栈的空间是否需要很严格的限制，还有对于赋值14H给某一确定地址的用意是什么，寄存器间的关系等等。这些思考的问题虽然与能否成功解除炸弹没有很大的关系，但是我想也应该是一些基本的底层的操作。幸好的是这些问题一些博客还是有比较清楚的解释的，我也更清楚了高级语言在底层是怎样操作的，也明白了一些寄存器和半寄存器的关系，还有一些按位操作符的用途在底层中具体用途是那么广的，我觉得通过这个实验能够真正将计组理论课上学到的东西转化为实际所用也是掌握理论知识的好途径。

【程序代码】

```
./bomb:     file format elf32-i386


Disassembly of section .init:

080486f4 <_init>:
 80486f4:   53                      push   %ebx
 80486f5:   83 ec 08                sub    $0x8,%esp
 80486f8:   e8 33 02 00 00          call   8048930 <__x86.get_pc_thunk.bx>
```

```
 80486fd:   81 c3 03 39 00 00       add    $0x3903,%ebx
 8048703:   8b 83 fc ff ff ff       mov    -0x4(%ebx),%eax
 8048709:   85 c0                   test   %eax,%eax
 804870b:   74 05                   je     8048712 <_init+0x1e>
 804870d:   e8 be 01 00 00          call   80488d0 <__gmon_start__@plt>
 8048712:   83 c4 08                add    $0x8,%esp
 8048715:   5b                      pop    %ebx
 8048716:   c3                      ret

Disassembly of section .plt:

08048720 <.plt>:
 8048720:   ff 35 04 c0 04 08       pushl  0x804c004
 8048726:   ff 25 08 c0 04 08       jmp    *0x804c008
 804872c:   00 00                   add    %al,(%eax)
    ...

08048730 <read@plt>:
 8048730:   ff 25 0c c0 04 08       jmp    *0x804c00c
 8048736:   68 00 00 00 00          push   $0x0
 804873b:   e9 e0 ff ff ff          jmp    8048720 <.plt>

08048740 <fflush@plt>:
 8048740:   ff 25 10 c0 04 08       jmp    *0x804c010
 8048746:   68 08 00 00 00          push   $0x8
 804874b:   e9 d0 ff ff ff          jmp    8048720 <.plt>

08048750 <fgets@plt>:
 8048750:   ff 25 14 c0 04 08       jmp    *0x804c014
```

```
 8048756:   68 10 00 00 00          push   $0x10
 804875b:   e9 c0 ff ff ff          jmp    8048720 <.plt>

08048760 <signal@plt>:
 8048760:   ff 25 18 c0 04 08       jmp    *0x804c018
 8048766:   68 18 00 00 00          push   $0x18
 804876b:   e9 b0 ff ff ff          jmp    8048720 <.plt>

08048770 <sleep@plt>:
 8048770:   ff 25 1c c0 04 08       jmp    *0x804c01c
 8048776:   68 20 00 00 00          push   $0x20
 804877b:   e9 a0 ff ff ff          jmp    8048720 <.plt>

08048780 <alarm@plt>:
 8048780:   ff 25 20 c0 04 08       jmp    *0x804c020
 8048786:   68 28 00 00 00          push   $0x28
 804878b:   e9 90 ff ff ff          jmp    8048720 <.plt>

08048790 <__stack_chk_fail@plt>:
 8048790:   ff 25 24 c0 04 08       jmp    *0x804c024
 8048796:   68 30 00 00 00          push   $0x30
 804879b:   e9 80 ff ff ff          jmp    8048720 <.plt>

080487a0 <strcpy@plt>:
 80487a0:   ff 25 28 c0 04 08       jmp    *0x804c028
 80487a6:   68 38 00 00 00          push   $0x38
 80487ab:   e9 70 ff ff ff          jmp    8048720 <.plt>

080487b0 <getenv@plt>:
```

```
 80487b0:   ff 25 2c c0 04 08          jmp     *0x804c02c
 80487b6:   68 40 00 00 00             push    $0x40
 80487bb:   e9 60 ff ff ff             jmp     8048720 <.plt>

080487c0 <puts@plt>:
 80487c0:   ff 25 30 c0 04 08          jmp     *0x804c030
 80487c6:   68 48 00 00 00             push    $0x48
 80487cb:   e9 50 ff ff ff             jmp     8048720 <.plt>

080487d0 <__memmove_chk@plt>:
 80487d0:   ff 25 34 c0 04 08          jmp     *0x804c034
 80487d6:   68 50 00 00 00             push    $0x50
 80487db:   e9 40 ff ff ff             jmp     8048720 <.plt>

080487e0 <exit@plt>:
 80487e0:   ff 25 38 c0 04 08          jmp     *0x804c038
 80487e6:   68 58 00 00 00             push    $0x58
 80487eb:   e9 30 ff ff ff             jmp     8048720 <.plt>

080487f0 <__libc_start_main@plt>:
 80487f0:   ff 25 3c c0 04 08          jmp     *0x804c03c
 80487f6:   68 60 00 00 00             push    $0x60
 80487fb:   e9 20 ff ff ff             jmp     8048720 <.plt>

08048800 <write@plt>:
 8048800:   ff 25 40 c0 04 08          jmp     *0x804c040
 8048806:   68 68 00 00 00             push    $0x68
 804880b:   e9 10 ff ff ff             jmp     8048720 <.plt>
```

```
08048810 <__isoc99_sscanf@plt>:
 8048810:  ff 25 44 c0 04 08          jmp     *0x804c044
 8048816:  68 70 00 00 00             push    $0x70
 804881b:  e9 00 ff ff ff             jmp     8048720 <.plt>

08048820 <fopen@plt>:
 8048820:  ff 25 48 c0 04 08          jmp     *0x804c048
 8048826:  68 78 00 00 00             push    $0x78
 804882b:  e9 f0 fe ff ff             jmp     8048720 <.plt>

08048830 <__errno_location@plt>:
 8048830:  ff 25 4c c0 04 08          jmp     *0x804c04c
 8048836:  68 80 00 00 00             push    $0x80
 804883b:  e9 e0 fe ff ff             jmp     8048720 <.plt>

08048840 <__printf_chk@plt>:
 8048840:  ff 25 50 c0 04 08          jmp     *0x804c050
 8048846:  68 88 00 00 00             push    $0x88
 804884b:  e9 d0 fe ff ff             jmp     8048720 <.plt>

08048850 <socket@plt>:
 8048850:  ff 25 54 c0 04 08          jmp     *0x804c054
 8048856:  68 90 00 00 00             push    $0x90
 804885b:  e9 c0 fe ff ff             jmp     8048720 <.plt>

08048860 <__fprintf_chk@plt>:
 8048860:  ff 25 58 c0 04 08          jmp     *0x804c058
 8048866:  68 98 00 00 00             push    $0x98
 804886b:  e9 b0 fe ff ff             jmp     8048720 <.plt>
```

```
08048870 <gethostbyname@plt>:
 8048870:   ff 25 5c c0 04 08          jmp     *0x804c05c
 8048876:   68 a0 00 00 00             push    $0xa0
 804887b:   e9 a0 fe ff ff             jmp     8048720 <.plt>

08048880 <strtol@plt>:
 8048880:   ff 25 60 c0 04 08          jmp     *0x804c060
 8048886:   68 a8 00 00 00             push    $0xa8
 804888b:   e9 90 fe ff ff             jmp     8048720 <.plt>

08048890 <connect@plt>:
 8048890:   ff 25 64 c0 04 08          jmp     *0x804c064
 8048896:   68 b0 00 00 00             push    $0xb0
 804889b:   e9 80 fe ff ff             jmp     8048720 <.plt>

080488a0 <close@plt>:
 80488a0:   ff 25 68 c0 04 08          jmp     *0x804c068
 80488a6:   68 b8 00 00 00             push    $0xb8
 80488ab:   e9 70 fe ff ff             jmp     8048720 <.plt>

080488b0 <__ctype_b_loc@plt>:
 80488b0:   ff 25 6c c0 04 08          jmp     *0x804c06c
 80488b6:   68 c0 00 00 00             push    $0xc0
 80488bb:   e9 60 fe ff ff             jmp     8048720 <.plt>

080488c0 <__sprintf_chk@plt>:
 80488c0:   ff 25 70 c0 04 08          jmp     *0x804c070
 80488c6:   68 c8 00 00 00             push    $0xc8
```

```
 80488cb:   e9 50 fe ff ff          jmp    8048720 <.plt>

Disassembly of section .plt.got:

080488d0 <__gmon_start__@plt>:
 80488d0:   ff 25 fc bf 04 08       jmp    *0x804bffc
 80488d6:   66 90                   xchg   %ax,%ax

Disassembly of section .text:

080488e0 <_start>:
 80488e0:   31 ed                   xor    %ebp,%ebp
 80488e2:   5e                      pop    %esi
 80488e3:   89 e1                   mov    %esp,%ecx
 80488e5:   83 e4 f0                and    $0xfffffff0,%esp
 80488e8:   50                      push   %eax
 80488e9:   54                      push   %esp
 80488ea:   52                      push   %edx
 80488eb:   e8 23 00 00 00          call   8048913 <_start+0x33>
 80488f0:   81 c3 10 37 00 00       add    $0x3710,%ebx
 80488f6:   8d 83 00 e1 ff ff       lea    -0x1f00(%ebx),%eax
 80488fc:   50                      push   %eax
 80488fd:   8d 83 a0 e0 ff ff       lea    -0x1f60(%ebx),%eax
 8048903:   50                      push   %eax
 8048904:   51                      push   %ecx
 8048905:   56                      push   %esi
 8048906:   c7 c0 f6 89 04 08       mov    $0x80489f6,%eax
 804890c:   50                      push   %eax
 804890d:   e8 de fe ff ff          call   80487f0 <__libc_start_main@plt>
```

```
 8048912:  f4                    hlt
 8048913:  8b 1c 24              mov    (%esp),%ebx
 8048916:  c3                    ret
 8048917:  66 90                 xchg   %ax,%ax
 8048919:  66 90                 xchg   %ax,%ax
 804891b:  66 90                 xchg   %ax,%ax
 804891d:  66 90                 xchg   %ax,%ax
 804891f:  90                    nop

08048920 <_dl_relocate_static_pie>:
 8048920:  f3 c3                 repz ret
 8048922:  66 90                 xchg   %ax,%ax
 8048924:  66 90                 xchg   %ax,%ax
 8048926:  66 90                 xchg   %ax,%ax
 8048928:  66 90                 xchg   %ax,%ax
 804892a:  66 90                 xchg   %ax,%ax
 804892c:  66 90                 xchg   %ax,%ax
 804892e:  66 90                 xchg   %ax,%ax

08048930 <__x86.get_pc_thunk.bx>:
 8048930:  8b 1c 24              mov    (%esp),%ebx
 8048933:  c3                    ret
 8048934:  66 90                 xchg   %ax,%ax
 8048936:  66 90                 xchg   %ax,%ax
 8048938:  66 90                 xchg   %ax,%ax
 804893a:  66 90                 xchg   %ax,%ax
 804893c:  66 90                 xchg   %ax,%ax
 804893e:  66 90                 xchg   %ax,%ax
```

```
08048940 <deregister_tm_clones>:
 8048940:   b8 c0 c7 04 08          mov    $0x804c7c0,%eax
 8048945:   3d c0 c7 04 08          cmp    $0x804c7c0,%eax
 804894a:   74 24                   je                            8048970
<deregister_tm_clones+0x30>
 804894c:   b8 00 00 00 00          mov    $0x0,%eax
 8048951:   85 c0                   test   %eax,%eax
 8048953:   74 1b                   je                            8048970
<deregister_tm_clones+0x30>
 8048955:   55                      push   %ebp
 8048956:   89 e5                   mov    %esp,%ebp
 8048958:   83 ec 14                sub    $0x14,%esp
 804895b:   68 c0 c7 04 08          push   $0x804c7c0
 8048960:   ff d0                   call   *%eax
 8048962:   83 c4 10                add    $0x10,%esp
 8048965:   c9                      leave
 8048966:   c3                      ret
 8048967:   89 f6                   mov    %esi,%esi
 8048969:   8d bc 27 00 00 00 00    lea    0x0(%edi,%eiz,1),%edi
 8048970:   f3 c3                   repz ret
 8048972:   8d b4 26 00 00 00 00    lea    0x0(%esi,%eiz,1),%esi
 8048979:   8d bc 27 00 00 00 00    lea    0x0(%edi,%eiz,1),%edi

08048980 <register_tm_clones>:
 8048980:   b8 c0 c7 04 08          mov    $0x804c7c0,%eax
 8048985:   2d c0 c7 04 08          sub    $0x804c7c0,%eax
 804898a:   c1 f8 02                sar    $0x2,%eax
 804898d:   89 c2                   mov    %eax,%edx
 804898f:   c1 ea 1f                shr    $0x1f,%edx
```

```
 8048992:  01 d0                    add    %edx,%eax
 8048994:  d1 f8                    sar    %eax
 8048996:  74 20                    je     80489b8 <register_tm_clones+0x38>
 8048998:  ba 00 00 00 00           mov    $0x0,%edx
 804899d:  85 d2                    test   %edx,%edx
 804899f:  74 17                    je     80489b8 <register_tm_clones+0x38>
 80489a1:  55                       push   %ebp
 80489a2:  89 e5                    mov    %esp,%ebp
 80489a4:  83 ec 10                 sub    $0x10,%esp
 80489a7:  50                       push   %eax
 80489a8:  68 c0 c7 04 08           push   $0x804c7c0
 80489ad:  ff d2                    call   *%edx
 80489af:  83 c4 10                 add    $0x10,%esp
 80489b2:  c9                       leave
 80489b3:  c3                       ret
 80489b4:  8d 74 26 00              lea    0x0(%esi,%eiz,1),%esi
 80489b8:  f3 c3                    repz ret
 80489ba:  8d b6 00 00 00 00        lea    0x0(%esi),%esi

080489c0 <__do_global_dtors_aux>:
 80489c0:  80 3d e8 c7 04 08 00     cmpb   $0x0,0x804c7e8
 80489c7:  75 17                    jne                                  80489e0
<__do_global_dtors_aux+0x20>
 80489c9:  55                       push   %ebp
 80489ca:  89 e5                    mov    %esp,%ebp
 80489cc:  83 ec 08                 sub    $0x8,%esp
 80489cf:  e8 6c ff ff ff           call   8048940 <deregister_tm_clones>
 80489d4:  c6 05 e8 c7 04 08 01     movb   $0x1,0x804c7e8
 80489db:  c9                       leave
```

```
 80489dc:   c3                      ret
 80489dd:   8d 76 00                lea    0x0(%esi),%esi
 80489e0:   f3 c3                   repz ret
 80489e2:   8d b4 26 00 00 00 00    lea    0x0(%esi,%eiz,1),%esi
 80489e9:   8d bc 27 00 00 00 00    lea    0x0(%edi,%eiz,1),%edi

080489f0 <frame_dummy>:
 80489f0:   55                      push   %ebp
 80489f1:   89 e5                   mov    %esp,%ebp
 80489f3:   5d                      pop    %ebp
 80489f4:   eb 8a                   jmp    8048980 <register_tm_clones>

080489f6 <main>:
 80489f6:   8d 4c 24 04             lea    0x4(%esp),%ecx
 80489fa:   83 e4 f0                and    $0xfffffff0,%esp
 80489fd:   ff 71 fc             pushl  -0x4(%ecx)
 8048a00:   55                      push   %ebp
 8048a01:   89 e5                   mov    %esp,%ebp
 8048a03:   53                      push   %ebx
 8048a04:   51                      push   %ecx
 8048a05:   8b 01                   mov    (%ecx),%eax
 8048a07:   8b 59 04                mov    0x4(%ecx),%ebx
 8048a0a:   83 f8 01                cmp    $0x1,%eax
 8048a0d:   0f 84 fe 00 00 00       je     8048b11 <main+0x11b>
 8048a13:   83 f8 02                cmp    $0x2,%eax
 8048a16:   0f 85 21 01 00 00       jne    8048b3d <main+0x147>
 8048a1c:   83 ec 08                sub    $0x8,%esp
 8048a1f:   68 20 a1 04 08          push   $0x804a120
 8048a24:   ff 73 04                pushl  0x4(%ebx)
```

```
8048a27:  e8 f4 fd ff ff          call   8048820 <fopen@plt>
8048a2c:  a3 f0 c7 04 08          mov    %eax,0x804c7f0
8048a31:  83 c4 10                add    $0x10,%esp
8048a34:  85 c0                   test   %eax,%eax
8048a36:  0f 84 e4 00 00 00       je     8048b20 <main+0x12a>
8048a3c:  e8 aa 06 00 00          call   80490eb <initialize_bomb>
8048a41:  83 ec 0c                sub    $0xc,%esp
8048a44:  68 a4 a1 04 08          push   $0x804a1a4
8048a49:  e8 72 fd ff ff          call   80487c0 <puts@plt>
8048a4e:  c7 04 24 e0 a1 04 08    movl   $0x804a1e0,(%esp)
8048a55:  e8 66 fd ff ff          call   80487c0 <puts@plt>
