《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 ：	张洪宾
学　　号	18340208
时　　间 ：	2019　年　10　月　1　日

成 绩 ：

实验一 ： X86汇编基础二进制炸弹

一. 实验目的

1. 熟悉汇编语言的基本操作以及反汇编
2. 熟悉gdb调试工具

二. 实验内容

通过Linux上一个可执行的32位的bomb文件，将所有关卡的密码找到，输入密码完成拆弹。若某一关密码错误，会调用explode_bomb函数，直接结束进程。所以必须找到准确的密码精确地拆弹。而为了从执行文件得到汇编的语句，我们需要用gdb的

三. 实验器材

PC机一台，装有Linux操作系统的虚拟机一套。

对于Ubuntu64位，无法执行32位地可执行文件，可输入以下地命令行：

```
1.  sudo apt-get install lib32ncurses5
2.  sudo apt-get install lib32z1
```

这样子Ubuntu64位操作系统就可以执行32位的bomb文件了。

四. 实验过程与结果

首先我下载了我的学号18340208对应的炸弹bomb11，打开压缩包找到bomb可执行文件。用gdb调试工具提供的反汇编功能将可执行文件的汇编代码，

```
3.  ~/Desktop/bomb11$ objdump -d ./bomb > dump.txt
```

然后就可以通过sublime打开dump.txt，用CTRL+F进行对函数的定位。

4.1 第一关

通过汇编代码可以看出调用了<strings_not_equal>函数，我就猜它是比较字符串，然后就根据这么久的编程经验猜出它是要比较输入的字符串与给定字符串是否相同，联想C语言中的strcmp函数返回int型变量，我大胆猜测，这个<strings_not_equal>函数也是返回

一个整数来表达两个字符串之间的关系。然后我再看了一下<strings_not_equal>函数周围的语句（除去压栈操作之外的代码）。如图，首先%eax常常用来表示函数的返回值，然后又百度得到test函数是对两个值进行与操作，而jne在 ZF = 0即两个字符串不相等的时候跳转到爆炸函数，于是就很明显了，当输入的字符串与内存中的字符串不相等是爆炸。

```
0x08048b5a <+0>:     push   %ebp
0x08048b5b <+1>:     mov    %esp,%ebp
0x08048b5d <+3>:     sub    $0x10,%esp
0x08048b60 <+6>:     push   $0x804a244
0x08048b65 <+11>:    pushl  0x8(%ebp)
0x08048b68 <+14>:    call   0x804905d <strings_not_equal>
0x08048b6d <+19>:    add    $0x10,%esp
0x08048b70 <+22>:    test   %eax,%eax
0x08048b72 <+24>:    jne    0x8048b76 <phase_1+28>
0x08048b74 <+26>:    leave
0x08048b75 <+27>:    ret
0x08048b76 <+28>:    call   0x8049291 <explode_bomb>
0x08048b7b <+33>:    jmp    0x8048b74 <phase_1+26>
```

于是我们最重要的事情就是找到这该死的炸弹中该死的字符串了。虽然看不太懂汇编代码，但是结合计算机组成原理中的压栈操作，我就大胆猜测，那个看起来奇奇怪怪的地址应该是与字符串有关的。于是我先在phase_1设了个断点，然后用x/s 0x804a244看了一下，于是就有了如下的惊喜：

```
(gdb) x/s 0x804a244
0x804a244:      "I can see Russia from my house!"
```

然后第一关的密码就到手了。

答案是：I can see Russia from my house!

（dbq，我一没有房子，二看不到俄罗斯）

4.2 第二关

然后我开始看第二关的代码。第二关显然比第一关长了好多，不过总体来说还不算特别难。

在本地看dump.txt的phase_2函数，第一眼看到某行call了一下函数<read_six_numbers>，这不是明摆着让人输六个整数吗，于是我兴冲冲地输入了1，2，3，4，5，6，bomb!

于是我开始冷静分析，看这个phase_2这么多jmp，像极了分支语句。但是仔细看一下发现很多跳转都是跳到前面的语句，那就是循环语句了。然后结合第一问的经验，我进行猜

测，大概是让我们输入6个数，与它通过循环出来的数进行匹配，如果完全匹配则通过，否则bomb。

首先根据如下

```
0x08048b96 <+25>:    call   0x80492d1 <read_six_numbers
0x08048b9b <+30>:    add    $0x10,%esp
0x08048b9e <+33>:    cmpl   $0x3,-0x24(%ebp)
0x08048ba2 <+37>:    jne    0x8048bab <phase_2+46>
```

我通过在0x08048b9e设断点，然后用gdb调试工具看-0x24(%ebp)处的值，发现该处的值是我们输入的第一个值。然后就很明显了，compare之后，如果该值不等于3就跳转。跳转的地址对应一个call <explode_bomb>，所以很明显就是要求输入的六个整数中第一个整数时3。然后就进行接下来的推测。

当第一个整数是3的时候，它就把1赋值给%ebp，然后每次运行对%ebp进行自增，当寄存器中的值是6的时候再做跳转。这像极了for循环:for(int i = 1; i < 6; i++)。

```
0x08048b9e <+33>:    cmpl   $0x3,-0x24(%ebp)
0x08048ba2 <+37>:    jne    0x8048bab <phase_2+46>
0x08048ba4 <+39>:    mov    $0x1,%ebx
0x08048ba9 <+44>:    jmp    0x8048bba <phase_2+61>
0x08048bab <+46>:    call   0x8049291 <explode_bomb>
0x08048bb0 <+51>:    jmp    0x8048ba4 <phase_2+39>
0x08048bb2 <+53>:    add    $0x1,%ebx
0x08048bb5 <+56>:    cmp    $0x6,%ebx
0x08048bb8 <+59>:    je     0x8048bd4 <phase_2+87>
```