8048a5a:  e8 d0 08 00 00          call   804932f <read_line>
8048a5f:  89 04 24                mov    %eax,(%esp)
8048a62:  e8 f3 00 00 00          call   8048b5a <phase_1>
8048a67:  e8 d4 09 00 00          call   8049440 <phase_defused>
8048a6c:  c7 04 24 0c a2 04 08    movl   $0x804a20c,(%esp)
8048a73:  e8 48 fd ff ff          call   80487c0 <puts@plt>
8048a78:  e8 b2 08 00 00          call   804932f <read_line>
8048a7d:  89 04 24                mov    %eax,(%esp)
8048a80:  e8 f8 00 00 00          call   8048b7d <phase_2>
8048a85:  e8 b6 09 00 00          call   8049440 <phase_defused>
8048a8a:  c7 04 24 59 a1 04 08    movl   $0x804a159,(%esp)
8048a91:  e8 2a fd ff ff          call   80487c0 <puts@plt>
8048a96:  e8 94 08 00 00          call   804932f <read_line>
8048a9b:  89 04 24                mov    %eax,(%esp)
8048a9e:  e8 45 01 00 00          call   8048be8 <phase_3>
8048aa3:  e8 98 09 00 00          call   8049440 <phase_defused>
8048aa8:  c7 04 24 77 a1 04 08    movl   $0x804a177,(%esp)
8048aaf:  e8 0c fd ff ff          call   80487c0 <puts@plt>
```

```
8048ab4:  e8 76 08 00 00          call   804932f <read_line>
8048ab9:  89 04 24                mov    %eax,(%esp)
8048abc:  e8 89 02 00 00          call   8048d4a <phase_4>
8048ac1:  e8 7a 09 00 00          call   8049440 <phase_defused>
8048ac6:  c7 04 24 38 a2 04 08    movl   $0x804a238,(%esp)
8048acd:  e8 ee fc ff ff          call   80487c0 <puts@plt>
8048ad2:  e8 58 08 00 00          call   804932f <read_line>
8048ad7:  89 04 24                mov    %eax,(%esp)
8048ada:  e8 d6 02 00 00          call   8048db5 <phase_5>
8048adf:  e8 5c 09 00 00          call   8049440 <phase_defused>
8048ae4:  c7 04 24 86 a1 04 08    movl   $0x804a186,(%esp)
8048aeb:  e8 d0 fc ff ff          call   80487c0 <puts@plt>
8048af0:  e8 3a 08 00 00          call   804932f <read_line>
8048af5:  89 04 24                mov    %eax,(%esp)
8048af8:  e8 3a 03 00 00          call   8048e37 <phase_6>
8048afd:  e8 3e 09 00 00          call   8049440 <phase_defused>
8048b02:  b8 00 00 00 00          mov    $0x0,%eax
8048b07:  8d 65 f8                lea    -0x8(%ebp),%esp
8048b0a:  59                      pop    %ecx
8048b0b:  5b                      pop    %ebx
8048b0c:  5d                      pop    %ebp
8048b0d:  8d 61 fc                lea    -0x4(%ecx),%esp
8048b10:  c3                      ret
8048b11:  a1 e0 c7 04 08          mov    0x804c7e0,%eax
8048b16:  a3 f0 c7 04 08          mov    %eax,0x804c7f0
8048b1b:  e9 1c ff ff ff          jmp    8048a3c <main+0x46>
8048b20:  ff 73 04                pushl  0x4(%ebx)
8048b23:  ff 33                   pushl  (%ebx)
8048b25:  68 22 a1 04 08          push   $0x804a122
```

```
 8048b2a:  6a 01                    push   $0x1
 8048b2c:  e8 0f fd ff ff           call   8048840 <__printf_chk@plt>
 8048b31:  c7 04 24 08 00 00 00     movl   $0x8,(%esp)
 8048b38:  e8 a3 fc ff ff           call   80487e0 <exit@plt>
 8048b3d:  83 ec 04                 sub    $0x4,%esp
 8048b40:  ff 33                    pushl  (%ebx)
 8048b42:  68 3f a1 04 08           push   $0x804a13f
 8048b47:  6a 01                    push   $0x1
 8048b49:  e8 f2 fc ff ff           call   8048840 <__printf_chk@plt>
 8048b4e:  c7 04 24 08 00 00 00     movl   $0x8,(%esp)
 8048b55:  e8 86 fc ff ff           call   80487e0 <exit@plt>

08048b5a <phase_1>:
 8048b5a:  55                       push   %ebp
 8048b5b:  89 e5                    mov    %esp,%ebp
 8048b5d:  83 ec 10                 sub    $0x10,%esp
 8048b60:  68 5c a2 04 08           push   $0x804a25c
 8048b65:  ff 75 08                 pushl  0x8(%ebp)
 8048b68:  e8 14 05 00 00           call   8049081 <strings_not_equal>
 8048b6d:  83 c4 10                 add    $0x10,%esp
 8048b70:  85 c0                    test   %eax,%eax   //test为将两数按位相与
 8048b72:  75 02                    jne          8048b76   <phase_1+0x1c>// 即
仅%eax = 0时不跳转
 8048b74:  c9                       leave
 8048b75:  c3                       ret
 8048b76:  e8 3a 07 00 00           call   80492b5 <explode_bomb>
 8048b7b:  eb f7                    jmp    8048b74 <phase_1+0x1a>

08048b7d <phase_2>:
```

```
8048b7d:  55                    push   %ebp
8048b7e:  89 e5                 mov    %esp,%ebp
8048b80:  53                    push   %ebx
8048b81:  83 ec 3c              sub    $0x3c,%esp
8048b84:  65 a1 14 00 00 00     mov    %gs:0x14,%eax
8048b8a:  89 45 f4              mov    %eax,-0xc(%ebp)
8048b8d:  31 c0                 xor    %eax,%eax
8048b8f:  8d 45 dc              lea    -0x24(%ebp),%eax
8048b92:  50                    push   %eax
8048b93:  ff 75 08              pushl  0x8(%ebp)
8048b96:  e8 5a 07 00 00        call   80492f5 <read_six_numbers>

8048b9b:  83 c4 10              add    $0x10,%esp
8048b9e:  83 7d dc 05           cmpl   $0x5,-0x24(%ebp)
8048ba2:  77 07                 ja     8048bab <phase_2+0x2e>
   //-0x24(%ebp) <= 5

8048ba4:  bb 01 00 00 00        mov    $0x1,%ebx       //%ebx = 1
8048ba9:  eb 0f                 jmp    8048bba <phase_2+0x3d>
8048bab:  e8 05 07 00 00        call   80492b5 <explode_bomb>
8048bb0:  eb f2                 jmp    8048ba4 <phase_2+0x27>

//
8048bb2:  83 c3 01              add    $0x1,%ebx  //%ebx+=1
8048bb5:  83 fb 06              cmp    $0x6,%ebx  //循环5次
8048bb8:  74 18                 je     8048bd2 <phase_2+0x55>

8048bba:  8b 44 9d d8           mov    -0x28(%ebp,%ebx,4),%eax
8048bbe:  89 45 d4              mov    %eax,-0x2c(%ebp)
```

```
 8048bc1:   89 d9                    mov     %ebx,%ecx       //%ecx = %ebx
 8048bc3:   d3 e0                    shl     %cl,%eax        //%eax 左 移 %cl
位,%cl是%ecx的低8位
 8048bc5:   39 44 9d dc              cmp     %eax,-0x24(%ebp,%ebx,4)
 8048bc9:   74 e7                    je      8048bb2 <phase_2+0x35>
    //%eax == -0x24(%ebp,%ebx,4)
 //以上为一个循环，第一个数的范围是0到5，而后5个数分别是由前一个数左移1-5位得
到

 8048bcb:   e8 e5 06 00 00           call    80492b5 <explode_bomb>
 8048bd0:   eb e0                    jmp     8048bb2 <phase_2+0x35>

 8048bd2:   8b 45 f4                 mov     -0xc(%ebp),%eax
 8048bd5:   65 33 05 14 00 00 00     xor     %gs:0x14,%eax
 8048bdc:   75 05                    jne     8048be3 <phase_2+0x66>
 8048bde:   8b 5d fc                 mov     -0x4(%ebp),%ebx
 8048be1:   c9                       leave

 8048be2:   c3                       ret
 8048be3:   e8 a8 fb ff ff           call    8048790 <__stack_chk_fail@plt>

08048be8 <phase_3>:
 8048be8:   55                       push    %ebp
 8048be9:   89 e5                    mov     %esp,%ebp
 8048beb:   83 ec 24                 sub     $0x24,%esp //开24H空间
 8048bee:   65 a1 14 00 00 00        mov     %gs:0x14,%eax //eax = 14H
 8048bf4:   89 45 f4                 mov     %eax,-0xc(%ebp)
 8048bf7:   31 c0                    xor     %eax,%eax   //eax = 0
 8048bf9:   8d 45 f0                 lea     -0x10(%ebp),%eax        //三%d
```

```
 8048bfc:   50                      push   %eax
 8048bfd:   8d 45 eb                lea    -0x15(%ebp),%eax        //二%c
 8048c00:   50                      push   %eax
 8048c01:   8d 45 ec                lea    -0x14(%ebp),%eax            //第一
个参数%d
 8048c04:   50                      push   %eax
 8048c05:   68 b2 a2 04 08          push   $0x804a2b2
 8048c0a:   ff 75 08                pushl  0x8(%ebp)
 8048c0d:   e8 fe fb ff ff          call   8048810 <__isoc99_sscanf@plt>
 8048c12:   83 c4 20                add    $0x20,%esp
 8048c15:   83 f8 02                cmp    $0x2,%eax       //eax为scanf的返回
值
 8048c18:   7e 14                   jle    8048c2e <phase_3+0x46>
    //eax>2, eax<=2爆炸

 8048c1a:   83 7d ec 07             cmpl   $0x7,-0x14(%ebp)
 8048c1e:   0f 87 b4 00 00 00       ja     8048cd8 <phase_3+0xf0>//
    (i)-0x14(%ebp)<=7

 8048c24:   8b 45 ec                mov    -0x14(%ebp),%eax
 8048c27:   ff 24 85 c4 a2 04 08    jmp    *0x804a2c4(,%eax,4)         //j
8048c35 + 4i
 8048c2e:   e8 82 06 00 00          call   80492b5 <explode_bomb>
 8048c33:   eb e5                   jmp    8048c1a <phase_3+0x32>

 8048c35:   b8 71 00 00 00          mov    $0x71,%eax          //i = 0跳转到此
 8048c3a:   81 7d f0 95 03 00 00    cmpl   $0x395,-0x10(%ebp)
 8048c41:   0f 84 9b 00 00 00       je             8048ce2 <phase_3+0xfa>
    //-0x10(%ebp) = 395H
```

```
 8048c47:   e8 69 06 00 00          call    80492b5 <explode_bomb>
 8048c4c:   b8 71 00 00 00          mov     $0x71,%eax

 8048c51:   e9 8c 00 00 00          jmp     8048ce2 <phase_3+0xfa>          //i
= 7
 8048c56:   b8 6e 00 00 00          mov     $0x6e,%eax
 8048c5b:   81 7d f0 c5 02 00 00    cmpl    $0x2c5,-0x10(%ebp)
 8048c62:   74 7e                   je      8048ce2 <phase_3+0xfa>
 8048c64:   e8 4c 06 00 00          call    80492b5 <explode_bomb>

 8048c69:   b8 6e 00 00 00          mov     $0x6e,%eax      //i = 13
 8048c6e:   eb 72                   jmp     8048ce2 <phase_3+0xfa>

 8048c70:   b8 79 00 00 00          mov     $0x79,%eax

 8048c75:   81 7d f0 d3 01 00 00    cmpl    $0x1d3,-0x10(%ebp)      //i = 16
 8048c7c:   74 64                   je      8048ce2 <phase_3+0xfa>
 8048c7e:   e8 32 06 00 00          call    80492b5 <explode_bomb>

 8048c83:   b8 79 00 00 00          mov     $0x79,%eax
 8048c88:   eb 58                   jmp     8048ce2 <phase_3+0xfa>
 8048c8a:   b8 68 00 00 00          mov     $0x68,%eax
 8048c8f:   81 7d f0 3c 03 00 00    cmpl    $0x33c,-0x10(%ebp)
 8048c96:   74 4a                   je      8048ce2 <phase_3+0xfa>
 8048c98:   e8 18 06 00 00          call    80492b5 <explode_bomb>

 8048c9d:   b8 68 00 00 00          mov     $0x68,%eax
 8048ca2:   eb 3e                   jmp     8048ce2 <phase_3+0xfa>
```

```
 8048ca4:   b8 64 00 00 00          mov     $0x64,%eax
 8048ca9:   81 7d f0 f1 02 00 00    cmpl    $0x2f1,-0x10(%ebp)
 8048cb0:   74 30                   je      8048ce2 <phase_3+0xfa>
 8048cb2:   e8 fe 05 00 00          call    80492b5 <explode_bomb>
 8048cb7:   b8 64 00 00 00          mov     $0x64,%eax
 8048cbc:   eb 24                   jmp     8048ce2 <phase_3+0xfa>
 8048cbe:   b8 6a 00 00 00          mov     $0x6a,%eax
 8048cc3:   81 7d f0 9a 02 00 00    cmpl    $0x29a,-0x10(%ebp)
 8048cca:   74 16                   je      8048ce2 <phase_3+0xfa>
 8048ccc:   e8 e4 05 00 00          call    80492b5 <explode_bomb>
 8048cd1:   b8 6a 00 00 00          mov     $0x6a,%eax
 8048cd6:   eb 0a                   jmp     8048ce2 <phase_3+0xfa>
 8048cd8:   e8 d8 05 00 00          call    80492b5 <explode_bomb>
 8048cdd:   b8 66 00 00 00          mov     $0x66,%eax

 8048ce2:   38 45 eb                cmp     %al,-0x15(%ebp)
 8048ce5:   74 05                   je      8048cec <phase_3+0x104>
    //%al == -0x15(%ebp)
 8048ce7:   e8 c9 05 00 00          call    80492b5 <explode_bomb>
 8048cec:   8b 45 f4                mov     -0xc(%ebp),%eax
 8048cef:   65 33 05 14 00 00 00    xor     %gs:0x14,%eax
 8048cf6:   75 02                   jne     8048cfa <phase_3+0x112>
 8048cf8:   c9                      leave
 8048cf9:   c3                      ret
 8048cfa:   e8 91 fa ff ff          call    8048790 <__stack_chk_fail@plt>

08048cff <func4>:
 8048cff:   55                      push    %ebp
 8048d00:   89 e5                   mov     %esp,%ebp
```

```
 8048d02:  57                    push   %edi            //保留现场
 8048d03:  56                    push   %esi
 8048d04:  53                    push   %ebx

 8048d05:  83 ec 0c              sub    $0xc,%esp
 8048d08:  8b 75 08              mov    0x8(%ebp),%esi     //参数1 %esi =
7
 8048d0b:  8b 7d 0c              mov    0xc(%ebp),%edi     //参数2%edi  =
j
 8048d0e:  b8 00 00 00 00        mov    $0x0,%eax          //%eax = 0
 8048d13:  85 f6                 test   %esi,%esi
 8048d15:  7e 07                 jle    8048d1e <func4+0x1f>
    //%esi<=0则跳转离开fun4
 8048d17:  89 f8                 mov    %edi,%eax          //%eax = %edi
= j
 8048d19:  83 fe 01              cmp    $0x1,%esi
 8048d1c:  75 08                 jne    8048d26 <func4+0x27>      //%esi
= 1离开fun4

 8048d1e:  8d 65 f4              lea    -0xc(%ebp),%esp

 8048d21:  5b                    pop    %ebx           //恢复现场
 8048d22:  5e                    pop    %esi
 8048d23:  5f                    pop    %edi
 8048d24:  5d                    pop    %ebp
 8048d25:  c3                    ret

 8048d26:  83 ec 08              sub    $0x8,%esp      //开辟2个数的空间
 8048d29:  57                    push   %edi               //参数2 %edi =
```

```
j
 8048d2a:  8d 46 ff                lea     -0x1(%esi),%eax
 8048d2d:  50                      push    %eax                  //参数1  %eax
-= 1

 8048d2e:  e8 cc ff ff ff          call    8048cff <func4>
 8048d33:  83 c4 08                add     $0x8,%esp
 8048d36:  8d 1c 38                lea     (%eax,%edi,1),%ebx
 8048d39:  57                      push    %edi                  //参数1 %edi =
j
 8048d3a:  83 ee 02                sub     $0x2,%esi
 8048d3d:  56                      push    %esi                  //参数2  %esi
-= 2
 8048d3e:  e8 bc ff ff ff          call    8048cff <func4>
 8048d43:  83 c4 10                add     $0x10,%esp
 8048d46:  01 d8                   add     %ebx,%eax
 8048d48:  eb d4                   jmp     8048d1e <func4+0x1f>

08048d4a <phase_4>:
 8048d4a:  55                      push    %ebp
 8048d4b:  89 e5                   mov     %esp,%ebp
 8048d4d:  83 ec 18                sub     $0x18,%esp
 8048d50:  65 a1 14 00 00 00       mov     %gs:0x14,%eax
 8048d56:  89 45 f4                mov     %eax,-0xc(%ebp)
 8048d59:  31 c0                   xor     %eax,%eax
 8048d5b:  8d 45 ec                lea     -0x14(%ebp),%eax      //第二个参
数j
 8048d5e:  50                      push    %eax
 8048d5f:  8d 45 f0                lea     -0x10(%ebp),%eax      //第一个参
```