这段代码必须忽视call <explode_bomb>，因为我们的前提是输入的第一个数是3，也就不可能激发炸弹。

所以我们开始研究这个循环。

```
0x08048bb5 <+56>:    cmp    $0x6,%ebx
0x08048bb8 <+59>:    je     0x8048bd4 <phase_2+87>
0x08048bba <+61>:    mov    -0x28(%ebp,%ebx,4),%eax
0x08048bbe <+65>:    mov    %eax,-0x2c(%ebp)
0x08048bc1 <+68>:    add    $0x1,%eax
0x08048bc4 <+71>:    imul   %ebx,%eax
0x08048bc7 <+74>:    cmp    %eax,-0x24(%ebp,%ebx,4)
0x08048bcb <+78>:    je     0x8048bb2 <phase_2+53>
0x08048bcd <+80>:    call   0x8049291 <explode_bomb>
0x08048bd2 <+85>:    jmp    0x8048bb2 <phase_2+53>
0x08048bd4 <+87>:    mov    -0xc(%ebp),%eax
0x08048bd7 <+90>:    xor    %gs:0x14,%eax
0x08048bde <+97>:    jne    0x8048be5 <phase_2+104>
0x08048be0 <+99>:    mov    -0x4(%ebp),%ebx
0x08048be3 <+102>:   leave
0x08048be4 <+103>:   ret
```

当循环变量小于6的时候，在上图中第三行可以看到，出现了计组课上似乎提到过的偏移寻址，就猜测类似于数组。然后再看形式，大致猜测出，首先用for的第i次循环对寄存器中的值进行处理，然后每处理出一个结果就跟数组中的第i个元素比较，相等则继续，不相等就爆炸并结束。

然后就看对寄存器中的数的处理方法。首先毋庸置疑，初始值是3，然后根据以下几行反汇编代码：

```
8048bc1:    83 c0 01                add     $0x1,%eax
8048bc4:    0f af c3                imul    %ebx,%eax
8048bc7:    39 44 9d dc             cmp     %eax,-0x24(%ebp,%ebx,4)
```

百度得到imul是两个寄存器的值相乘后存到后一个寄存器中，于是我们就可以将这个关卡用C语言模拟出来：

```
int main(){
    int a[6];
    for(int i = 0;i < 6;i++){
        cin >> a[i];
    }
    if(a[0] != 3)return 1;//bomb!
    int temp = 3;
    for(int i = 1; i < 6; i++){
        temp++;
        temp *= i;
        if(a[i] != temp)return 1;//bomb!
    }
    return 0;//return 0 代表成功过关
}
```

观察这段代码，为了return 0，我们需要让输入的六个数一直于temp相等，首先第一个数是3，于是我们可以得到：

```
a[0] = 3;
a[1] = (a[0] + 1) * 1 = 4;
a[2] = (a[1] + 1) * 2 = 10;
a[3] = (a[2] + 1) * 3 = 33;
a[4] = (a[3] + 1) * 4 = 136;
a[5] = (a[4] + 1) * 5 = 685;
```

于是第二个的答案是：3 4 10 33 136 685；

```
Phase 1 defused. How about the next one?
3 4 10 33 136 685
That's number 2.  Keep going!
```

就这样划水过了第二关。

4.3 第三关

第三关开始难度就开始略有提升。首先，我结合了第一问的经验，对push的每一个地址都深表怀疑，然后用x/s查看，然后就发现了新大陆：

```
0x08048c07 <+29>:    push   $0x804a28a
0x08048c0c <+34>:    pushl  0x8(%ebp)
0x08048c0f <+37>:    call   0x8048810 <__isoc99_sscanf@plt>
```

用x/s查看0x804a28a存储的内容：

```
(gdb) x/s 0x804a28a
0x804a28a:      "%d %c %d"
```

这就是C语言中的转义说明符嘛，就很明显了，它要我们输入一个整数，一个字符，然后再输一个整数。

问题的关键来了，要输什么样的整数和字符呢？还是采用老办法，设断点，反汇编，慢慢分析。

```
0x08048bf0 <+6>:     mov    %gs:0x14,%eax
0x08048bf6 <+12>:    mov    %eax,-0xc(%ebp)
0x08048bf9 <+15>:    xor    %eax,%eax
0x08048bfb <+17>:    lea    -0x10(%ebp),%eax
0x08048bfe <+20>:    push   %eax
0x08048bff <+21>:    lea    -0x15(%ebp),%eax
0x08048c02 <+24>:    push   %eax
0x08048c03 <+25>:    lea    -0x14(%ebp),%eax
0x08048c06 <+28>:    push   %eax
0x08048c07 <+29>:    push   $0x804a28a
0x08048c0c <+34>:    pushl  0x8(%ebp)
0x08048c0f <+37>:    call   0x8048810 <__isoc99_sscanf@plt>
0x08048c14 <+42>:    add    $0x20,%esp
0x08048c17 <+45>:    cmp    $0x2,%eax
0x08048c1a <+48>:    jle    0x8048c30 <phase_3+70>
0x08048c1c <+50>:    cmpl   $0x7,-0x14(%ebp)
0x08048c20 <+54>:    ja     0x8048cda <phase_3+240>
0x08048c26 <+60>:    mov    -0x14(%ebp),%eax
0x08048c29 <+63>:    jmp    *0x804a2a0(,%eax,4)
0x08048c30 <+70>:    call   0x8049291 <explode_bomb>
0x08048c35 <+75>:    jmp    0x8048c1c <phase_3+50>
0x08048c37 <+77>:    mov    $0x70,%eax
0x08048c3c <+82>:    cmpl   $0x136,-0x10(%ebp)
0x08048c43 <+89>:    je     0x8048ce4 <phase_3+250>
0x08048c49 <+95>:    call   0x8049291 <explode_bomb>
```

在0x08048c17 <+45>处，比较的是2与输入的第一个整数的大小关系。如果第一个整数小于2，跳转到爆炸函数。

然后再来看另外一个条件：在0x08048c1c <+50>处有一个比较：若第一个数大于7，

则跳转到爆炸函数，所以可以得到，第一个整数的范围是2-7。

于是我先取第一个数字为2试试看,然后注意到如图所示的反汇编代码：

```
0x08048c26 <+60>:    mov    -0x14(%ebp),%eax
0x08048c29 <+63>:    jmp    *0x804a2a0(,%eax,4)
```

很容易等待%eax这个寄存器在这个时候存储的是输入的第一个整数，也就是2，于是通过命令p查看跳转的地址：

```
(gdb) x/s *(0x804a2a0 + 4 * 2)
0x8048c58 <phase_3+110>:        "\270v
```

于是它跳转到了0x8048c58处，这次它非常直接，没有任何拐弯抹角：

```
0x08048c58 <+110>:   mov    $0x76,%eax
0x08048c5d <+115>:   cmpl   $0x22b,-0x10(%ebp)
0x08048c64 <+122>:   je     0x8048ce4 <phase_3+250>
0x08048c66 <+124>:   call   0x8049291 <explode_bomb>
```

比较的是0x22b与输入的第三个变量，即第二个整数的关系，而0x22b转化为十进制是555，所以我们现在只需要研究清楚输入的第二个变量，即中间的字符的值。

然后根据它的跳转跳到了0x08048ce4处，如下：

```
0x08048ce4 <+250>:   cmp    %al,-0x15(%ebp)
0x08048ce7 <+253>:   je     0x8048cee <phase_3+260>
0x08048ce9 <+255>:   call   0x8049291 <explode_bomb>
```

即比较输入的第二个变量即中间的字符与寄存器%al中的值，不相等则bomb。于是我查看了一下在这个时候%al中的值：

```
(gdb) p $al
$4 = 118
```

118是字符v的ASCII码，于是这一组答案是2 v 555。运行到这一关并输入：

```
2 v 555
Halfway there!
```

第三关就这样pass了。

但事实上，我事后研究了一下，这应该是一个switch语句的汇编代码，会有很多组答案，根据输入的第一个整数决定后面的答案。由前面的分析可以得到第一个整数的范围是2-7，就根据输入的第一个整数，决定跳转的位置，然后再得到后面的炸弹。

4.4 第四关

第四关是整个过程中最艰难的一关，因为涉及到苦逼的递归，就很难受。我找到答案是用投机取巧的方法，并没有深入理解这个递归函数，但是写实验报告的时候不能这么水，只能慢慢来研究这个苦逼的玩意。

首先贴反汇编代码吧。

```
0x08048d59 <+0>:     push   %ebp
0x08048d5a <+1>:     mov    %esp,%ebp
0x08048d5c <+3>:     sub    $0x18,%esp
0x08048d5f <+6>:     mov    %gs:0x14,%eax
0x08048d65 <+12>:    mov    %eax,-0xc(%ebp)
0x08048d68 <+15>:    xor    %eax,%eax
0x08048d6a <+17>:    lea    -0x10(%ebp),%eax
0x08048d6d <+20>:    push   %eax
0x08048d6e <+21>:    lea    -0x14(%ebp),%eax
0x08048d71 <+24>:    push   %eax
0x08048d72 <+25>:    push   $0x804a4d1
0x08048d77 <+30>:    pushl  0x8(%ebp)
0x08048d7a <+33>:    call   0x8048810 <__isoc99_sscanf@plt>
0x08048d7f <+38>:    add    $0x10,%esp
0x08048d82 <+41>:    cmp    $0x2,%eax
0x08048d85 <+44>:    jne    0x8048d8d <phase_4+52>
0x08048d87 <+46>:    cmpl   $0xe,-0x14(%ebp)
0x08048d8b <+50>:    jbe    0x8048d92 <phase_4+57>
0x08048d8d <+52>:    call   0x8049291 <explode_bomb>
0x08048d92 <+57>:    sub    $0x4,%esp
0x08048d95 <+60>:    push   $0xe
0x08048d97 <+62>:    push   $0x0
-Type <return> to continue, or q <return> to quit---
0x08048d99 <+64>:    pushl  -0x14(%ebp)
0x08048d9c <+67>:    call   0x8048d01 <func4>
0x08048da1 <+72>:    add    $0x10,%esp
0x08048da4 <+75>:    cmp    $0x2b,%eax
0x08048da7 <+78>:    jne    0x8048daf <phase_4+86>
0x08048da9 <+80>:    cmpl   $0x2b,-0x10(%ebp)
0x08048dad <+84>:    je     0x8048db4 <phase_4+91>
0x08048daf <+86>:    call   0x8049291 <explode_bomb>
0x08048db4 <+91>:    mov    -0xc(%ebp),%eax
0x08048db7 <+94>:    xor    %gs:0x14,%eax
0x08048dbe <+101>:   jne    0x8048dc2 <phase_4+105>
0x08048dc0 <+103>:   leave
0x08048dc1 <+104>:   ret
0x08048dc2 <+105>:   call   0x8048790 <__stack_chk_fail@plt>
```