数i

8048d62: 50 push %eax

8048d63: 68 c5 a4 04 08 push $0x804a4c5 //%d %d

8048d68: ff 75 08 pushl 0x8(%ebp) //stdin

8048d6b: e8 a0 fa ff ff call 8048810 <__isoc99_sscanf@plt>

//scanf

8048d70: 83 c4 10 add $0x10,%esp

8048d73: 83 f8 02 cmp $0x2,%eax //输入两个数

8048d76: 75 0b jne 8048d83 <phase_4+0x39>

8048d78: 8b 45 ec mov -0x14(%ebp),%eax

8048d7b: 83 e8 02 sub $0x2,%eax //%eax = j - 2

8048d7e: 83 f8 02 cmp $0x2,%eax //%eax >= 2 无符号比较

8048d81: 76 05 jbe 8048d88 <phase_4+0x3e> //j <= 4

8048d83: e8 2d 05 00 00 call 80492b5 <explode_bomb>

8048d88: 83 ec 08 sub $0x8,%esp

8048d8b: ff 75 ec pushl -0x14(%ebp) //参2 j

8048d8e: 6a 07 push $0x7 //参1 7

8048d90: e8 6a ff ff ff call 8048cff <func4>

8048d95: 83 c4 10 add $0x10,%esp

8048d98: 39 45 f0 cmp %eax,-0x10(%ebp)

8048d9b: 74 05 je 8048da2 <phase_4+0x58>

//-0x10(%ebp) == %eax

8048d9d: e8 13 05 00 00 call 80492b5 <explode_bomb>

8048da2: 8b 45 f4 mov -0xc(%ebp),%eax

```
 8048da5:   65 33 05 14 00 00 00    xor     %gs:0x14,%eax
 8048dac:   75 02                   jne     8048db0 <phase_4+0x66>
 8048dae:   c9                      leave
 8048daf:   c3                      ret
 8048db0:   e8 db f9 ff ff         call    8048790 <__stack_chk_fail@plt>

08048db5 <phase_5>:
 8048db5:   55                      push    %ebp
 8048db6:   89 e5                   mov     %esp,%ebp
 8048db8:   53                      push    %ebx
 8048db9:   83 ec 20                sub     $0x20,%esp
 8048dbc:   8b 5d 08                mov     0x8(%ebp),%ebx
 8048dbf:   65 a1 14 00 00 00       mov     %gs:0x14,%eax
 8048dc5:   89 45 f4                mov     %eax,-0xc(%ebp)
 8048dc8:   31 c0                   xor     %eax,%eax
 8048dca:   53                      push    %ebx
 8048dcb:   e8 8f 02 00 00          call    804905f <string_length>
 8048dd0:   83 c4 10                add     $0x10,%esp
 8048dd3:   83 f8 06                cmp     $0x6,%eax                  //%eax = 6
 8048dd6:   75 4c                   jne     8048e24 <phase_5+0x6f>
    //%ebx的长度为6
 8048dd8:   b8 00 00 00 00          mov     $0x0,%eax             //%eax = 0

 8048ddd:   0f b6 14 03             movzbl (%ebx,%eax,1),%edx //%edx
= %ebx[0]
 8048de1:   83 e2 0f                and     $0xf,%edx             //取%edx的低
四位
 8048de4:   0f b6 92 e4 a2 04 08    movzbl                 0x804a2e4(%edx),%edx
    //0x804a2e4为数组首地址，%edx为下标，%edx = 数组[%edx]元素
```

```
8048deb:   88 54 05 ed              mov     %dl,-0x13(%ebp,%eax,1)     //
取%edx的低8位存进-0x13(%ebp,%eax,1)中
8048def:   83 c0 01                 add     $0x1,%eax        //%eax += 1
8048df2:   83 f8 06                 cmp     $0x6,%eax        //循环直到%eax = 6
退出    (0-5)
8048df5:   75 e6                    jne     8048ddd <phase_5+0x28>

8048df7:   c6 45 f3 00              movb    $0x0,-0xd(%ebp)
8048dfb:   83 ec 08                 sub     $0x8,%esp
8048dfe:   68 bb a2 04 08           push    $0x804a2bb          // “bruins”
8048e03:   8d 45 ed                 lea     -0x13(%ebp),%eax
8048e06:   50                       push    %eax
8048e07:   e8 75 02 00 00           call          8049081    <strings_not_equal>
    //equal即-0x13(%ebp)为首地址字符串 = “bruins”
8048e0c:   83 c4 10                 add     $0x10,%esp
8048e0f:   85 c0                    test    %eax,%eax
8048e11:   75 18                    jne               8048e2b    <phase_5+0x76>
    //%eax = 0即equal
8048e13:   8b 45 f4                 mov     -0xc(%ebp),%eax
8048e16:   65 33 05 14 00 00 00     xor     %gs:0x14,%eax
8048e1d:   75 13                    jne     8048e32 <phase_5+0x7d>
8048e1f:   8b 5d fc                 mov     -0x4(%ebp),%ebx
8048e22:   c9                       leave
8048e23:   c3                       ret
8048e24:   e8 8c 04 00 00           call    80492b5 <explode_bomb>
8048e29:   eb ad                    jmp     8048dd8 <phase_5+0x23>
8048e2b:   e8 85 04 00 00           call    80492b5 <explode_bomb>
8048e30:   eb e1                    jmp     8048e13 <phase_5+0x5e>
8048e32:   e8 59 f9 ff ff           call    8048790 <__stack_chk_fail@plt>
```

```
08048e37 <phase_6>:
 8048e37:   55                      push   %ebp
 8048e38:   89 e5                   mov    %esp,%ebp
 8048e3a:   56                      push   %esi
 8048e3b:   53                      push   %ebx
 8048e3c:   83 ec 48                sub    $0x48,%esp
 8048e3f:   65 a1 14 00 00 00       mov    %gs:0x14,%eax
 8048e45:   89 45 f4                mov    %eax,-0xc(%ebp)
 8048e48:   31 c0                   xor    %eax,%eax
 8048e4a:   8d 45 c4                lea    -0x3c(%ebp),%eax
 8048e4d:   50                      push   %eax
 8048e4e:   ff 75 08                pushl  0x8(%ebp)
 8048e51:   e8 9f 04 00 00          call   80492f5 <read_six_numbers> //读取
6个数
 //
 8048e56:   83 c4 10                add    $0x10,%esp
 8048e59:   be 00 00 00 00          mov    $0x0,%esi               //%esi = 0

 8048e5e:   8b 44 b5 c4             mov              -0x3c(%ebp,%esi,4),%eax
    //%eax = %esi * 4 - 3c
 8048e62:   83 e8 01                sub    $0x1,%eax               //%eax  -=
1
 8048e65:   83 f8 05                cmp    $0x5,%eax
 8048e68:   77 0c                   ja     8048e76 <phase_6+0x3f>//%eax  <=
5 %eax > 5则炸

 8048e6a:   83 c6 01                add    $0x1,%esi               //%esi  +=
```

1

```
 8048e6d:   83 fe 06                cmp     $0x6,%esi
 8048e70:   74 51                   je      8048ec3 <phase_6+0x8c>//%esi   ==
6则跳转
 8048e72:   89 f3                   mov     %esi,%ebx                 //%ebx
= %esi
 8048e74:   eb 0f                   jmp     8048e85 <phase_6+0x4e>
 8048e76:   e8 3a 04 00 00          call    80492b5 <explode_bomb>
 8048e7b:   eb ed                   jmp     8048e6a <phase_6+0x33>

 8048e7d:   83 c3 01                add     $0x1,%ebx             //%ebx += 1
 8048e80:   83 fb 05                cmp     $0x5,%ebx
 8048e83:   7f d9                   jg              8048e5e    <phase_6+0x27>
    //%ebx > 5则跳转即%ebx = 6
 8048e85:   8b 44 9d c4             mov               -0x3c(%ebp,%ebx,4),%eax
    //%eax = %ebx * 4 - 3c
 8048e89:   39 44 b5 c0             cmp     %eax,-0x40(%ebp,%esi,4)
 8048e8d:   75 ee                   jne     8048e7d <phase_6+0x46>
    //%eax != %esi * 4 - 40
 //以上循环说明6个数不相等
 8048e8f:   e8 21 04 00 00          call    80492b5 <explode_bomb>
 8048e94:   eb e7                   jmp     8048e7d <phase_6+0x46>

 //
 //
 8048e96:   8b 52 08                mov     0x8(%edx),%edx      //%edx        =
(%edx) + 8
 8048e99:   83 c0 01                add     $0x1,%eax             //%eax += 1
 8048e9c:   39 c8                   cmp     %ecx,%eax
```

```
 8048e9e:   75 f6                    jne          8048e96    <phase_6+0x5f>
    //%eax != %ecx则跳转

 8048ea0:   89 54 b5 dc              mov     %edx,-0x24(%ebp,%esi,4)    //%esi
* 4 - 24 = %edx
 8048ea4:   83 c3 01                 add     $0x1,%ebx           //%ebx += 1
 8048ea7:   83 fb 06                 cmp     $0x6,%ebx
 8048eaa:   74 1e                    je           8048eca    <phase_6+0x93>
    //%ebx = 6则跳转

 8048eac:   89 de                    mov     %ebx,%esi               //%esi
= %ebx
 8048eae:   8b 4c 9d c4              mov          -0x3c(%ebp,%ebx,4),%ecx
    //%ecx = %ebx * 4 - 3c
 8048eb2:   b8 01 00 00 00           mov     $0x1,%eax               //%eax = 1
 8048eb7:   ba 54 c1 04 08           mov     $0x804c154,%edx
    //$0x804c154 = 1be
 8048ebc:   83 f9 01                 cmp     $0x1,%ecx
 8048ebf:   7f d5                    jg      8048e96 <phase_6+0x5f> //%ecx > 1
则跳转
 8048ec1:   eb dd                    jmp     8048ea0 <phase_6+0x69>
 //使输入的6个数匹配链表的序号并使-0x24 - -0x10表示相应的数

 8048ec3:   bb 00 00 00 00           mov     $0x0,%ebx           //%ebx = 0
 8048ec8:   eb e2                    jmp     8048eac <phase_6+0x75>

 8048eca:   8b 5d dc                 mov     -0x24(%ebp),%ebx        //%ebx   =
-24
 8048ecd:   8b 45 e0                 mov     -0x20(%ebp),%eax        //%eax   =
```