根据之前的经验，在scanf函数调用前的push进栈的内容往往有重要信息，就如图上图中标注出来的部分，我扫描了该字符串对应的地址的内容：

```
(gdb) x /s 0x804a4d1
0x804a4d1:      "%d %d"
```

于是就可以知道这一关要输入两个整数。然后接下来的便是如果scanf返回值不为2则调用bomb函数，这个也印证了要输入两个整数。然后在0x08048d87 <+46>这一行，比较0xe与-0x14(%ebp)的大小关系。0xe即十进制的14，而-0x14(%ebp)是压进栈的第一个

整数，而下面接着一个jbe和一个bomb语句，也就是说，当输入的第一个整数大于14则爆炸，否则跳转。

跳转之后将0xe，0x0和-0x14(%ebp)压入栈中，调用func4，传了三个参数给func4。然后再看后面，我们发现，题目要求这个函数结束后返回的%eax与0x2b即十进制下的43相等，而始终未出现的我们输入的第二个整数也要求与43相等。所以我们大胆猜测：题目要求输入两个整数，第一个整数和0xe，0x0共同传入func4后返回值为43，要求第二个整数就是43。

于是这一切的一切就回到了这个func4上面。func4一堆跳转猛如虎，看得我一脸蒙蔽，我花了一个小时才大概明白它在干什么：

```
08048d01 <func4>:
 8048d01:   55                      push   %ebp
 8048d02:   89 e5                   mov    %esp,%ebp
 8048d04:   53                      push   %ebx
 8048d05:   83 ec 04                sub    $0x4,%esp
 8048d08:   8b 45 08                mov    0x8(%ebp),%eax
 8048d0b:   8b 55 0c                mov    0xc(%ebp),%edx
 8048d0e:   8b 4d 10                mov    0x10(%ebp),%ecx
 8048d11:   29 d1                   sub    %edx,%ecx
 8048d13:   89 cb                   mov    %ecx,%ebx
 8048d15:   c1 eb 1f                shr    $0x1f,%ebx
 8048d18:   01 cb                   add    %ecx,%ebx
 8048d1a:   d1 fb                   sar    %ebx
 8048d1c:   01 d3                   add    %edx,%ebx

 8048d1e:   39 c3                   cmp    %eax,%ebx
 8048d20:   7f 0b                   jg     8048d2d <func4+0x2c> //分支1

 8048d22:   39 c3                   cmp    %eax,%ebx
 8048d24:   7c 1c                   jl     8048d42 <func4+0x41> //分支2

 8048d26:   89 d8                   mov    %ebx,%eax
 8048d28:   8b 5d fc                mov    -0x4(%ebp),%ebx
 8048d2b:   c9                      leave
 8048d2c:   c3                      ret    //返回

 8048d2d:   83 ec 04                sub    $0x4,%esp
 8048d30:   8d 4b ff                lea    -0x1(%ebx),%ecx
 8048d33:   51                      push   %ecx
 8048d34:   52                      push   %edx
 8048d35:   50                      push   %eax
 8048d36:   e8 c6 ff ff ff          call   8048d01 <func4>
 8048d3b:   83 c4 10                add    $0x10,%esp
 8048d3e:   01 c3                   add    %eax,%ebx
 8048d40:   eb e4                   jmp    8048d26 <func4+0x25>
 8048d42:   83 ec 04                sub    $0x4,%esp
 8048d45:   ff 75 10                pushl  0x10(%ebp)
 8048d48:   8d 53 01                lea    0x1(%ebx),%edx
 8048d4b:   52                      push   %edx
 8048d4c:   50                      push   %eax
 8048d4d:   e8 af ff ff ff          call   8048d01 <func4>
 8048d52:   83 c4 10                add    $0x10,%esp
 8048d55:   01 c3                   add    %eax,%ebx
 8048d57:   eb cd                   jmp    8048d26 <func4+0x25>
```

在反汇编代码中，我在关键部分空格并注释。它首先将我们压入栈的三个参数分别传给%ecx，%edx，%eax三个寄存器，然后寄存器之间进行运算，然后根据运算后%eax

与%ebx的大小关系，将运算后的值再次压栈，调用func4。换句话说，func4是我们熟悉的递归函数。

鉴于这个反汇编代码过于迷幻，我将func4写成了c语言的形式：

```
int f( int a, int d, int c )
{
    int b;

    if ( c >= d ) b = ( c + d ) / 2;
    else b = ( 1 + c + d ) / 2;

    if ( b > a ) return f ( a, d, b - 1 ) + b;
    else if ( b == a ) return b;
    else return f( a, b + 1, c ) + b;
}
```

而传入函数的三个参数中，第一个是最后压入栈的-0x4(%ebp)，即输入的第一个整数，而第二个参数则是倒数第二个压入栈的0，最后一个即最先压入栈的14。所以我们的目标是找到一个小于14的x，使得f(x,0,14) = 43。为了实现这个目标我又写了一个程序进行遍历：

```
#include <iostream>
using namespace std;
 int f( int a, int d, int c )
{
    int b;
    if ( c >= d ) b = ( c + d ) / 2;
    else b = ( 1 + c + d ) / 2;
    if ( b > a ) return f ( a, d, b - 1 ) + b;
    else if ( b == a ) return b;
    else return f( a, b + 1, c ) + b;
}
int main(){
    for(int i = 0; i <= 14; i++){
        if(f(i,0,14) == 43){
            cout << i << endl;
        }
    }
    return 0;
}
```

输出为12，即输入的第一个整数为12时符合要求。于是得到第四个的答案为12 43：

```
12 43
So you got that one.  Try this one.
```

4.5 第五关

第五关相比第四关简单了不少。

```
0x08048dc7 <+0>:     push   %ebp
0x08048dc8 <+1>:     mov    %esp,%ebp
0x08048dca <+3>:     push   %ebx
0x08048dcb <+4>:     sub    $0x10,%esp
0x08048dce <+7>:     mov    0x8(%ebp),%ebx
0x08048dd1 <+10>:    push   %ebx
0x08048dd2 <+11>:    call   0x804903b <string_length>
0x08048dd7 <+16>:    add    $0x10,%esp
0x08048dda <+19>:    cmp    $0x6,%eax
0x08048ddd <+22>:    jne    0x8048e0c <phase_5+69>
0x08048ddf <+24>:    mov    %ebx,%eax
0x08048de1 <+26>:    add    $0x6,%ebx
0x08048de4 <+29>:    mov    $0x0,%ecx
0x08048de9 <+34>:    movzbl (%eax),%edx
0x08048dec <+37>:    and    $0xf,%edx
0x08048def <+40>:    add    0x804a2c0(,%edx,4),%ecx
0x08048df6 <+47>:    add    $0x1,%eax
0x08048df9 <+50>:    cmp    %ebx,%eax
0x08048dfb <+52>:    jne    0x8048de9 <phase_5+34>
0x08048dfd <+54>:    cmp    $0x3a,%ecx
0x08048e00 <+57>:    je     0x8048e07 <phase_5+64>
0x08048e02 <+59>:    call   0x8049291 <explode_bomb>
0x08048e07 <+64>:    mov    -0x4(%ebp),%ebx
0x08048e0a <+67>:    leave
0x08048e0b <+68>:    ret
0x08048e0c <+69>:    call   0x8049291 <explode_bomb>
0x08048e11 <+74>:    jmp    0x8048ddf <phase_5+24>
```

在图中标识处前面有一个<string_length>函数，然后将其返回值与6比较，如果不相等则跳转到爆炸函数，说明输入是一个字符串，且要求字符串长度为6。

然后将栈的值赋给%eax，而该值是输入字符的地址，通过循环变量控制。所以再取出该地址对应的字符，将字符的ASCII码再赋给%eax。然后输入字符的ASCII码与0xf与运算再赋给%eax，也就是取字符ASCII码的二进制表示的后四位。注意到将0赋值给%ecx，然后再循环过程中加上(0x804a2c0 + 4*%eax)的值。然后通过六次循环，将%ecx的值与0x3a（即十进制的58）比较，若相等则返回。用c语言描述如下：

```
char s[7];
cin >> s;
int total = 0;
for(int i = 0;i < 6;i++){
    int temp = s[i] & 15;
    total += target[temp];
}
if(total == 58)return 0;
return bomb;
```

所以我们现在最重要的事情是把各个i对应的(0x804a2c0 + 4*i)找出来，然后配凑出58。

```
(gdb) p *(0x804a2c0)
$1 = 2
(gdb) p *(0x804a2c0 + 4)
$2 = 10
(gdb) p *(0x804a2c0 + 4*2)
$3 = 6
```

```
(gdb) p *(0x804a2c0 + 4*3)
$4 = 1
(gdb) p *(0x804a2c0 + 4*4)
$5 = 12
```

看到这里我就没有再看了，因为58 = 12 + 6 + 10 + 10 + 10 + 10，恰好将58拆成其中的六个整数，也就是输入的六个字符的ASCII码取后四位后变成4、2、1、1、1、1。而我们注意到，‘0’的ASCII码的二进制表示是110000，‘1’的是110001……也就是说，数字字符与0xf与运算后得到的数字就是数字字符代表的数字，所以答案可以是421111。当然也可以是其他的字符串，只要满足对应的字符的ASCII码取后四位对应的地址的值的和为58即可。

```
So you got that one.  Try this one
421111
Good work!  On to the next...
```

4.6 第六关

第六关的反汇编代码似乎是所有关卡中最长的一个，而且充满了各种各样的跳转，最初一看真的是一脸懵逼。仔细分析，发现这个关卡分为两个部分，下图是第一部分：

```
08048e13 <phase_6>:
 8048e13:   55                      push   %ebp
 8048e14:   89 e5                   mov    %esp,%ebp
 8048e16:   56                      push   %esi
 8048e17:   53                      push   %ebx
 8048e18:   83 ec 48                sub    $0x48,%esp
 8048e1b:   65 a1 14 00 00 00       mov    %gs:0x14,%eax
 8048e21:   89 45 f4                mov    %eax,-0xc(%ebp)
 8048e24:   31 c0                   xor    %eax,%eax
 8048e26:   8d 45 c4                lea    -0x3c(%ebp),%eax
 8048e29:   50                      push   %eax
 8048e2a:   ff 75 08                pushl  0x8(%ebp)
 8048e2d:   e8 9f 04 00 00          call   80492d1 <read_six_numbers> //读入六个整数
 8048e32:   83 c4 10                add    $0x10,%esp
 8048e35:   be 00 00 00 00          mov    $0x0,%esi
 8048e3a:   8b 44 b5 c4             mov    -0x3c(%ebp,%esi,4),%eax

 8048e3e:   83 e8 01                sub    $0x1,%eax             //给每个整数减1
 8048e41:   83 f8 05                cmp    $0x5,%eax             //减1后与5比较
 8048e44:   77 0c                   ja     8048e52 <phase_6+0x3f> //大于则跳转

 8048e46:   83 c6 01                add    $0x1,%esi
 8048e49:   83 fe 06                cmp    $0x6,%esi
 8048e4c:   74 51                   je     8048e9f <phase_6+0x8c>

 8048e4e:   89 f3                   mov    %esi,%ebx
 8048e50:   eb 0f                   jmp    8048e61 <phase_6+0x4e>

 8048e52:   e8 3a 04 00 00          call   8049291 <explode_bomb>
 8048e57:   eb ed                   jmp    8048e46 <phase_6+0x33>

 8048e59:   83 c3 01                add    $0x1,%ebx
 8048e5c:   83 fb 05                cmp    $0x5,%ebx
 8048e5f:   7f d9                   jg     8048e3a <phase_6+0x27>

 8048e61:   8b 44 9d c4             mov    -0x3c(%ebp,%ebx,4),%eax
 8048e65:   39 44 b5 c0             cmp    %eax,-0x40(%ebp,%esi,4) // 比较相邻两个数是否相等
 8048e69:   75 ee                   jne    8048e59 <phase_6+0x46>

 8048e6b:   e8 21 04 00 00          call   8049291 <explode_bomb>
 8048e70:   eb e7                   jmp    8048e59 <phase_6+0x46>
```