-20

8048ed0: 89 43 08 mov %eax,0x8(%ebx) //%ebx + 8 = %eax

8048ed3: 8b 55 e4 mov -0x1c(%ebp),%edx //%edx = -1c

8048ed6: 89 50 08 mov %edx,0x8(%eax) //%eax + 8 = %edx

8048ed9: 8b 45 e8 mov -0x18(%ebp),%eax //%eax = -18

8048edc: 89 42 08 mov %eax,0x8(%edx) //%edx + 8 = %eax

8048edf: 8b 55 ec mov -0x14(%ebp),%edx //%edx = -14

8048ee2: 89 50 08 mov %edx,0x8(%eax) //%eax + 8 = %edx

8048ee5: 8b 45 f0 mov -0x10(%ebp),%eax //%eax = -10

8048ee8: 89 42 08 mov %eax,0x8(%edx) //%edx + 8 = %eax

8048eeb: c7 40 08 00 00 00 00 movl $0x0,0x8(%eax) //%eax + 8 = 0

8048ef2: be 05 00 00 00 mov $0x5,%esi //%esi = 5

8048ef7: eb 08 jmp 8048f01 <phase_6+0xca>

8048ef9: 8b 5b 08 mov 0x8(%ebx),%ebx //%ebx = %ebx + 8

8048efc: 83 ee 01 sub $0x1,%esi //%esi -= 1

8048eff: 74 10 je 8048f11 <phase_6+0xda>//%esi = 0

结束

```
 8048f01:   8b 43 08                mov    0x8(%ebx),%eax      //%eax = %ebx
+ 8
 8048f04:   8b 00                   mov    (%eax),%eax              //%eax   =
(%eax)
 8048f06:   39 03                   cmp    %eax,(%ebx)
 8048f08:   7d ef                   jge    8048ef9 <phase_6+0xc2> //%eax  >=
(%ebx)
 8048f0a:   e8 a6 03 00 00          call   80492b5 <explode_bomb>
 8048f0f:   eb e8                   jmp    8048ef9 <phase_6+0xc2>

 8048f11:   8b 45 f4                mov    -0xc(%ebp),%eax          //%eax   =
-c
 8048f14:   65 33 05 14 00 00 00    xor    %gs:0x14,%eax
 8048f1b:   75 07                   jne    8048f24 <phase_6+0xed>
 8048f1d:   8d 65 f8                lea    -0x8(%ebp),%esp
 8048f20:   5b                      pop    %ebx
 8048f21:   5e                      pop    %esi
 8048f22:   5d                      pop    %ebp
 8048f23:   c3                      ret
 8048f24:   e8 67 f8 ff ff          call   8048790 <__stack_chk_fail@plt>


08048f29 <fun7>:
 8048f29:   55                      push   %ebp
 8048f2a:   89 e5                   mov    %esp,%ebp
 8048f2c:   53                      push   %ebx
 8048f2d:   83 ec 04                sub    $0x4,%esp
 8048f30:   8b 55 08                mov    0x8(%ebp),%edx          //%edx   =
24H
```

```
8048f33:   8b 4d 0c              mov     0xc(%ebp),%ecx      //%ecx = j
8048f36:   85 d2                 test    %edx,%edx
8048f38:   74 3c                 je      8048f76 <fun7+0x4d>
8048f3a:   8b 1a                 mov     (%edx),%ebx         //%ebx =
(%edx)
8048f3c:   39 cb                 cmp     %ecx,%ebx
8048f3e:   7f 0e                 jg      8048f4e <fun7+0x25>         //
若%ecx > *%ebx则跳转
8048f40:   b8 00 00 00 00        mov     $0x0,%eax       //%eax = 0
8048f45:   39 cb                 cmp     %ecx,%ebx
8048f47:   75 18                 jne     8048f61 <fun7+0x38>   //若相等则返
回

8048f49:   8b 5d fc              mov     -0x4(%ebp),%ebx
8048f4c:   c9                    leave
8048f4d:   c3                    ret

8048f4e:   83 ec 08              sub     $0x8,%esp
8048f51:   51                    push    %ecx
8048f52:   ff 72 04              pushl   0x4(%edx)
8048f55:   e8 cf ff ff ff        call    8048f29 <fun7>      //fun7((%edx)+4, j)

8048f5a:   83 c4 10              add     $0x10,%esp
8048f5d:   01 c0                 add     %eax,%eax
8048f5f:   eb e8                 jmp             8048f49     <fun7+0x20>
//%eax=2*%eax

8048f61:   83 ec 08              sub     $0x8,%esp
8048f64:   51                    push    %ecx
```

```
 8048f65:   ff 72 08              pushl  0x8(%edx)
 8048f68:   e8 bc ff ff ff        call   8048f29 <fun7>      //fun7((%edx)+8, j)
 8048f6d:   83 c4 10              add    $0x10,%esp
 8048f70:   8d 44 00 01           lea              0x1(%eax,%eax,1),%eax
//%eax=2*%eax+1
 8048f74:   eb d3                 jmp    8048f49 <fun7+0x20>
 8048f76:   b8 ff ff ff ff        mov    $0xffffffff,%eax
 8048f7b:   eb cc                 jmp    8048f49 <fun7+0x20>

08048f7d <secret_phase>:
 8048f7d:   55                    push   %ebp
 8048f7e:   89 e5                 mov    %esp,%ebp
 8048f80:   53                    push   %ebx
 8048f81:   83 ec 04              sub    $0x4,%esp
 8048f84:   e8 a6 03 00 00        call   804932f <read_line>
 8048f89:   83 ec 04              sub    $0x4,%esp
 8048f8c:   6a 0a                 push   $0xa
 8048f8e:   6a 00                 push   $0x0
 8048f90:   50                    push   %eax
 8048f91:   e8 ea f8 ff ff        call   8048880 <strtol@plt>
 8048f96:   89 c3                 mov    %eax,%ebx          //%eax      =
j, %ebx = j
 8048f98:   8d 40 ff              lea    -0x1(%eax),%eax     //%eax = j - 1
 8048f9b:   83 c4 10              add    $0x10,%esp
 8048f9e:   3d e8 03 00 00        cmp    $0x3e8,%eax
 8048fa3:   77 35                 ja           8048fda <secret_phase+0x5d>
    //%eax(j - 1) <= 3e8
 8048fa5:   83 ec 08              sub    $0x8,%esp
 8048fa8:   53                    push   %ebx                //%ebx = j
```

```
 8048fa9:   68 a0 c0 04 08          push   $0x804c0a0     //0x804c0a0      =
24H
 8048fae:   e8 76 ff ff ff         call   8048f29 <fun7>
 8048fb3:   83 c4 10                add    $0x10,%esp
 8048fb6:   83 f8 04                cmp    $0x4,%eax
 8048fb9:   74 05                   je             8048fc0   <secret_phase+0x43>
    //%eax = 4
 8048fbb:   e8 f5 02 00 00          call   80492b5 <explode_bomb>
 8048fc0:   83 ec 0c                sub    $0xc,%esp
 8048fc3:   68 8c a2 04 08          push   $0x804a28c
 8048fc8:   e8 f3 f7 ff ff         call   80487c0 <puts@plt>
 8048fcd:   e8 6e 04 00 00          call   8049440 <phase_defused>
 8048fd2:   83 c4 10                add    $0x10,%esp
 8048fd5:   8b 5d fc                mov    -0x4(%ebp),%ebx
 8048fd8:   c9                      leave
 8048fd9:   c3                      ret
 8048fda:   e8 d6 02 00 00          call   80492b5 <explode_bomb>
 8048fdf:   eb c4                   jmp    8048fa5 <secret_phase+0x28>

08048fe1 <sig_handler>:
 8048fe1:   55                      push   %ebp
 8048fe2:   89 e5                   mov    %esp,%ebp
 8048fe4:   83 ec 14                sub    $0x14,%esp
 8048fe7:   68 f4 a2 04 08          push   $0x804a2f4
 8048fec:   e8 cf f7 ff ff         call   80487c0 <puts@plt>
 8048ff1:   c7 04 24 03 00 00 00    movl   $0x3,(%esp)
 8048ff8:   e8 73 f7 ff ff         call   8048770 <sleep@plt>
 8048ffd:   83 c4 08                add    $0x8,%esp
 8049000:   68 41 a4 04 08          push   $0x804a441
```

```
 8049005:   6a 01                    push   $0x1
 8049007:   e8 34 f8 ff ff           call   8048840 <__printf_chk@plt>
 804900c:   83 c4 04                 add    $0x4,%esp
 804900f:   ff 35 e4 c7 04 08        pushl  0x804c7e4
 8049015:   e8 26 f7 ff ff           call   8048740 <fflush@plt>
 804901a:   c7 04 24 01 00 00 00     movl   $0x1,(%esp)
 8049021:   e8 4a f7 ff ff           call   8048770 <sleep@plt>
 8049026:   c7 04 24 49 a4 04 08     movl   $0x804a449,(%esp)
 804902d:   e8 8e f7 ff ff           call   80487c0 <puts@plt>
 8049032:   c7 04 24 10 00 00 00     movl   $0x10,(%esp)
 8049039:   e8 a2 f7 ff ff           call   80487e0 <exit@plt>

0804903e <invalid_phase>:
 804903e:   55                       push   %ebp
 804903f:   89 e5                    mov    %esp,%ebp
 8049041:   83 ec 0c                 sub    $0xc,%esp
 8049044:   ff 75 08                 pushl  0x8(%ebp)
 8049047:   68 51 a4 04 08           push   $0x804a451
 804904c:   6a 01                    push   $0x1
 804904e:   e8 ed f7 ff ff           call   8048840 <__printf_chk@plt>
 8049053:   c7 04 24 08 00 00 00     movl   $0x8,(%esp)
 804905a:   e8 81 f7 ff ff           call   80487e0 <exit@plt>

0804905f <string_length>:
 804905f:   55                       push   %ebp
 8049060:   89 e5                    mov    %esp,%ebp
 8049062:   8b 55 08                 mov    0x8(%ebp),%edx
 8049065:   80 3a 00                 cmpb   $0x0,(%edx)
 8049068:   74 10                    je     804907a <string_length+0x1b>
```

```
804906a:  b8 00 00 00 00          mov     $0x0,%eax
804906f:  83 c0 01                add     $0x1,%eax
8049072:  80 3c 02 00             cmpb    $0x0,(%edx,%eax,1)
8049076:  75 f7                   jne     804906f <string_length+0x10>
8049078:  5d                      pop     %ebp
8049079:  c3                      ret
804907a:  b8 00 00 00 00          mov     $0x0,%eax
804907f:  eb f7                   jmp     8049078 <string_length+0x19>

08049081 <strings_not_equal>:
8049081:  55                      push    %ebp
8049082:  89 e5                   mov     %esp,%ebp
8049084:  57                      push    %edi
8049085:  56                      push    %esi
8049086:  53                      push    %ebx
8049087:  8b 5d 08                mov     0x8(%ebp),%ebx
804908a:  8b 75 0c                mov     0xc(%ebp),%esi
804908d:  53                      push    %ebx
804908e:  e8 cc ff ff ff          call    804905f <string_length>
8049093:  89 c7                   mov     %eax,%edi
8049095:  89 34 24                mov     %esi,(%esp)
8049098:  e8 c2 ff ff ff          call    804905f <string_length>
804909d:  83 c4 04                add     $0x4,%esp
80490a0:  ba 01 00 00 00          mov     $0x1,%edx
80490a5:  39 c7                   cmp     %eax,%edi
80490a7:  74 0a                   je      80490b3 <strings_not_equal+0x32>
80490a9:  89 d0                   mov     %edx,%eax
80490ab:  8d 65 f4                lea     -0xc(%ebp),%esp
80490ae:  5b                      pop     %ebx
```

```
80490af:   5e                    pop    %esi
80490b0:   5f                    pop    %edi
80490b1:   5d                    pop    %ebp
80490b2:   c3                    ret
80490b3:   0f b6 03              movzbl (%ebx),%eax
80490b6:   84 c0                 test   %al,%al
80490b8:   74 23                 je     80490dd <strings_not_equal+0x5c>
80490ba:   3a 06                 cmp    (%esi),%al
80490bc:   75 26                 jne    80490e4 <strings_not_equal+0x63>
80490be:   83 c3 01              add    $0x1,%ebx
80490c1:   83 c6 01              add    $0x1,%esi
80490c4:   0f b6 03              movzbl (%ebx),%eax
80490c7:   84 c0                 test   %al,%al
80490c9:   74 0b                 je     80490d6 <strings_not_equal+0x55>
80490cb:   38 06                 cmp    %al,(%esi)
80490cd:   74 ef                 je     80490be <strings_not_equal+0x3d>
80490cf:   ba 01 00 00 00        mov    $0x1,%edx
80490d4:   eb d3                 jmp    80490a9 <strings_not_equal+0x28>
80490d6:   ba 00 00 00 00        mov    $0x0,%edx
80490db:   eb cc                 jmp    80490a9 <strings_not_equal+0x28>
80490dd:   ba 00 00 00 00        mov    $0x0,%edx
80490e2:   eb c5                 jmp    80490a9 <strings_not_equal+0x28>
80490e4:   ba 01 00 00 00        mov    $0x1,%edx
80490e9:   eb be                 jmp    80490a9 <strings_not_equal+0x28>

080490eb <initialize_bomb>:
80490eb:   55                    push   %ebp
80490ec:   89 e5                 mov    %esp,%ebp
80490ee:   81 ec 20 20 00 00     sub    $0x2020,%esp
```

```
 80490f4:   65 a1 14 00 00 00        mov     %gs:0x14,%eax
 80490fa:   89 45 f4                 mov     %eax,-0xc(%ebp)
 80490fd:   31 c0                    xor     %eax,%eax
 80490ff:   68 e1 8f 04 08           push    $0x8048fe1
 8049104:   6a 02                    push    $0x2
 8049106:   e8 55 f6 ff ff           call    8048760 <signal@plt>
 804910b:   8d 85 f4 df ff ff        lea     -0x200c(%ebp),%eax
 8049111:   89 04 24                 mov     %eax,(%esp)
 8049114:   e8 2b 0d 00 00           call    8049e44 <init_driver>
 8049119:   83 c4 10                 add     $0x10,%esp
 804911c:   85 c0                    test    %eax,%eax
 804911e:   78 0e                    js      804912e <initialize_bomb+0x43>
 8049120:   8b 45 f4                 mov     -0xc(%ebp),%eax
 8049123:   65 33 05 14 00 00 00     xor     %gs:0x14,%eax
 804912a:   75 24                    jne     8049150 <initialize_bomb+0x65>
 804912c:   c9                       leave
 804912d:   c3                       ret
 804912e:   83 ec 04                 sub     $0x4,%esp
 8049131:   8d 85 f4 df ff ff        lea     -0x200c(%ebp),%eax
 8049137:   50                       push    %eax
 8049138:   68 62 a4 04 08           push    $0x804a462
 804913d:   6a 01                    push    $0x1
 804913f:   e8 fc f6 ff ff           call    8048840 <__printf_chk@plt>
 8049144:   c7 04 24 08 00 00 00     movl    $0x8,(%esp)
 804914b:   e8 90 f6 ff ff           call    80487e0 <exit@plt>
 8049150:   e8 3b f6 ff ff           call    8048790 <__stack_chk_fail@plt>

08049155 <initialize_bomb_solve>:
 8049155:   55                       push    %ebp
```

```
 8049156:  89 e5                 mov    %esp,%ebp
 8049158:  5d                    pop    %ebp
 8049159:  c3                    ret

0804915a <blank_line>:
 804915a:  55                    push   %ebp
 804915b:  89 e5                 mov    %esp,%ebp
 804915d:  56                    push   %esi
 804915e:  53                    push   %ebx
 804915f:  8b 75 08              mov    0x8(%ebp),%esi
 8049162:  0f b6 1e              movzbl (%esi),%ebx
 8049165:  84 db                 test   %bl,%bl
 8049167:  74 1b                 je     8049184 <blank_line+0x2a>
 8049169:  e8 42 f7 ff ff        call   80488b0 <__ctype_b_loc@plt>
 804916e:  83 c6 01              add    $0x1,%esi
 8049171:  0f be db              movsbl %bl,%ebx
 8049174:  8b 00                 mov    (%eax),%eax
 8049176:  f6 44 58 01 20        testb  $0x20,0x1(%eax,%ebx,2)
 804917b:  75 e5                 jne    8049162 <blank_line+0x8>
 804917d:  b8 00 00 00 00        mov    $0x0,%eax
 8049182:  eb 05                 jmp    8049189 <blank_line+0x2f>
 8049184:  b8 01 00 00 00        mov    $0x1,%eax
 8049189:  5b                    pop    %ebx
 804918a:  5e                    pop    %esi
 804918b:  5d                    pop    %ebp
 804918c:  c3                    ret

0804918d <skip>:
 804918d:  55                    push   %ebp
```

```
 804918e:   89 e5                    mov     %esp,%ebp
 8049190:   53                       push    %ebx
 8049191:   83 ec 04                 sub     $0x4,%esp
 8049194:   83 ec 04                 sub     $0x4,%esp
 8049197:   ff 35 f0 c7 04 08        pushl   0x804c7f0
 804919d:   6a 50                    push    $0x50
 804919f:   a1 ec c7 04 08           mov     0x804c7ec,%eax
 80491a4:   8d 04 80                 lea     (%eax,%eax,4),%eax
 80491a7:   c1 e0 04                 shl     $0x4,%eax
 80491aa:   05 00 c8 04 08           add     $0x804c800,%eax
 80491af:   50                       push    %eax
 80491b0:   e8 9b f5 ff ff           call    8048750 <fgets@plt>
 80491b5:   89 c3                    mov     %eax,%ebx
 80491b7:   83 c4 10                 add     $0x10,%esp