可以看到，首先调用了之前调用过的<read_six_numbers>读入六个整数，然后存入栈中，对每个整数，调用sub函数自减1，如果减1后的整数大于5，则调用bomb函数。由此可见，输入的六个整数都必须小于等于6。而如果输入负数或0，会发现在此处也会bomb，上网查看得知cmp是无符号的比较，如果输入负数，根据补码规则，负数会变成一个非常大的正数，也会爆炸。所以得到输入的整数必须在1-6之间。

然后接下来会有两个嵌套的循环，如果出现输入的两个数相等的情况，则bomb。用C++描述如下：

```
int a[6];
for(int i = 0;i < 6;i++){
    std::cin >> a[i];
    if(a[i] - 1 > 5)bomb();
}
for(int i = 0;i < 6;i++){
    for(int i = j + 1;i < 6;j++){
        if(a[i] == a[j])bomb();
    }
}
```

由此可见，输入的六个数是1，2，3，4，5，6全排列中的一种，于是问题的关键就变成了找到符合题目要求的排列。这也是这个关卡的第二部分。

当循环变量比较完所有的整数，使得输入的六个整数都符合上述要求的时候，就会跳转到图中所指位置：

```
=> 0x08048e88 <+117>:   mov    %ebx,%esi
   0x08048e8a <+119>:   mov    -0x3c(%ebp,%ebx,4),%ecx
   0x08048e8e <+123>:   mov    $0x1,%eax
   0x08048e93 <+128>:   mov    $0x804c154,%edx
   0x08048e98 <+133>:   cmp    $0x1,%ecx
   0x08048e9b <+136>:   jg     0x8048e72 <phase_6+95>
   0x08048e9d <+138>:   jmp    0x8048e7c <phase_6+105>
   0x08048e9f <+140>:   mov    $0x0,%ebx
   0x08048ea4 <+145>:   jmp    0x8048e88 <phase_6+117>
   0x08048ea6 <+147>:   mov    -0x24(%ebp),%ebx
   0x08048ea9 <+150>:   mov    -0x20(%ebp),%eax
   0x08048eac <+153>:   mov    %eax,0x8(%ebx)
   0x08048eaf <+156>:   mov    -0x1c(%ebp),%edx
   0x08048eb2 <+159>:   mov    %edx,0x8(%eax)
   0x08048eb5 <+162>:   mov    -0x18(%ebp),%eax
   0x08048eb8 <+165>:   mov    %eax,0x8(%edx)
   0x08048ebb <+168>:   mov    -0x14(%ebp),%edx
   0x08048ebe <+171>:   mov    %edx,0x8(%eax)
   0x08048ec1 <+174>:   mov    -0x10(%ebp),%eax
   0x08048ec4 <+177>:   mov    %eax,0x8(%edx)
   0x08048ec7 <+180>:   movl   $0x0,0x8(%eax)
   0x08048ece <+187>:   mov    $0x5,%esi
   0x08048ed3 <+192>:   jmp    0x8048edd <phase_6+202>
   0x08048ed5 <+194>:   mov    0x8(%ebx),%ebx
   0x08048ed8 <+197>:   sub    $0x1,%esi
   0x08048edb <+200>:   je     0x8048eed <phase_6+218>
   0x08048edd <+202>:   mov    0x8(%ebx),%eax
   0x08048ee0 <+205>:   mov    (%eax),%eax
   0x08048ee2 <+207>:   cmp    %eax,(%ebx)
   0x08048ee4 <+209>:   jge    0x8048ed5 <phase_6+194>
   0x08048ee6 <+211>:   call   0x8049291 <explode_bomb>
   0x08048eeb <+216>:   jmp    0x8048ed5 <phase_6+194>
---Type <return> to continue, or q <return> to quit---
   0x08048eed <+218>:   mov    -0xc(%ebp),%eax
   0x08048ef0 <+221>:   xor    %gs:0x14,%eax
   0x08048ef7 <+228>:   jne    0x8048f00 <phase_6+237>
```

其中%ebx在跳转前被赋为0，然后我们注意到它把0x804c154这个地址赋给%edx，把输入的整数逐步赋值给%ebx，然后进行一个偏移寻址，将寻得的地址用%eax存起来：

```
8048ed5:   8b 5b 08              mov    0x8(%ebx),%ebx
8048ed8:   83 ee 01              sub    $0x1,%esi
8048edb:   74 10                 je     8048eed <phase_6+0xda>

8048edd:   8b 43 08              mov    0x8(%ebx),%eax
8048ee0:   8b 00                 mov    (%eax),%eax //偏移寻址，存储偏移后下一个整数对应地址对应的值
8048ee2:   39 03                 cmp    %eax,(%ebx) //比较当前整数对应的地址对应的值与下一整数对应地址对应的值
8048ee4:   7d ef                 jge    8048ed5 <phase_6+0xc2>

8048ee6:   e8 a6 03 00 00        call   8049291 <explode_bomb>
8048eeb:   eb e8                 jmp    8048ed5 <phase_6+0xc2>
```

不难看出，这个东西是一个类似于链表的结构，每隔一段就会有一个地址存储一个整数。而我们要通过比较各个整数，得到正确的偏移顺序，让图中注释出比较当前整数对应的地址对应的值与下一整数对应地址对应的值时可以使后一个大，跳转避开炸弹。

问题的关键就是那个链表头对应后面的整数值了。我们需要根据后面的值，将链表的序号根据链表表头值的降序排列排列，因此我读取了链表后面的地址对应的值：

```
(gdb) p *(0x804c154)@18
$3 = {735, 1, 134529376, 358, 2, 134529388, 732, 3, 134529400, 106, 4, 134529412, 368, 5, 134529424,
  122, 6, 0}
```

735 > 732 > 368 > 258 > 122 > 106，对应的节点编号是1 3 5 2 6 4，这就是我们要找的答案了。

```
Good work!  On to the next...
1 3 5 2 6 4
Congratulations! You've defused the bomb!
```

4.7 隐藏关

当破解完第六关的时候，程序退出了，也就是说，可能需要一些神奇的操作来触发隐藏关。

```
8049437:   83 3d ec c7 04 08 06    cmpl   $0x6,0x804c7ec
804943e:   74 12                   je     8049452 <phase_defused+0x36>
8049440:   8b 45 f4                mov    -0xc(%ebp),%eax
8049443:   65 33 05 14 00 00 00    xor    %gs:0x14,%eax
804944a:   0f 85 81 00 00 00       jne    80494d1 <phase_defused+0xb5>
8049450:   c9                      leave
8049451:   c3                      ret
8049452:   83 ec 0c                sub    $0xc,%esp
8049455:   8d 45 a4                lea    -0x5c(%ebp),%eax
8049458:   50                      push   %eax
8049459:   8d 45 a0                lea    -0x60(%ebp),%eax
804945c:   50                      push   %eax
804945d:   8d 45 9c                lea    -0x64(%ebp),%eax
8049460:   50                      push   %eax
8049461:   68 2b a5 04 08          push   $0x804a52b
8049466:   68 f0 c8 04 08          push   $0x804c8f0
804946b:   e8 a0 f3 ff ff          call   8048810 <__isoc99_sscanf@plt>
```

这是第六关调用的phase_defused的部分代码。图中第一行代码中的0x804c7ec，我在不同关卡的phase_defused设置断点查看这个函数值，发现它代表的是关卡数，也就是

到了第六关才会触发接下来的隐藏关。然后跳转到接下来的语句，我发现它在scanf前面push了六个地址，根据前面的经验，我决定查看一下这两个地址对应的值。

```
(gdb) x/s 0x804a52b
0x804a52b:      "%d %d %s"
(gdb) x/s 0x804c8f0
0x804c8f0 <input_strings+240>:  "12 43
```

这好像是我第四关的答案！然后转义说明符告诉我在后面还要加一个字符串。大概就是说，在第四关的答案后面加一个特定的字符串，就会触发隐藏关特效。然后根据跳转我找到了接下来要执行的代码：

```
8049496:   83 ec 08              sub    $0x8,%esp
8049499:   68 34 a5 04 08        push   $0x804a534
804949e:   8d 45 a4              lea    -0x5c(%ebp),%eax

80494a1:   50                    push   %eax
80494a2:   e8 b6 fb ff ff        call   804905d <strings_not_equal>
80494a7:   83 c4 10              add    $0x10,%esp
80494aa:   85 c0                 test   %eax,%eax
80494ac:   75 ca                 jne    8049478 <phase_defused+0x5c>
80494ae:   83 ec 0c              sub    $0xc,%esp
80494b1:   68 80 a3 04 08        push   $0x804a380
80494b6:   e8 05 f3 ff ff        call   80487c0 <puts@plt>
80494bb:   c7 04 24 a8 a3 04 08  movl   $0x804a3a8,(%esp)
80494c2:   e8 f9 f2 ff ff        call   80487c0 <puts@plt>
80494c7:   e8 8d fa ff ff        call   8048f59 <secret_phase>    //here is secret
80494cc:   83 c4 10              add    $0x10,%esp
80494cf:   eb a7                 jmp    8049478 <phase_defused+0x5c>
80494d1:   e8 ba f2 ff ff        call   8048790 <__stack_chk_fail@plt>
```

往往在调用函数前推进栈的东西是传递给函数的参数，而且这个函数是比较字符串是否相同，跟我的猜测相吻合。于是我查看了一下push的地址：

```
(gdb) x/s 0x804a534
0x804a534:      "SecretSYSU"
```

这个字符串本身就告诉我们它是隐藏关的钥匙。于是我退出调试，重新gdb并在第四关的答案后面补充这句话，并且在secret_phase设置了断点，就成功进入了隐藏关。

隐藏关里面好像也没什么东西，根据之前的经验以及编程的经验，如图：

```
0x08048f59 <+0>:     push   %ebp
0x08048f5a <+1>:     mov    %esp,%ebp
0x08048f5c <+3>:     push   %ebx
0x08048f5d <+4>:     sub    $0x4,%esp
0x08048f60 <+7>:     call   0x804930b <read_line>
0x08048f65 <+12>:    sub    $0x4,%esp
0x08048f68 <+15>:    push   $0xa
0x08048f6a <+17>:    push   $0x0
0x08048f6c <+19>:    push   %eax
0x08048f6d <+20>:    call   0x8048880 <strtol@plt>
0x08048f72 <+25>:    mov    %eax,%ebx
0x08048f74 <+27>:    lea    -0x1(%eax),%eax
0x08048f77 <+30>:    add    $0x10,%esp
0x08048f7a <+33>:    cmp    $0x3e8,%eax
0x08048f7f <+38>:    ja     0x8048fb6 <secret_phase+93>
```