 80491ba:   85 c0                    test    %eax,%eax
 80491bc:   74 10                    je      80491ce <skip+0x41>
 80491be:   83 ec 0c                 sub     $0xc,%esp
 80491c1:   50                       push    %eax
 80491c2:   e8 93 ff ff ff           call    804915a <blank_line>
 80491c7:   83 c4 10                 add     $0x10,%esp
 80491ca:   85 c0                    test    %eax,%eax
 80491cc:   75 c6                    jne     8049194 <skip+0x7>
 80491ce:   89 d8                    mov     %ebx,%eax
 80491d0:   8b 5d fc                 mov     -0x4(%ebp),%ebx
 80491d3:   c9                       leave
 80491d4:   c3                       ret

080491d5 <send_msg>:
 80491d5:   55                       push    %ebp
```

```
80491d6:  89 e5                  mov     %esp,%ebp
80491d8:  57                     push    %edi
80491d9:  56                     push    %esi
80491da:  53                     push    %ebx
80491db:  81 ec 1c 40 00 00      sub     $0x401c,%esp
80491e1:  65 a1 14 00 00 00      mov     %gs:0x14,%eax
80491e7:  89 45 e4               mov     %eax,-0x1c(%ebp)
80491ea:  31 c0                  xor     %eax,%eax
80491ec:  8b 1d ec c7 04 08      mov     0x804c7ec,%ebx
80491f2:  8d 54 9b fb            lea     -0x5(%ebx,%ebx,4),%edx
80491f6:  c1 e2 04               shl     $0x4,%edx
80491f9:  81 c2 00 c8 04 08      add     $0x804c800,%edx
80491ff:  b9 ff ff ff ff         mov     $0xffffffff,%ecx
8049204:  89 d7                  mov     %edx,%edi
8049206:  f2 ae                  repnz scas %es:(%edi),%al
8049208:  89 c8                  mov     %ecx,%eax
804920a:  f7 d0                  not     %eax
804920c:  83 c0 63               add     $0x63,%eax
804920f:  3d 00 20 00 00         cmp     $0x2000,%eax
8049214:  77 64                  ja      804927a <send_msg+0xa5>
8049216:  83 7d 08 00            cmpl    $0x0,0x8(%ebp)
804921a:  b8 7c a4 04 08         mov     $0x804a47c,%eax
804921f:  b9 84 a4 04 08         mov     $0x804a484,%ecx
8049224:  0f 44 c1               cmove   %ecx,%eax
8049227:  52                     push    %edx
8049228:  53                     push    %ebx
8049229:  50                     push    %eax
804922a:  ff 35 a0 c5 04 08      pushl   0x804c5a0
8049230:  68 8d a4 04 08         push    $0x804a48d
```

```
8049235:  68 00 20 00 00          push   $0x2000
804923a:  6a 01                   push   $0x1
804923c:  8d 9d e4 bf ff ff       lea    -0x401c(%ebp),%ebx
8049242:  53                      push   %ebx
8049243:  e8 78 f6 ff ff          call   80488c0 <__sprintf_chk@plt>
8049248:  83 c4 20                add    $0x20,%esp
804924b:  8d 85 e4 df ff ff       lea    -0x201c(%ebp),%eax
8049251:  50                      push   %eax
8049252:  6a 00                   push   $0x0
8049254:  53                      push   %ebx
8049255:  68 a0 c1 04 08          push   $0x804c1a0
804925a:  e8 bf 0d 00 00          call   804a01e <driver_post>
804925f:  83 c4 10                add    $0x10,%esp
8049262:  85 c0                   test   %eax,%eax
8049264:  78 2f                   js     8049295 <send_msg+0xc0>
8049266:  8b 45 e4                mov    -0x1c(%ebp),%eax
8049269:  65 33 05 14 00 00 00    xor    %gs:0x14,%eax
8049270:  75 3e                   jne    80492b0 <send_msg+0xdb>
8049272:  8d 65 f4                lea    -0xc(%ebp),%esp
8049275:  5b                      pop    %ebx
8049276:  5e                      pop    %esi
8049277:  5f                      pop    %edi
8049278:  5d                      pop    %ebp
8049279:  c3                      ret
804927a:  83 ec 08                sub    $0x8,%esp
804927d:  68 2c a3 04 08          push   $0x804a32c
8049282:  6a 01                   push   $0x1
8049284:  e8 b7 f5 ff ff          call   8048840 <__printf_chk@plt>
8049289:  c7 04 24 08 00 00 00    movl   $0x8,(%esp)
```

```
 8049290:   e8 4b f5 ff ff          call    80487e0 <exit@plt>
 8049295:   83 ec 0c                sub     $0xc,%esp
 8049298:   8d 85 e4 df ff ff       lea     -0x201c(%ebp),%eax
 804929e:   50                      push    %eax
 804929f:   e8 1c f5 ff ff          call    80487c0 <puts@plt>
 80492a4:   c7 04 24 00 00 00 00    movl    $0x0,(%esp)
 80492ab:   e8 30 f5 ff ff          call    80487e0 <exit@plt>
 80492b0:   e8 db f4 ff ff          call    8048790 <__stack_chk_fail@plt>

080492b5 <explode_bomb>:
 80492b5:   55                      push    %ebp
 80492b6:   89 e5                   mov     %esp,%ebp
 80492b8:   83 ec 14                sub     $0x14,%esp
 80492bb:   68 99 a4 04 08          push    $0x804a499
 80492c0:   e8 fb f4 ff ff          call    80487c0 <puts@plt>
 80492c5:   c7 04 24 a2 a4 04 08    movl    $0x804a4a2,(%esp)
 80492cc:   e8 ef f4 ff ff          call    80487c0 <puts@plt>
 80492d1:   c7 04 24 00 00 00 00    movl    $0x0,(%esp)
 80492d8:   e8 f8 fe ff ff          call    80491d5 <send_msg>
 80492dd:   c7 04 24 50 a3 04 08    movl    $0x804a350,(%esp)
 80492e4:   e8 d7 f4 ff ff          call    80487c0 <puts@plt>
 80492e9:   c7 04 24 08 00 00 00    movl    $0x8,(%esp)
 80492f0:   e8 eb f4 ff ff          call    80487e0 <exit@plt>

080492f5 <read_six_numbers>:
 80492f5:   55                      push    %ebp
 80492f6:   89 e5                   mov     %esp,%ebp
 80492f8:   83 ec 08                sub     $0x8,%esp
 80492fb:   8b 45 0c                mov     0xc(%ebp),%eax
```

```
 80492fe:   8d 50 14                lea      0x14(%eax),%edx
 8049301:   52                      push     %edx
 8049302:   8d 50 10                lea      0x10(%eax),%edx
 8049305:   52                      push     %edx
 8049306:   8d 50 0c                lea      0xc(%eax),%edx
 8049309:   52                      push     %edx
 804930a:   8d 50 08                lea      0x8(%eax),%edx
 804930d:   52                      push     %edx
 804930e:   8d 50 04                lea      0x4(%eax),%edx
 8049311:   52                      push     %edx
 8049312:   50                      push     %eax
 8049313:   68 b9 a4 04 08          push     $0x804a4b9
 8049318:   ff 75 08                pushl    0x8(%ebp)
 804931b:   e8 f0 f4 ff ff          call     8048810 <__isoc99_sscanf@plt>
 8049320:   83 c4 20                add      $0x20,%esp
 8049323:   83 f8 05                cmp      $0x5,%eax
 8049326:   7e 02                   jle      804932a <read_six_numbers+0x35>
 8049328:   c9                      leave
 8049329:   c3                      ret
 804932a:   e8 86 ff ff ff          call     80492b5 <explode_bomb>

0804932f <read_line>:
 804932f:   55                      push     %ebp
 8049330:   89 e5                   mov      %esp,%ebp
 8049332:   57                      push     %edi
 8049333:   56                      push     %esi
 8049334:   53                      push     %ebx
 8049335:   83 ec 0c                sub      $0xc,%esp
 8049338:   e8 50 fe ff ff          call     804918d <skip>
```

```
804933d:  85 c0                   test   %eax,%eax
804933f:  74 53                   je     8049394 <read_line+0x65>
8049341:  8b 15 ec c7 04 08       mov    0x804c7ec,%edx
8049347:  8d 1c 92                lea    (%edx,%edx,4),%ebx
804934a:  c1 e3 04                shl    $0x4,%ebx
804934d:  81 c3 00 c8 04 08       add    $0x804c800,%ebx
8049353:  b9 ff ff ff ff          mov    $0xffffffff,%ecx
8049358:  b8 00 00 00 00          mov    $0x0,%eax
804935d:  89 df                   mov    %ebx,%edi
804935f:  f2 ae                   repnz scas %es:(%edi),%al
8049361:  89 ce                   mov    %ecx,%esi
8049363:  f7 d6                   not    %esi
8049365:  89 f1                   mov    %esi,%ecx
8049367:  83 e9 01                sub    $0x1,%ecx
804936a:  83 f9 4e                cmp    $0x4e,%ecx
804936d:  0f 8f 95 00 00 00       jg     8049408 <read_line+0xd9>
8049373:  8d 04 92                lea    (%edx,%edx,4),%eax
8049376:  c1 e0 04                shl    $0x4,%eax
8049379:  c6 84 01 ff c7 04 08    movb   $0x0,0x804c7ff(%ecx,%eax,1)
8049380:  00
8049381:  83 c2 01                add    $0x1,%edx
8049384:  89 15 ec c7 04 08       mov    %edx,0x804c7ec
804938a:  89 d8                   mov    %ebx,%eax
804938c:  8d 65 f4                lea    -0xc(%ebp),%esp
804938f:  5b                      pop    %ebx
8049390:  5e                      pop    %esi
8049391:  5f                      pop    %edi
8049392:  5d                      pop    %ebp
8049393:  c3                      ret
```

```
8049394:  a1 e0 c7 04 08          mov    0x804c7e0,%eax
8049399:  39 05 f0 c7 04 08       cmp    %eax,0x804c7f0
804939f:  74 1e                   je     80493bf <read_line+0x90>
80493a1:  83 ec 0c                sub    $0xc,%esp
80493a4:  68 e9 a4 04 08          push   $0x804a4e9
80493a9:  e8 02 f4 ff ff          call   80487b0 <getenv@plt>
80493ae:  83 c4 10                add    $0x10,%esp
80493b1:  85 c0                   test   %eax,%eax
80493b3:  74 23                   je     80493d8 <read_line+0xa9>
80493b5:  83 ec 0c                sub    $0xc,%esp
80493b8:  6a 00                   push   $0x0
80493ba:  e8 21 f4 ff ff          call   80487e0 <exit@plt>
80493bf:  83 ec 0c                sub    $0xc,%esp
80493c2:  68 cb a4 04 08          push   $0x804a4cb
80493c7:  e8 f4 f3 ff ff          call   80487c0 <puts@plt>
80493cc:  c7 04 24 08 00 00 00    movl   $0x8,(%esp)
80493d3:  e8 08 f4 ff ff          call   80487e0 <exit@plt>
80493d8:  a1 e0 c7 04 08          mov    0x804c7e0,%eax
80493dd:  a3 f0 c7 04 08          mov    %eax,0x804c7f0
80493e2:  e8 a6 fd ff ff          call   804918d <skip>
80493e7:  85 c0                   test   %eax,%eax
80493e9:  0f 85 52 ff ff ff       jne    8049341 <read_line+0x12>
80493ef:  83 ec 0c                sub    $0xc,%esp
80493f2:  68 cb a4 04 08          push   $0x804a4cb
80493f7:  e8 c4 f3 ff ff          call   80487c0 <puts@plt>
80493fc:  c7 04 24 00 00 00 00    movl   $0x0,(%esp)
8049403:  e8 d8 f3 ff ff          call   80487e0 <exit@plt>
8049408:  83 ec 0c                sub    $0xc,%esp
804940b:  68 f4 a4 04 08          push   $0x804a4f4
```

```
 8049410:  e8 ab f3 ff ff          call   80487c0 <puts@plt>
 8049415:  a1 ec c7 04 08          mov    0x804c7ec,%eax
 804941a:  8d 50 01                lea    0x1(%eax),%edx
 804941d:  89 15 ec c7 04 08       mov    %edx,0x804c7ec
 8049423:  6b c0 50                imul   $0x50,%eax,%eax
 8049426:  05 00 c8 04 08          add    $0x804c800,%eax
 804942b:  ba 0f a5 04 08          mov    $0x804a50f,%edx
 8049430:  b9 04 00 00 00          mov    $0x4,%ecx
 8049435:  89 c7                   mov    %eax,%edi
 8049437:  89 d6                   mov    %edx,%esi
 8049439:  f3 a5                   rep movsl %ds:(%esi),%es:(%edi)
 804943b:  e8 75 fe ff ff          call   80492b5 <explode_bomb>

08049440 <phase_defused>:
 8049440:  55                      push   %ebp
 8049441:  89 e5                   mov    %esp,%ebp
 8049443:  83 ec 74                sub    $0x74,%esp
 8049446:  65 a1 14 00 00 00       mov    %gs:0x14,%eax
 804944c:  89 45 f4                mov    %eax,-0xc(%ebp)
 804944f:  31 c0                   xor    %eax,%eax
 8049451:  6a 01                   push   $0x1
 8049453:  e8 7d fd ff ff          call   80491d5 <send_msg>
 8049458:  83 c4 10                add    $0x10,%esp
 804945b:  83 3d ec c7 04 08 06    cmpl   $0x6,0x804c7ec
 8049462:  74 12                   je     8049476 <phase_defused+0x36>
 8049464:  8b 45 f4                mov    -0xc(%ebp),%eax
 8049467:  65 33 05 14 00 00 00    xor    %gs:0x14,%eax
 804946e:  0f 85 81 00 00 00       jne    80494f5 <phase_defused+0xb5>
 8049474:  c9                      leave
```

```
8049475:  c3                      ret
8049476:  83 ec 0c                sub    $0xc,%esp
8049479:  8d 45 a4                lea    -0x5c(%ebp),%eax
804947c:  50                      push   %eax
804947d:  8d 45 a0                lea    -0x60(%ebp),%eax
8049480:  50                      push   %eax
8049481:  8d 45 9c                lea    -0x64(%ebp),%eax
8049484:  50                      push   %eax
8049485:  68 1f a5 04 08          push   $0x804a51f
804948a:  68 f0 c8 04 08          push   $0x804c8f0
804948f:  e8 7c f3 ff ff          call   8048810 <__isoc99_sscanf@plt>
8049494:  83 c4 20                add    $0x20,%esp
8049497:  83 f8 03                cmp    $0x3,%eax
804949a:  74 1e                   je     80494ba <phase_defused+0x7a>
804949c:  83 ec 0c                sub    $0xc,%esp
804949f:  68 d4 a3 04 08          push   $0x804a3d4
80494a4:  e8 17 f3 ff ff          call   80487c0 <puts@plt>
80494a9:  c7 04 24 00 a4 04 08    movl   $0x804a400,(%esp)
80494b0:  e8 0b f3 ff ff          call   80487c0 <puts@plt>
80494b5:  83 c4 10                add    $0x10,%esp
80494b8:  eb aa                   jmp    8049464 <phase_defused+0x24>
80494ba:  83 ec 08                sub    $0x8,%esp
80494bd:  68 28 a5 04 08          push   $0x804a528
80494c2:  8d 45 a4                lea    -0x5c(%ebp),%eax
80494c5:  50                      push   %eax
80494c6:  e8 b6 fb ff ff          call   8049081 <strings_not_equal>
80494cb:  83 c4 10                add    $0x10,%esp
80494ce:  85 c0                   test   %eax,%eax
80494d0:  75 ca                   jne    804949c <phase_defused+0x5c>
```

```
 80494d2:   83 ec 0c                 sub    $0xc,%esp
 80494d5:   68 74 a3 04 08           push   $0x804a374
 80494da:   e8 e1 f2 ff ff           call   80487c0 <puts@plt>
 80494df:   c7 04 24 9c a3 04 08     movl   $0x804a39c,(%esp)
 80494e6:   e8 d5 f2 ff ff           call   80487c0 <puts@plt>
 80494eb:   e8 8d fa ff ff           call   8048f7d <secret_phase>
 80494f0:   83 c4 10                 add    $0x10,%esp
 80494f3:   eb a7                    jmp    804949c <phase_defused+0x5c>
 80494f5:   e8 96 f2 ff ff           call   8048790 <__stack_chk_fail@plt>

080494fa <sigalrm_handler>:
 80494fa:   55                       push   %ebp
 80494fb:   89 e5                    mov    %esp,%ebp
 80494fd:   83 ec 08                 sub    $0x8,%esp
 8049500:   6a 00                    push   $0x0
 8049502:   68 80 a5 04 08           push   $0x804a580
 8049507:   6a 01                    push   $0x1
 8049509:   ff 35 c0 c7 04 08        pushl  0x804c7c0
 804950f:   e8 4c f3 ff ff           call   8048860 <__fprintf_chk@plt>
 8049514:   c7 04 24 01 00 00 00     movl   $0x1,(%esp)
 804951b:   e8 c0 f2 ff ff           call   80487e0 <exit@plt>

08049520 <rio_readlineb>:
 8049520:   55                       push   %ebp
 8049521:   89 e5                    mov    %esp,%ebp
 8049523:   57                       push   %edi
 8049524:   56                       push   %esi
 8049525:   53                       push   %ebx
 8049526:   83 ec 1c                 sub    $0x1c,%esp
```

```
8049529:  89 d7                   mov    %edx,%edi
804952b:  83 f9 01                cmp    $0x1,%ecx
804952e:  76 7d                   jbe    80495ad <rio_readlineb+0x8d>
8049530:  89 c3                   mov    %eax,%ebx
8049532:  8d 44 0a ff             lea    -0x1(%edx,%ecx,1),%eax
8049536:  89 45 e0                mov    %eax,-0x20(%ebp)
8049539:  c7 45 e4 01 00 00 00    movl   $0x1,-0x1c(%ebp)
8049540:  8d 73 0c                lea    0xc(%ebx),%esi
8049543:  eb 0a                   jmp    804954f <rio_readlineb+0x2f>
8049545:  e8 e6 f2 ff ff          call   8048830 <__errno_location@plt>
804954a:  83 38 04                cmpl   $0x4,(%eax)
804954d:  75 67                   jne    80495b6 <rio_readlineb+0x96>
804954f:  8b 43 04                mov    0x4(%ebx),%eax
8049552:  85 c0                   test   %eax,%eax
8049554:  7f 23                   jg     8049579 <rio_readlineb+0x59>
8049556:  83 ec 04                sub    $0x4,%esp
8049559:  68 00 20 00 00          push   $0x2000
804955e:  56                      push   %esi
804955f:  ff 33                   pushl  (%ebx)
8049561:  e8 ca f1 ff ff          call   8048730 <read@plt>
8049566:  89 43 04                mov    %eax,0x4(%ebx)
8049569:  83 c4 10                add    $0x10,%esp
804956c:  85 c0                   test   %eax,%eax
804956e:  78 d5                   js     8049545 <rio_readlineb+0x25>
8049570:  85 c0                   test   %eax,%eax
8049572:  74 49                   je     80495bd <rio_readlineb+0x9d>
8049574:  89 73 08                mov    %esi,0x8(%ebx)
8049577:  eb d6                   jmp    804954f <rio_readlineb+0x2f>
8049579:  8b 4b 08                mov    0x8(%ebx),%ecx
```

```
804957c:   0f b6 11                movzbl (%ecx),%edx
804957f:   83 c1 01                add    $0x1,%ecx
8049582:   89 4b 08                mov    %ecx,0x8(%ebx)
8049585:   83 e8 01                sub    $0x1,%eax
8049588:   89 43 04                mov    %eax,0x4(%ebx)
804958b:   83 c7 01                add    $0x1,%edi
804958e:   88 57 ff                mov    %dl,-0x1(%edi)
8049591:   80 fa 0a                cmp    $0xa,%dl
8049594:   74 09                   je     804959f <rio_readlineb+0x7f>
8049596:   83 45 e4 01             addl   $0x1,-0x1c(%ebp)
804959a:   3b 7d e0                cmp    -0x20(%ebp),%edi
804959d:   75 b0                   jne    804954f <rio_readlineb+0x2f>
804959f:   c6 07 00                movb   $0x0,(%edi)
80495a2:   8b 45 e4                mov    -0x1c(%ebp),%eax
80495a5:   8d 65 f4                lea    -0xc(%ebp),%esp
80495a8:   5b                      pop    %ebx
80495a9:   5e                      pop    %esi
80495aa:   5f                      pop    %edi
80495ab:   5d                      pop    %ebp
80495ac:   c3                      ret
80495ad:   c7 45 e4 01 00 00 00    movl   $0x1,-0x1c(%ebp)
80495b4:   eb e9                   jmp    804959f <rio_readlineb+0x7f>
80495b6:   b8 ff ff ff ff          mov    $0xffffffff,%eax
80495bb:   eb 05                   jmp    80495c2 <rio_readlineb+0xa2>
80495bd:   b8 00 00 00 00          mov    $0x0,%eax
80495c2:   85 c0                   test   %eax,%eax
80495c4:   75 0f                   jne    80495d5 <rio_readlineb+0xb5>
80495c6:   83 7d e4 01             cmpl   $0x1,-0x1c(%ebp)
80495ca:   75 d3                   jne    804959f <rio_readlineb+0x7f>
```

```
 80495cc:   c7 45 e4 00 00 00 00    movl   $0x0,-0x1c(%ebp)
 80495d3:   eb cd                   jmp    80495a2 <rio_readlineb+0x82>
 80495d5:   c7 45 e4 ff ff ff ff    movl   $0xffffffff,-0x1c(%ebp)
 80495dc:   eb c4                   jmp    80495a2 <rio_readlineb+0x82>

080495de <submitr>:
 80495de:   55                      push   %ebp
 80495df:   89 e5                   mov    %esp,%ebp
 80495e1:   57                      push   %edi
 80495e2:   56                      push   %esi
 80495e3:   53                      push   %ebx
 80495e4:   81 ec 60 a0 00 00       sub    $0xa060,%esp
 80495ea:   8b 75 08                mov    0x8(%ebp),%esi
 80495ed:   8b 45 10                mov    0x10(%ebp),%eax
 80495f0:   89 85 ac 5f ff ff       mov    %eax,-0xa054(%ebp)
 80495f6:   8b 45 14                mov    0x14(%ebp),%eax
 80495f9:   89 85 a8 5f ff ff       mov    %eax,-0xa058(%ebp)
 80495ff:   8b 45 18                mov    0x18(%ebp),%eax
 8049602:   89 85 a4 5f ff ff       mov    %eax,-0xa05c(%ebp)
 8049608:   8b 5d 1c                mov    0x1c(%ebp),%ebx
 804960b:   8b 45 20                mov    0x20(%ebp),%eax
 804960e:   89 85 a0 5f ff ff       mov    %eax,-0xa060(%ebp)
 8049614:   65 a1 14 00 00 00       mov    %gs:0x14,%eax
 804961a:   89 45 e4                mov    %eax,-0x1c(%ebp)
 804961d:   31 c0                   xor    %eax,%eax
 804961f:   c7 85 c4 5f ff ff 00    movl   $0x0,-0xa03c(%ebp)
 8049626:   00 00 00
 8049629:   6a 00                   push   $0x0
 804962b:   6a 01                   push   $0x1
```

```
804962d:  6a 02                 push   $0x2
804962f:  e8 1c f2 ff ff        call   8048850 <socket@plt>
8049634:  89 85 b0 5f ff ff     mov    %eax,-0xa050(%ebp)
804963a:  83 c4 10              add    $0x10,%esp
804963d:  85 c0                 test   %eax,%eax
804963f:  0f 88 30 01 00 00     js     8049775 <submitr+0x197>
8049645:  83 ec 0c              sub    $0xc,%esp
8049648:  56                    push   %esi
8049649:  e8 22 f2 ff ff        call   8048870 <gethostbyname@plt>
804964e:  83 c4 10              add    $0x10,%esp
8049651:  85 c0                 test   %eax,%eax
8049653:  0f 84 70 01 00 00     je     80497c9 <submitr+0x1eb>
8049659:  8d b5 c8 5f ff ff     lea    -0xa038(%ebp),%esi
804965f:  c7 85 ca 5f ff ff 00  movl   $0x0,-0xa036(%ebp)
8049666:  00 00 00
8049669:  c7 85 ce 5f ff ff 00  movl   $0x0,-0xa032(%ebp)
8049670:  00 00 00
8049673:  c7 85 d2 5f ff ff 00  movl   $0x0,-0xa02e(%ebp)
804967a:  00 00 00
804967d:  66 c7 85 d6 5f ff ff  movw   $0x0,-0xa02a(%ebp)
8049684:  00 00
8049686:  66 c7 85 c8 5f ff ff  movw   $0x2,-0xa038(%ebp)
804968d:  02 00
804968f:  6a 0c                 push   $0xc
8049691:  ff 70 0c              pushl  0xc(%eax)
8049694:  8b 40 10              mov    0x10(%eax),%eax
8049697:  ff 30                 pushl  (%eax)
8049699:  8d 85 cc 5f ff ff     lea    -0xa034(%ebp),%eax
804969f:  50                    push   %eax
```

```
80496a0:  e8 2b f1 ff ff        call   80487d0 <__memmove_chk@plt>
80496a5:  0f b7 45 0c              movzwl 0xc(%ebp),%eax
80496a9:  66 c1 c8 08              ror    $0x8,%ax
80496ad:  66 89 85 ca 5f ff ff  mov    %ax,-0xa036(%ebp)
80496b4:  83 c4 0c                 add    $0xc,%esp
80496b7:  6a 10                    push   $0x10
80496b9:  56                       push   %esi
80496ba:  ff b5 b0 5f ff ff     pushl  -0xa050(%ebp)
80496c0:  e8 cb f1 ff ff        call   8048890 <connect@plt>
80496c5:  83 c4 10                 add    $0x10,%esp
80496c8:  85 c0                    test   %eax,%eax
80496ca:  0f 88 70 01 00 00        js     8049840 <submitr+0x262>
80496d0:  ba ff ff ff ff        mov    $0xffffffff,%edx
80496d5:  b8 00 00 00 00           mov    $0x0,%eax
80496da:  89 d1                    mov    %edx,%ecx
80496dc:  89 df                    mov    %ebx,%edi
80496de:  f2 ae                    repnz scas %es:(%edi),%al
80496e0:  89 ce                    mov    %ecx,%esi
80496e2:  f7 d6                    not    %esi
80496e4:  89 d1                    mov    %edx,%ecx
80496e6:  8b bd ac 5f ff ff     mov    -0xa054(%ebp),%edi
80496ec:  f2 ae                    repnz scas %es:(%edi),%al
80496ee:  89 8d b4 5f ff ff     mov    %ecx,-0xa04c(%ebp)
80496f4:  89 d1                    mov    %edx,%ecx
80496f6:  8b bd a8 5f ff ff     mov    -0xa058(%ebp),%edi
80496fc:  f2 ae                    repnz scas %es:(%edi),%al
80496fe:  89 cf                    mov    %ecx,%edi
8049700:  f7 d7                    not    %edi
8049702:  89 bd 9c 5f ff ff     mov    %edi,-0xa064(%ebp)
```

```
8049708:  89 d1                   mov    %edx,%ecx
804970a:  8b bd a4 5f ff ff       mov    -0xa05c(%ebp),%edi
8049710:  f2 ae                   repnz scas %es:(%edi),%al
8049712:  8b 95 9c 5f ff ff       mov    -0xa064(%ebp),%edx
8049718:  2b 95 b4 5f ff ff       sub    -0xa04c(%ebp),%edx
804971e:  29 ca                   sub    %ecx,%edx
8049720:  8d 44 76 fd             lea    -0x3(%esi,%esi,2),%eax
8049724:  8d 44 02 7b             lea    0x7b(%edx,%eax,1),%eax
8049728:  3d 00 20 00 00          cmp    $0x2000,%eax
804972d:  0f 87 76 01 00 00       ja     80498a9 <submitr+0x2cb>
8049733:  8d 95 e4 9f ff ff       lea    -0x601c(%ebp),%edx
8049739:  b9 00 08 00 00          mov    $0x800,%ecx
804973e:  b8 00 00 00 00          mov    $0x0,%eax
8049743:  89 d7                   mov    %edx,%edi
8049745:  f3 ab                   rep stos %eax,%es:(%edi)
8049747:  b9 ff ff ff ff          mov    $0xffffffff,%ecx
804974c:  89 df                   mov    %ebx,%edi
804974e:  f2 ae                   repnz scas %es:(%edi),%al
8049750:  89 ca                   mov    %ecx,%edx
8049752:  f7 d2                   not    %edx
8049754:  89 d1                   mov    %edx,%ecx
8049756:  83 e9 01                sub    $0x1,%ecx
8049759:  89 8d b4 5f ff ff       mov    %ecx,-0xa04c(%ebp)
804975f:  0f 84 3b 06 00 00       je     8049da0 <submitr+0x7c2>
8049765:  8d b5 e4 9f ff ff       lea    -0x601c(%ebp),%esi
804976b:  bf 01 00 00 00          mov    $0x1,%edi
8049770:  e9 cb 01 00 00          jmp    8049940 <submitr+0x362>
8049775:  8b 85 a0 5f ff ff       mov    -0xa060(%ebp),%eax
804977b:  c7 00 45 72 72 6f       movl   $0x6f727245,(%eax)
```

```
8049781:  c7 40 04 72 3a 20 43     movl    $0x43203a72,0x4(%eax)
8049788:  c7 40 08 6c 69 65 6e     movl    $0x6e65696c,0x8(%eax)
804978f:  c7 40 0c 74 20 75 6e     movl    $0x6e752074,0xc(%eax)
8049796:  c7 40 10 61 62 6c 65     movl    $0x656c6261,0x10(%eax)
804979d:  c7 40 14 20 74 6f 20     movl    $0x206f7420,0x14(%eax)
80497a4:  c7 40 18 63 72 65 61     movl    $0x61657263,0x18(%eax)
80497ab:  c7 40 1c 74 65 20 73     movl    $0x73206574,0x1c(%eax)
80497b2:  c7 40 20 6f 63 6b 65     movl    $0x656b636f,0x20(%eax)
80497b9:  66 c7 40 24 74 00        movw    $0x74,0x24(%eax)
80497bf:  b8 ff ff ff ff           mov     $0xffffffff,%eax
80497c4:  e9 f2 04 00 00           jmp     8049cbb <submitr+0x6dd>
80497c9:  8b 85 a0 5f ff ff        mov     -0xa060(%ebp),%eax
80497cf:  c7 00 45 72 72 6f        movl    $0x6f727245,(%eax)
80497d5:  c7 40 04 72 3a 20 44     movl    $0x44203a72,0x4(%eax)
80497dc:  c7 40 08 4e 53 20 69     movl    $0x6920534e,0x8(%eax)
80497e3:  c7 40 0c 73 20 75 6e     movl    $0x6e752073,0xc(%eax)
80497ea:  c7 40 10 61 62 6c 65     movl    $0x656c6261,0x10(%eax)
80497f1:  c7 40 14 20 74 6f 20     movl    $0x206f7420,0x14(%eax)
80497f8:  c7 40 18 72 65 73 6f     movl    $0x6f736572,0x18(%eax)
80497ff:  c7 40 1c 6c 76 65 20     movl    $0x2065766c,0x1c(%eax)
8049806:  c7 40 20 73 65 72 76     movl    $0x76726573,0x20(%eax)
804980d:  c7 40 24 65 72 20 61     movl    $0x61207265,0x24(%eax)
8049814:  c7 40 28 64 64 72 65     movl    $0x65726464,0x28(%eax)
804981b:  66 c7 40 2c 73 73        movw    $0x7373,0x2c(%eax)
8049821:  c6 40 2e 00              movb    $0x0,0x2e(%eax)
8049825:  83 ec 0c                 sub     $0xc,%esp
8049828:  ff b5 b0 5f ff ff        pushl   -0xa050(%ebp)
804982e:  e8 6d f0 ff ff           call    80488a0 <close@plt>
8049833:  83 c4 10                 add     $0x10,%esp
```

```
8049836:  b8 ff ff ff ff          mov     $0xffffffff,%eax
804983b:  e9 7b 04 00 00          jmp     8049cbb <submitr+0x6dd>
8049840:  8b 85 a0 5f ff ff       mov     -0xa060(%ebp),%eax
8049846:  c7 00 45 72 72 6f       movl    $0x6f727245,(%eax)
804984c:  c7 40 04 72 3a 20 55    movl    $0x55203a72,0x4(%eax)
8049853:  c7 40 08 6e 61 62 6c    movl    $0x6c62616e,0x8(%eax)
804985a:  c7 40 0c 65 20 74 6f    movl    $0x6f742065,0xc(%eax)
8049861:  c7 40 10 20 63 6f 6e    movl    $0x6e6f6320,0x10(%eax)
8049868:  c7 40 14 6e 65 63 74    movl    $0x7463656e,0x14(%eax)
804986f:  c7 40 18 20 74 6f 20    movl    $0x206f7420,0x18(%eax)
8049876:  c7 40 1c 74 68 65 20    movl    $0x20656874,0x1c(%eax)
804987d:  c7 40 20 73 65 72 76    movl    $0x76726573,0x20(%eax)
8049884:  66 c7 40 24 65 72       movw    $0x7265,0x24(%eax)
804988a:  c6 40 26 00             movb    $0x0,0x26(%eax)
804988e:  83 ec 0c                sub     $0xc,%esp
8049891:  ff b5 b0 5f ff ff       pushl   -0xa050(%ebp)
8049897:  e8 04 f0 ff ff          call    80488a0 <close@plt>
804989c:  83 c4 10                add     $0x10,%esp
804989f:  b8 ff ff ff ff          mov     $0xffffffff,%eax
80498a4:  e9 12 04 00 00          jmp     8049cbb <submitr+0x6dd>
80498a9:  8b 85 a0 5f ff ff       mov     -0xa060(%ebp),%eax
80498af:  c7 00 45 72 72 6f       movl    $0x6f727245,(%eax)
80498b5:  c7 40 04 72 3a 20 52    movl    $0x52203a72,0x4(%eax)
80498bc:  c7 40 08 65 73 75 6c    movl    $0x6c757365,0x8(%eax)
80498c3:  c7 40 0c 74 20 73 74    movl    $0x74732074,0xc(%eax)
80498ca:  c7 40 10 72 69 6e 67    movl    $0x676e6972,0x10(%eax)
80498d1:  c7 40 14 20 74 6f 6f    movl    $0x6f6f7420,0x14(%eax)
80498d8:  c7 40 18 20 6c 61 72    movl    $0x72616c20,0x18(%eax)
80498df:  c7 40 1c 67 65 2e 20    movl    $0x202e6567,0x1c(%eax)
```

```
80498e6:  c7 40 20 49 6e 63 72     movl   $0x72636e49,0x20(%eax)
80498ed:  c7 40 24 65 61 73 65     movl   $0x65736165,0x24(%eax)
80498f4:  c7 40 28 20 53 55 42     movl   $0x42555320,0x28(%eax)
80498fb:  c7 40 2c 4d 49 54 52     movl   $0x5254494d,0x2c(%eax)
8049902:  c7 40 30 5f 4d 41 58     movl   $0x58414d5f,0x30(%eax)
8049909:  c7 40 34 42 55 46 00     movl   $0x465542,0x34(%eax)
8049910:  83 ec 0c                 sub    $0xc,%esp
8049913:  ff b5 b0 5f ff ff        pushl  -0xa050(%ebp)
8049919:  e8 82 ef ff ff           call   80488a0 <close@plt>
804991e:  83 c4 10                 add    $0x10,%esp
8049921:  b8 ff ff ff ff           mov    $0xffffffff,%eax
8049926:  e9 90 03 00 00           jmp    8049cbb <submitr+0x6dd>
804992b:  88 16                    mov    %dl,(%esi)
804992d:  8d 76 01                 lea    0x1(%esi),%esi
8049930:  83 c3 01                 add    $0x1,%ebx
8049933:  83 ad b4 5f ff ff 01     subl   $0x1,-0xa04c(%ebp)
804993a:  0f 84 60 04 00 00        je     8049da0 <submitr+0x7c2>
8049940:  0f b6 13                 movzbl (%ebx),%edx
8049943:  8d 4a d6                 lea    -0x2a(%edx),%ecx
8049946:  89 f8                    mov    %edi,%eax
8049948:  80 f9 0f                 cmp    $0xf,%cl
804994b:  77 0d                    ja     804995a <submitr+0x37c>
804994d:  b8 d9 ff 00 00           mov    $0xffd9,%eax
8049952:  d3 e8                    shr    %cl,%eax
8049954:  83 f0 01                 xor    $0x1,%eax
8049957:  83 e0 01                 and    $0x1,%eax
804995a:  84 c0                    test   %al,%al
804995c:  74 cd                    je     804992b <submitr+0x34d>
804995e:  80 fa 5f                 cmp    $0x5f,%dl
```

```
8049961:  74 c8                  je      804992b <submitr+0x34d>
8049963:  89 d0                  mov     %edx,%eax
8049965:  83 e0 df               and     $0xffffffdf,%eax
8049968:  83 e8 41               sub     $0x41,%eax
804996b:  3c 19                  cmp     $0x19,%al
804996d:  76 bc                  jbe     804992b <submitr+0x34d>
804996f:  80 fa 20               cmp     $0x20,%dl
8049972:  74 54                  je      80499c8 <submitr+0x3ea>
8049974:  8d 42 e0               lea     -0x20(%edx),%eax
8049977:  3c 5f                  cmp     $0x5f,%al
8049979:  76 09                  jbe     8049984 <submitr+0x3a6>
804997b:  80 fa 09               cmp     $0x9,%dl
804997e:  0f 85 d1 03 00 00      jne     8049d55 <submitr+0x777>
8049984:  83 ec 0c               sub     $0xc,%esp
8049987:  0f b6 d2               movzbl %dl,%edx
804998a:  52                     push    %edx
804998b:  68 8c a6 04 08         push    $0x804a68c
8049990:  6a 08                  push    $0x8
8049992:  6a 01                  push    $0x1
8049994:  8d 85 e4 df ff ff      lea     -0x201c(%ebp),%eax
804999a:  50                     push    %eax
804999b:  e8 20 ef ff ff         call    80488c0 <__sprintf_chk@plt>
80499a0:  0f b6 85 e4 df ff ff   movzbl -0x201c(%ebp),%eax
80499a7:  88 06                  mov     %al,(%esi)
80499a9:  0f b6 85 e5 df ff ff   movzbl -0x201b(%ebp),%eax
80499b0:  88 46 01               mov     %al,0x1(%esi)
80499b3:  0f b6 85 e6 df ff ff   movzbl -0x201a(%ebp),%eax
80499ba:  88 46 02               mov     %al,0x2(%esi)
80499bd:  83 c4 20               add     $0x20,%esp
```

```
80499c0:   8d 76 03                lea    0x3(%esi),%esi
80499c3:   e9 68 ff ff ff          jmp    8049930 <submitr+0x352>
80499c8:   c6 06 2b                movb   $0x2b,(%esi)
80499cb:   8d 76 01                lea    0x1(%esi),%esi
80499ce:   e9 5d ff ff ff          jmp    8049930 <submitr+0x352>
80499d3:   01 c6                   add    %eax,%esi
80499d5:   29 c3                   sub    %eax,%ebx
80499d7:   74 27                   je     8049a00 <submitr+0x422>
80499d9:   83 ec 04                sub    $0x4,%esp
80499dc:   53                      push   %ebx
80499dd:   56                      push   %esi
80499de:   57                      push   %edi
80499df:   e8 1c ee ff ff          call   8048800 <write@plt>
80499e4:   83 c4 10                add    $0x10,%esp
80499e7:   85 c0                   test   %eax,%eax
80499e9:   7f e8                   jg     80499d3 <submitr+0x3f5>
80499eb:   e8 40 ee ff ff          call   8048830 <__errno_location@plt>
80499f0:   83 38 04                cmpl   $0x4,(%eax)
80499f3:   0f 85 41 01 00 00       jne    8049b3a <submitr+0x55c>
80499f9:   b8 00 00 00 00          mov    $0x0,%eax
80499fe:   eb d3                   jmp    80499d3 <submitr+0x3f5>
8049a00:   8b bd b4 5f ff ff       mov    -0xa04c(%ebp),%edi
8049a06:   85 ff                   test   %edi,%edi
8049a08:   0f 88 2c 01 00 00       js     8049b3a <submitr+0x55c>
8049a0e:   8b 85 b0 5f ff ff       mov    -0xa050(%ebp),%eax
8049a14:   89 85 d8 5f ff ff       mov    %eax,-0xa028(%ebp)
8049a1a:   c7 85 dc 5f ff ff 00    movl   $0x0,-0xa024(%ebp)
8049a21:   00 00 00
8049a24:   8d 85 e4 5f ff ff       lea    -0xa01c(%ebp),%eax
```

```
8049a2a:  89 85 e0 5f ff ff     mov     %eax,-0xa020(%ebp)
8049a30:  b9 00 20 00 00        mov     $0x2000,%ecx
8049a35:  8d 95 e4 7f ff ff     lea     -0x801c(%ebp),%edx
8049a3b:  8d 85 d8 5f ff ff     lea     -0xa028(%ebp),%eax
8049a41:  e8 da fa ff ff        call    8049520 <rio_readlineb>
8049a46:  85 c0                 test    %eax,%eax
8049a48:  0f 8e 59 01 00 00     jle     8049ba7 <submitr+0x5c9>
8049a4e:  83 ec 0c              sub     $0xc,%esp
8049a51:  8d 85 e4 df ff ff     lea     -0x201c(%ebp),%eax
8049a57:  50                    push    %eax
8049a58:  8d 85 c4 5f ff ff     lea     -0xa03c(%ebp),%eax
8049a5e:  50                    push    %eax
8049a5f:  8d 85 e4 bf ff ff     lea     -0x401c(%ebp),%eax
8049a65:  50                    push    %eax
8049a66:  68 93 a6 04 08        push    $0x804a693
8049a6b:  8d 85 e4 7f ff ff     lea     -0x801c(%ebp),%eax
8049a71:  50                    push    %eax
8049a72:  e8 99 ed ff ff        call    8048810 <__isoc99_sscanf@plt>
8049a77:  8b 85 c4 5f ff ff     mov     -0xa03c(%ebp),%eax
8049a7d:  83 c4 20              add     $0x20,%esp
8049a80:  3d c8 00 00 00        cmp     $0xc8,%eax
8049a85:  0f 85 9d 01 00 00     jne     8049c28 <submitr+0x64a>
8049a8b:  8d 9d e4 7f ff ff     lea     -0x801c(%ebp),%ebx
8049a91:  bf a4 a6 04 08        mov     $0x804a6a4,%edi
8049a96:  b9 03 00 00 00        mov     $0x3,%ecx
8049a9b:  89 de                 mov     %ebx,%esi
8049a9d:  f3 a6                 repz cmpsb %es:(%edi),%ds:(%esi)
8049a9f:  0f 97 c0              seta    %al
8049aa2:  1c 00                 sbb     $0x0,%al
```

```
8049aa4:  84 c0                    test   %al,%al
8049aa6:  0f 84 b3 01 00 00        je     8049c5f <submitr+0x681>
8049aac:  b9 00 20 00 00           mov    $0x2000,%ecx
8049ab1:  89 da                    mov    %ebx,%edx
8049ab3:  8d 85 d8 5f ff ff        lea    -0xa028(%ebp),%eax
8049ab9:  e8 62 fa ff ff           call   8049520 <rio_readlineb>
8049abe:  85 c0                    test   %eax,%eax
8049ac0:  7f cf                    jg     8049a91 <submitr+0x4b3>
8049ac2:  8b 85 a0 5f ff ff        mov    -0xa060(%ebp),%eax
8049ac8:  c7 00 45 72 72 6f        movl   $0x6f727245,(%eax)
8049ace:  c7 40 04 72 3a 20 43     movl   $0x43203a72,0x4(%eax)
8049ad5:  c7 40 08 6c 69 65 6e     movl   $0x6e65696c,0x8(%eax)
8049adc:  c7 40 0c 74 20 75 6e     movl   $0x6e752074,0xc(%eax)
8049ae3:  c7 40 10 61 62 6c 65     movl   $0x656c6261,0x10(%eax)
8049aea:  c7 40 14 20 74 6f 20     movl   $0x206f7420,0x14(%eax)
8049af1:  c7 40 18 72 65 61 64     movl   $0x64616572,0x18(%eax)
8049af8:  c7 40 1c 20 68 65 61     movl   $0x61656820,0x1c(%eax)
8049aff:  c7 40 20 64 65 72 73     movl   $0x73726564,0x20(%eax)
8049b06:  c7 40 24 20 66 72 6f     movl   $0x6f726620,0x24(%eax)
8049b0d:  c7 40 28 6d 20 73 65     movl   $0x6573206d,0x28(%eax)
8049b14:  c7 40 2c 72 76 65 72     movl   $0x72657672,0x2c(%eax)
8049b1b:  c6 40 30 00              movb   $0x0,0x30(%eax)
8049b1f:  83 ec 0c                 sub    $0xc,%esp
8049b22:  ff b5 b0 5f ff ff        pushl  -0xa050(%ebp)
8049b28:  e8 73 ed ff ff           call   80488a0 <close@plt>
8049b2d:  83 c4 10                 add    $0x10,%esp
8049b30:  b8 ff ff ff ff           mov    $0xffffffff,%eax
8049b35:  e9 81 01 00 00           jmp    8049cbb <submitr+0x6dd>
8049b3a:  8b 85 a0 5f ff ff        mov    -0xa060(%ebp),%eax
```

```
8049b40:  c7 00 45 72 72 6f        movl   $0x6f727245,(%eax)
8049b46:  c7 40 04 72 3a 20 43     movl   $0x43203a72,0x4(%eax)
8049b4d:  c7 40 08 6c 69 65 6e     movl   $0x6e65696c,0x8(%eax)
8049b54:  c7 40 0c 74 20 75 6e     movl   $0x6e752074,0xc(%eax)
8049b5b:  c7 40 10 61 62 6c 65     movl   $0x656c6261,0x10(%eax)
8049b62:  c7 40 14 20 74 6f 20     movl   $0x206f7420,0x14(%eax)
8049b69:  c7 40 18 77 72 69 74     movl   $0x74697277,0x18(%eax)
8049b70:  c7 40 1c 65 20 74 6f     movl   $0x6f742065,0x1c(%eax)
8049b77:  c7 40 20 20 74 68 65     movl   $0x65687420,0x20(%eax)
8049b7e:  c7 40 24 20 73 65 72     movl   $0x72657320,0x24(%eax)
8049b85:  c7 40 28 76 65 72 00     movl   $0x726576,0x28(%eax)
8049b8c:  83 ec 0c                 sub    $0xc,%esp
8049b8f:  ff b5 b0 5f ff ff        pushl  -0xa050(%ebp)
8049b95:  e8 06 ed ff ff           call   80488a0 <close@plt>
8049b9a:  83 c4 10                 add    $0x10,%esp
8049b9d:  b8 ff ff ff ff           mov    $0xffffffff,%eax
8049ba2:  e9 14 01 00 00           jmp    8049cbb <submitr+0x6dd>
8049ba7:  8b 85 a0 5f ff ff        mov    -0xa060(%ebp),%eax
8049bad:  c7 00 45 72 72 6f        movl   $0x6f727245,(%eax)
8049bb3:  c7 40 04 72 3a 20 43     movl   $0x43203a72,0x4(%eax)
8049bba:  c7 40 08 6c 69 65 6e     movl   $0x6e65696c,0x8(%eax)
8049bc1:  c7 40 0c 74 20 75 6e     movl   $0x6e752074,0xc(%eax)
8049bc8:  c7 40 10 61 62 6c 65     movl   $0x656c6261,0x10(%eax)
8049bcf:  c7 40 14 20 74 6f 20     movl   $0x206f7420,0x14(%eax)
8049bd6:  c7 40 18 72 65 61 64     movl   $0x64616572,0x18(%eax)
8049bdd:  c7 40 1c 20 66 69 72     movl   $0x72696620,0x1c(%eax)
8049be4:  c7 40 20 73 74 20 68     movl   $0x68207473,0x20(%eax)
8049beb:  c7 40 24 65 61 64 65     movl   $0x65646165,0x24(%eax)
8049bf2:  c7 40 28 72 20 66 72     movl   $0x72662072,0x28(%eax)
```

```
8049bf9:   c7 40 2c 6f 6d 20 73      movl   $0x73206d6f,0x2c(%eax)
8049c00:   c7 40 30 65 72 76 65      movl   $0x65767265,0x30(%eax)
8049c07:   66 c7 40 34 72 00         movw   $0x72,0x34(%eax)
8049c0d:   83 ec 0c                  sub    $0xc,%esp
8049c10:   ff b5 b0 5f ff ff         pushl  -0xa050(%ebp)
8049c16:   e8 85 ec ff ff            call   80488a0 <close@plt>
8049c1b:   83 c4 10                  add    $0x10,%esp
8049c1e:   b8 ff ff ff ff            mov    $0xffffffff,%eax
8049c23:   e9 93 00 00 00            jmp    8049cbb <submitr+0x6dd>
8049c28:   83 ec 08                  sub    $0x8,%esp
8049c2b:   8d 95 e4 df ff ff         lea    -0x201c(%ebp),%edx
8049c31:   52                        push   %edx
8049c32:   50                        push   %eax
8049c33:   68 a4 a5 04 08            push   $0x804a5a4
8049c38:   6a ff                     push   $0xffffffff
8049c3a:   6a 01                     push   $0x1
8049c3c:   ff b5 a0 5f ff ff         pushl  -0xa060(%ebp)
8049c42:   e8 79 ec ff ff            call   80488c0 <__sprintf_chk@plt>
8049c47:   83 c4 14                  add    $0x14,%esp
8049c4a:   ff b5 b0 5f ff ff         pushl  -0xa050(%ebp)
8049c50:   e8 4b ec ff ff            call   80488a0 <close@plt>
8049c55:   83 c4 10                  add    $0x10,%esp
8049c58:   b8 ff ff ff ff            mov    $0xffffffff,%eax
8049c5d:   eb 5c                     jmp    8049cbb <submitr+0x6dd>
8049c5f:   b9 00 20 00 00            mov    $0x2000,%ecx
8049c64:   8d 95 e4 7f ff ff         lea    -0x801c(%ebp),%edx
8049c6a:   8d 85 d8 5f ff ff         lea    -0xa028(%ebp),%eax
8049c70:   e8 ab f8 ff ff            call   8049520 <rio_readlineb>
8049c75:   85 c0                     test   %eax,%eax
```

```
8049c77:   7e 5a                    jle     8049cd3 <submitr+0x6f5>
8049c79:   83 ec 08                 sub     $0x8,%esp
8049c7c:   8d 85 e4 7f ff ff        lea     -0x801c(%ebp),%eax
8049c82:   50                       push    %eax
8049c83:   8b b5 a0 5f ff ff        mov     -0xa060(%ebp),%esi
8049c89:   56                       push    %esi
8049c8a:   e8 11 eb ff ff           call    80487a0 <strcpy@plt>
8049c8f:   83 c4 04                 add     $0x4,%esp
8049c92:   ff b5 b0 5f ff ff        pushl   -0xa050(%ebp)
8049c98:   e8 03 ec ff ff           call    80488a0 <close@plt>
8049c9d:   bf a7 a6 04 08           mov     $0x804a6a7,%edi
8049ca2:   b9 03 00 00 00           mov     $0x3,%ecx
8049ca7:   f3 a6                    repz cmpsb %es:(%edi),%ds:(%esi)
8049ca9:   0f 97 c0                 seta    %al
8049cac:   1c 00                    sbb     $0x0,%al
8049cae:   83 c4 10                 add     $0x10,%esp
8049cb1:   84 c0                    test    %al,%al
8049cb3:   0f 95 c0                 setne   %al
8049cb6:   0f b6 c0                 movzbl %al,%eax
8049cb9:   f7 d8                    neg     %eax
8049cbb:   8b 7d e4                 mov     -0x1c(%ebp),%edi
8049cbe:   65 33 3d 14 00 00 00     xor     %gs:0x14,%edi
8049cc5:   0f 85 3d 01 00 00        jne     8049e08 <submitr+0x82a>
8049ccb:   8d 65 f4                 lea     -0xc(%ebp),%esp
8049cce:   5b                       pop     %ebx
8049ccf:   5e                       pop     %esi
8049cd0:   5f                       pop     %edi
8049cd1:   5d                       pop     %ebp
8049cd2:   c3                       ret
```

```
8049cd3:   8b 85 a0 5f ff ff       mov      -0xa060(%ebp),%eax
8049cd9:   c7 00 45 72 72 6f          movl    $0x6f727245,(%eax)
8049cdf:   c7 40 04 72 3a 20 43    movl    $0x43203a72,0x4(%eax)
8049ce6:   c7 40 08 6c 69 65 6e    movl    $0x6e65696c,0x8(%eax)
8049ced:   c7 40 0c 74 20 75 6e    movl    $0x6e752074,0xc(%eax)
8049cf4:   c7 40 10 61 62 6c 65    movl    $0x656c6261,0x10(%eax)
8049cfb:   c7 40 14 20 74 6f 20    movl    $0x206f7420,0x14(%eax)
8049d02:   c7 40 18 72 65 61 64    movl    $0x64616572,0x18(%eax)
8049d09:   c7 40 1c 20 73 74 61    movl    $0x61747320,0x1c(%eax)
8049d10:   c7 40 20 74 75 73 20    movl    $0x20737574,0x20(%eax)
8049d17:   c7 40 24 6d 65 73 73    movl    $0x7373656d,0x24(%eax)
8049d1e:   c7 40 28 61 67 65 20    movl    $0x20656761,0x28(%eax)
8049d25:   c7 40 2c 66 72 6f 6d    movl    $0x6d6f7266,0x2c(%eax)
8049d2c:   c7 40 30 20 73 65 72    movl    $0x72657320,0x30(%eax)
8049d33:   c7 40 34 76 65 72 00    movl    $0x726576,0x34(%eax)
8049d3a:   83 ec 0c                   sub      $0xc,%esp
8049d3d:   ff b5 b0 5f ff ff       pushl   -0xa050(%ebp)
8049d43:   e8 58 eb ff ff          call     80488a0 <close@plt>
8049d48:   83 c4 10                   add      $0x10,%esp
8049d4b:   b8 ff ff ff ff          mov      $0xffffffff,%eax
8049d50:   e9 66 ff ff ff          jmp      8049cbb <submitr+0x6dd>
8049d55:   a1 d4 a5 04 08             mov      0x804a5d4,%eax
8049d5a:   8b bd a0 5f ff ff       mov      -0xa060(%ebp),%edi
8049d60:   89 07                      mov      %eax,(%edi)
8049d62:   a1 13 a6 04 08             mov      0x804a613,%eax
8049d67:   89 47 3f                   mov      %eax,0x3f(%edi)
8049d6a:   89 f8                      mov      %edi,%eax
8049d6c:   8d 7f 04                   lea      0x4(%edi),%edi
8049d6f:   83 e7 fc                   and      $0xfffffffc,%edi
```

```
8049d72:  29 f8                   sub    %edi,%eax
8049d74:  be d4 a5 04 08          mov    $0x804a5d4,%esi
8049d79:  29 c6                   sub    %eax,%esi
8049d7b:  83 c0 43                add    $0x43,%eax
8049d7e:  c1 e8 02                shr    $0x2,%eax
8049d81:  89 c1                   mov    %eax,%ecx
8049d83:  f3 a5                   rep movsl %ds:(%esi),%es:(%edi)
8049d85:  83 ec 0c                sub    $0xc,%esp
8049d88:  ff b5 b0 5f ff ff       pushl  -0xa050(%ebp)
8049d8e:  e8 0d eb ff ff          call   80488a0 <close@plt>
8049d93:  83 c4 10                add    $0x10,%esp
8049d96:  b8 ff ff ff ff          mov    $0xffffffff,%eax
8049d9b:  e9 1b ff ff ff          jmp    8049cbb <submitr+0x6dd>
8049da0:  8d 85 e4 9f ff ff       lea    -0x601c(%ebp),%eax
8049da6:  50                      push   %eax
8049da7:  ff b5 a4 5f ff ff       pushl  -0xa05c(%ebp)
8049dad:  ff b5 a8 5f ff ff       pushl  -0xa058(%ebp)
8049db3:  ff b5 ac 5f ff ff       pushl  -0xa054(%ebp)
8049db9:  68 18 a6 04 08          push   $0x804a618
8049dbe:  68 00 20 00 00          push   $0x2000
8049dc3:  6a 01                   push   $0x1
8049dc5:  8d bd e4 7f ff ff       lea    -0x801c(%ebp),%edi
8049dcb:  57                      push   %edi
8049dcc:  e8 ef ea ff ff          call   80488c0 <__sprintf_chk@plt>
8049dd1:  b9 ff ff ff ff          mov    $0xffffffff,%ecx
8049dd6:  b8 00 00 00 00          mov    $0x0,%eax
8049ddb:  f2 ae                   repnz scas %es:(%edi),%al
8049ddd:  89 cb                   mov    %ecx,%ebx
8049ddf:  f7 d3                   not    %ebx
```

```
 8049de1:   8d 7b ff                lea      -0x1(%ebx),%edi
 8049de4:   83 c4 20                add      $0x20,%esp
 8049de7:   89 fb                   mov      %edi,%ebx
 8049de9:   8d b5 e4 7f ff ff       lea      -0x801c(%ebp),%esi
 8049def:   85 ff                   test     %edi,%edi
 8049df1:   0f 84 17 fc ff ff       je       8049a0e <submitr+0x430>
 8049df7:   89 bd b4 5f ff ff       mov      %edi,-0xa04c(%ebp)
 8049dfd:   8b bd b0 5f ff ff       mov      -0xa050(%ebp),%edi
 8049e03:   e9 d1 fb ff ff          jmp      80499d9 <submitr+0x3fb>
 8049e08:   e8 83 e9 ff ff          call     8048790 <__stack_chk_fail@plt>

08049e0d <init_timeout>:
 8049e0d:   55                      push     %ebp
 8049e0e:   89 e5                   mov      %esp,%ebp
 8049e10:   53                      push     %ebx
 8049e11:   83 ec 04                sub      $0x4,%esp
 8049e14:   8b 5d 08                mov      0x8(%ebp),%ebx
 8049e17:   85 db                   test     %ebx,%ebx
 8049e19:   74 24                   je       8049e3f <init_timeout+0x32>
 8049e1b:   83 ec 08                sub      $0x8,%esp
 8049e1e:   68 fa 94 04 08          push     $0x80494fa
 8049e23:   6a 0e                   push     $0xe
 8049e25:   e8 36 e9 ff ff          call     8048760 <signal@plt>
 8049e2a:   85 db                   test     %ebx,%ebx
 8049e2c:   b8 00 00 00 00          mov      $0x0,%eax
 8049e31:   0f 48 d8                cmovs    %eax,%ebx
 8049e34:   89 1c 24                mov      %ebx,(%esp)
 8049e37:   e8 44 e9 ff ff          call     8048780 <alarm@plt>
 8049e3c:   83 c4 10                add      $0x10,%esp
```

```
 8049e3f:   8b 5d fc                mov     -0x4(%ebp),%ebx
 8049e42:   c9                      leave
 8049e43:   c3                      ret

08049e44 <init_driver>:
 8049e44:   55                      push    %ebp
 8049e45:   89 e5                   mov     %esp,%ebp
 8049e47:   57                      push    %edi
 8049e48:   56                      push    %esi
 8049e49:   53                      push    %ebx
 8049e4a:   83 ec 34                sub     $0x34,%esp
 8049e4d:   8b 75 08                mov     0x8(%ebp),%esi
 8049e50:   65 a1 14 00 00 00       mov     %gs:0x14,%eax
 8049e56:   89 45 e4                mov     %eax,-0x1c(%ebp)
 8049e59:   31 c0                   xor     %eax,%eax
 8049e5b:   6a 01                   push    $0x1
 8049e5d:   6a 0d                   push    $0xd
 8049e5f:   e8 fc e8 ff ff         call    8048760 <signal@plt>
 8049e64:   83 c4 08                add     $0x8,%esp
 8049e67:   6a 01                   push    $0x1
 8049e69:   6a 1d                   push    $0x1d
 8049e6b:   e8 f0 e8 ff ff         call    8048760 <signal@plt>
 8049e70:   83 c4 08                add     $0x8,%esp
 8049e73:   6a 01                   push    $0x1
 8049e75:   6a 1d                   push    $0x1d
 8049e77:   e8 e4 e8 ff ff         call    8048760 <signal@plt>
 8049e7c:   83 c4 0c                add     $0xc,%esp
 8049e7f:   6a 00                   push    $0x0
 8049e81:   6a 01                   push    $0x1
```

```
8049e83:   6a 02                    push   $0x2
8049e85:   e8 c6 e9 ff ff           call   8048850 <socket@plt>
8049e8a:   83 c4 10                 add    $0x10,%esp
8049e8d:   85 c0                    test   %eax,%eax
8049e8f:   0f 88 a0 00 00 00        js     8049f35 <init_driver+0xf1>
8049e95:   89 c3                    mov    %eax,%ebx
8049e97:   83 ec 0c                 sub    $0xc,%esp
8049e9a:   68 aa a6 04 08           push   $0x804a6aa
8049e9f:   e8 cc e9 ff ff           call   8048870 <gethostbyname@plt>
8049ea4:   83 c4 10                 add    $0x10,%esp
8049ea7:   85 c0                    test   %eax,%eax
8049ea9:   0f 84 d1 00 00 00        je     8049f80 <init_driver+0x13c>
8049eaf:   8d 7d d4                 lea    -0x2c(%ebp),%edi
8049eb2:   c7 45 d6 00 00 00 00     movl   $0x0,-0x2a(%ebp)
8049eb9:   c7 45 da 00 00 00 00     movl   $0x0,-0x26(%ebp)
8049ec0:   c7 45 de 00 00 00 00     movl   $0x0,-0x22(%ebp)
8049ec7:   66 c7 45 e2 00 00        movw   $0x0,-0x1e(%ebp)
8049ecd:   66 c7 45 d4 02 00        movw   $0x2,-0x2c(%ebp)
8049ed3:   6a 0c                    push   $0xc
8049ed5:   ff 70 0c                 pushl  0xc(%eax)
8049ed8:   8b 40 10                 mov    0x10(%eax),%eax
8049edb:   ff 30                    pushl  (%eax)
8049edd:   8d 45 d8                 lea    -0x28(%ebp),%eax
8049ee0:   50                       push   %eax
8049ee1:   e8 ea e8 ff ff           call   80487d0 <__memmove_chk@plt>
8049ee6:   66 c7 45 d6 22 b9        movw   $0xb922,-0x2a(%ebp)
8049eec:   83 c4 0c                 add    $0xc,%esp
8049eef:   6a 10                    push   $0x10
8049ef1:   57                       push   %edi
```

```
8049ef2:    53                      push    %ebx
8049ef3:    e8 98 e9 ff ff          call    8048890 <connect@plt>
8049ef8:    83 c4 10                add     $0x10,%esp
8049efb:    85 c0                   test    %eax,%eax
8049efd:    0f 88 e9 00 00 00       js      8049fec <init_driver+0x1a8>
8049f03:    83 ec 0c                sub     $0xc,%esp
8049f06:    53                      push    %ebx
8049f07:    e8 94 e9 ff ff          call    80488a0 <close@plt>
8049f0c:    66 c7 06 4f 4b          movw    $0x4b4f,(%esi)
8049f11:    c6 46 02 00             movb    $0x0,0x2(%esi)
8049f15:    83 c4 10                add     $0x10,%esp
8049f18:    b8 00 00 00 00          mov     $0x0,%eax
8049f1d:    8b 55 e4                mov     -0x1c(%ebp),%edx
8049f20:    65 33 15 14 00 00 00    xor     %gs:0x14,%edx
8049f27:    0f 85 ec 00 00 00       jne     804a019 <init_driver+0x1d5>
8049f2d:    8d 65 f4                lea     -0xc(%ebp),%esp
8049f30:    5b                      pop     %ebx
8049f31:    5e                      pop     %esi
8049f32:    5f                      pop     %edi
8049f33:    5d                      pop     %ebp
8049f34:    c3                      ret
8049f35:    c7 06 45 72 72 6f       movl    $0x6f727245,(%esi)
8049f3b:    c7 46 04 72 3a 20 43    movl    $0x43203a72,0x4(%esi)
8049f42:    c7 46 08 6c 69 65 6e    movl    $0x6e65696c,0x8(%esi)
8049f49:    c7 46 0c 74 20 75 6e    movl    $0x6e752074,0xc(%esi)
8049f50:    c7 46 10 61 62 6c 65    movl    $0x656c6261,0x10(%esi)
8049f57:    c7 46 14 20 74 6f 20    movl    $0x206f7420,0x14(%esi)
8049f5e:    c7 46 18 63 72 65 61    movl    $0x61657263,0x18(%esi)
8049f65:    c7 46 1c 74 65 20 73    movl    $0x73206574,0x1c(%esi)
```

```
8049f6c:   c7 46 20 6f 63 6b 65      movl   $0x656b636f,0x20(%esi)
8049f73:   66 c7 46 24 74 00         movw   $0x74,0x24(%esi)
8049f79:   b8 ff ff ff ff            mov    $0xffffffff,%eax
8049f7e:   eb 9d                     jmp    8049f1d <init_driver+0xd9>
8049f80:   c7 06 45 72 72 6f         movl   $0x6f727245,(%esi)
8049f86:   c7 46 04 72 3a 20 44      movl   $0x44203a72,0x4(%esi)
8049f8d:   c7 46 08 4e 53 20 69      movl   $0x6920534e,0x8(%esi)
8049f94:   c7 46 0c 73 20 75 6e      movl   $0x6e752073,0xc(%esi)
8049f9b:   c7 46 10 61 62 6c 65      movl   $0x656c6261,0x10(%esi)
8049fa2:   c7 46 14 20 74 6f 20      movl   $0x206f7420,0x14(%esi)
8049fa9:   c7 46 18 72 65 73 6f      movl   $0x6f736572,0x18(%esi)
8049fb0:   c7 46 1c 6c 76 65 20      movl   $0x2065766c,0x1c(%esi)
8049fb7:   c7 46 20 73 65 72 76      movl   $0x76726573,0x20(%esi)
8049fbe:   c7 46 24 65 72 20 61      movl   $0x61207265,0x24(%esi)
8049fc5:   c7 46 28 64 64 72 65      movl   $0x65726464,0x28(%esi)
8049fcc:   66 c7 46 2c 73 73         movw   $0x7373,0x2c(%esi)
8049fd2:   c6 46 2e 00               movb   $0x0,0x2e(%esi)
8049fd6:   83 ec 0c                  sub    $0xc,%esp
8049fd9:   53                        push   %ebx
8049fda:   e8 c1 e8 ff ff            call   80488a0 <close@plt>
8049fdf:   83 c4 10                  add    $0x10,%esp
8049fe2:   b8 ff ff ff ff            mov    $0xffffffff,%eax
8049fe7:   e9 31 ff ff ff            jmp    8049f1d <init_driver+0xd9>
8049fec:   83 ec 0c                  sub    $0xc,%esp
8049fef:   68 aa a6 04 08            push   $0x804a6aa
8049ff4:   68 64 a6 04 08            push   $0x804a664
8049ff9:   6a ff                     push   $0xffffffff
8049ffb:   6a 01                     push   $0x1
8049ffd:   56                        push   %esi
```

```
 8049ffe:   e8 bd e8 ff ff        call    80488c0 <__sprintf_chk@plt>
 804a003:   83 c4 14                  add     $0x14,%esp
 804a006:   53                        push    %ebx
 804a007:   e8 94 e8 ff ff        call    80488a0 <close@plt>
 804a00c:   83 c4 10                  add     $0x10,%esp
 804a00f:   b8 ff ff ff ff        mov     $0xffffffff,%eax
 804a014:   e9 04 ff ff ff        jmp     8049f1d <init_driver+0xd9>
 804a019:   e8 72 e7 ff ff        call    8048790 <__stack_chk_fail@plt>

0804a01e <driver_post>:
 804a01e:   55                        push    %ebp
 804a01f:   89 e5                     mov     %esp,%ebp
 804a021:   53                        push    %ebx
 804a022:   83 ec 04                  sub     $0x4,%esp
 804a025:   8b 55 08                  mov     0x8(%ebp),%edx
 804a028:   8b 45 10                  mov     0x10(%ebp),%eax
 804a02b:   8b 5d 14                  mov     0x14(%ebp),%ebx
 804a02e:   85 c0                     test    %eax,%eax
 804a030:   75 17                     jne     804a049 <driver_post+0x2b>
 804a032:   85 d2                     test    %edx,%edx
 804a034:   74 05                     je      804a03b <driver_post+0x1d>
 804a036:   80 3a 00                  cmpb    $0x0,(%edx)
 804a039:   75 33                     jne     804a06e <driver_post+0x50>
 804a03b:   66 c7 03 4f 4b            movw    $0x4b4f,(%ebx)
 804a040:   c6 43 02 00               movb    $0x0,0x2(%ebx)
 804a044:   8b 5d fc                  mov     -0x4(%ebp),%ebx
 804a047:   c9                        leave
 804a048:   c3                        ret
 804a049:   83 ec 04                  sub     $0x4,%esp
```

```
 804a04c:   ff 75 0c                 pushl  0xc(%ebp)
 804a04f:   68 b9 a6 04 08           push   $0x804a6b9
 804a054:   6a 01                    push   $0x1
 804a056:   e8 e5 e7 ff ff           call   8048840 <__printf_chk@plt>
 804a05b:   66 c7 03 4f 4b           movw   $0x4b4f,(%ebx)
 804a060:   c6 43 02 00              movb   $0x0,0x2(%ebx)
 804a064:   83 c4 10                 add    $0x10,%esp
 804a067:   b8 00 00 00 00           mov    $0x0,%eax
 804a06c:   eb d6                    jmp    804a044 <driver_post+0x26>
 804a06e:   83 ec 04                 sub    $0x4,%esp
 804a071:   53                       push   %ebx
 804a072:   ff 75 0c                 pushl  0xc(%ebp)
 804a075:   68 d0 a6 04 08           push   $0x804a6d0
 804a07a:   52                       push   %edx
 804a07b:   68 e7 a6 04 08           push   $0x804a6e7
 804a080:   68 b9 22 00 00           push   $0x22b9
 804a085:   68 aa a6 04 08           push   $0x804a6aa
 804a08a:   e8 4f f5 ff ff           call   80495de <submitr>
 804a08f:   83 c4 20                 add    $0x20,%esp
 804a092:   eb b0                    jmp    804a044 <driver_post+0x26>
 804a094:   66 90                    xchg   %ax,%ax
 804a096:   66 90                    xchg   %ax,%ax
 804a098:   66 90                    xchg   %ax,%ax
 804a09a:   66 90                    xchg   %ax,%ax
 804a09c:   66 90                    xchg   %ax,%ax
 804a09e:   66 90                    xchg   %ax,%ax

0804a0a0 <__libc_csu_init>:
 804a0a0:   55                       push   %ebp
```

```
804a0a1:  57                    push   %edi
804a0a2:  56                    push   %esi
804a0a3:  53                    push   %ebx
804a0a4:  e8 87 e8 ff ff        call   8048930 <__x86.get_pc_thunk.bx>
804a0a9:  81 c3 57 1f 00 00     add    $0x1f57,%ebx
804a0af:  83 ec 0c              sub    $0xc,%esp
804a0b2:  8b 6c 24 28           mov    0x28(%esp),%ebp
804a0b6:  8d b3 10 ff ff ff     lea    -0xf0(%ebx),%esi
804a0bc:  e8 33 e6 ff ff        call   80486f4 <_init>
804a0c1:  8d 83 0c ff ff ff     lea    -0xf4(%ebx),%eax
804a0c7:  29 c6                 sub    %eax,%esi
804a0c9:  c1 fe 02              sar    $0x2,%esi
804a0cc:  85 f6                 test   %esi,%esi
804a0ce:  74 25                 je     804a0f5 <__libc_csu_init+0x55>
804a0d0:  31 ff                 xor    %edi,%edi
804a0d2:  8d b6 00 00 00 00     lea    0x0(%esi),%esi
804a0d8:  83 ec 04              sub    $0x4,%esp
804a0db:  55                    push   %ebp
804a0dc:  ff 74 24 2c           pushl  0x2c(%esp)
804a0e0:  ff 74 24 2c           pushl  0x2c(%esp)
804a0e4:  ff 94 bb 0c ff ff ff  call   *-0xf4(%ebx,%edi,4)
804a0eb:  83 c7 01              add    $0x1,%edi
804a0ee:  83 c4 10              add    $0x10,%esp
804a0f1:  39 fe                 cmp    %edi,%esi
804a0f3:  75 e3                 jne    804a0d8 <__libc_csu_init+0x38>
804a0f5:  83 c4 0c              add    $0xc,%esp
804a0f8:  5b                    pop    %ebx
804a0f9:  5e                    pop    %esi
804a0fa:  5f                    pop    %edi
```

```
 804a0fb:   5d                      pop    %ebp
 804a0fc:   c3                      ret
 804a0fd:   8d 76 00                lea    0x0(%esi),%esi

0804a100 <__libc_csu_fini>:
 804a100:   f3 c3                   repz ret

Disassembly of section .fini:

0804a104 <_fini>:
 804a104:   53                      push   %ebx
 804a105:   83 ec 08                sub    $0x8,%esp
 804a108:   e8 23 e8 ff ff          call   8048930 <__x86.get_pc_thunk.bx>
 804a10d:   81 c3 f3 1e 00 00       add    $0x1ef3,%ebx
 804a113:   83 c4 08                add    $0x8,%esp
 804a116:   5b                      pop    %ebx
 804a117:   c3                      ret
```