可以看到最醒目的是<read_line>函数，然后就是读一整行，然后调用一个名叫<strtol@plt>的函数。而在C语言中，我学过一个名叫strtol的函数，将字符串转换为数字，然后再看一眼下面的内容，可以看到它与其他数字进行了比较，所以推测，该部分将输入的字符串变为整数，然后进行进行操作后再传入fun7。然后可以看到，输入的整数减1后要求比0x3e8小，即输入的整数应该小于1001，然后再传入fun7中。所以问题的关键就是这个fun7的求解。

其实看到fun7和第四关的func4一样单独列出一个函数，我就大概明白它会是一个递归，然后就开始头皮发麻，而且这次的数的范围非常大，是不太可能暴力遍历所有情况的，只能搞清楚这个递归。不过有了前面的经验，这也没有那么的难。

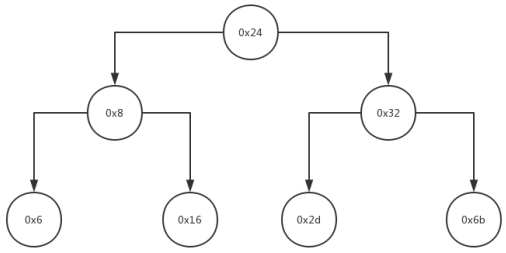
fun7如下图所示：

```
08048f05 <fun7>:
 8048f05:   55                      push   %ebp
 8048f06:   89 e5                   mov    %esp,%ebp
 8048f08:   53                      push   %ebx
 8048f09:   83 ec 04                sub    $0x4,%esp
 8048f0c:   8b 55 08                mov    0x8(%ebp),%edx
 8048f0f:   8b 4d 0c                mov    0xc(%ebp),%ecx

 8048f12:   85 d2                   test   %edx,%edx
 8048f14:   74 3c                   je     8048f52 <fun7+0x4d>

 8048f16:   8b 1a                   mov    (%edx),%ebx
 8048f18:   39 cb                   cmp    %ecx,%ebx
 8048f1a:   7f 0e                   jg     8048f2a <fun7+0x25>

 8048f1c:   b8 00 00 00 00          mov    $0x0,%eax
 8048f21:   39 cb                   cmp    %ecx,%ebx
 8048f23:   75 18                   jne    8048f3d <fun7+0x38>

 8048f25:   8b 5d fc                mov    -0x4(%ebp),%ebx
 8048f28:   c9                      leave
 8048f29:   c3                      ret

 8048f2a:   83 ec 08                sub    $0x8,%esp
 8048f2d:   51                      push   %ecx
 8048f2e:   ff 72 04                pushl  0x4(%edx)

 8048f31:   e8 cf ff ff ff          call   8048f05 <fun7>

 8048f36:   83 c4 10                add    $0x10,%esp
 8048f39:   01 c0                   add    %eax,%eax
 8048f3b:   eb e8                   jmp    8048f25 <fun7+0x20>

 8048f3d:   83 ec 08                sub    $0x8,%esp
 8048f40:   51                      push   %ecx
 8048f41:   ff 72 08                pushl  0x8(%edx)
 8048f44:   e8 bc ff ff ff          call   8048f05 <fun7>
 8048f49:   83 c4 10                add    $0x10,%esp
 8048f4c:   8d 44 00 01             lea    0x1(%eax,%eax,1),%eax
 8048f50:   eb d3                   jmp    8048f25 <fun7+0x20>

 8048f52:   b8 ff ff ff ff          mov    $0xffffffff,%eax
 8048f57:   eb cc                   jmp    8048f25 <fun7+0x20>
```

这个代码就有点类似于二叉查找树了，而我又注意到再secret_phase中调用<fun7>函数后，函数的返回值应该是3，通过cmp函数可以得到。于是我们的目标是让fun7返回3。

而secret_phase在调用<fun7>函数前推进栈的是二叉树的头结点，它的地址是0x804c0a0，于是我研究了一下前两层的二叉树，并将它画了出来：

```
(gdb) p/x *0x804c0a0@3
$4 = {0x24, 0x804c0ac, 0x804c0b8}
(gdb) p/x *0x804c0ac@3
$5 = {0x8, 0x804c0dc, 0x804c0c4}
(gdb) p/x *0x804c0b8@3
$6 = {0x32, 0x804c0d0, 0x804c0e8}
(gdb) p/x *0x804c0dc,
A syntax error in expression, near `
(gdb) p/x *0x804c0dc
$7 = 0x6
(gdb) p/x *0x804c0dc@3
$8 = {0x6, 0x804c100, 0x804c124}
(gdb) p/x *0x804c0c4@3
$9 = {0x16, 0x804c130, 0x804c118}
(gdb) p/x *0x804c0d0@3
$10 = {0x2d, 0x804c0f4, 0x804c13c}
(gdb) p/x *0x804c0e8@3
$11 = {0x6b, 0x804c10c, 0x804c148}
```



这是数据结构课上老师最喜欢的二叉查找树了。

然后我将fun7翻译成伪代码：

```
1.  middle = array [ 0 ];    //array 首地址存储在%edx 中，middle 存储在 middle 中
2.  fun7 ( input ) {           //input 存储在%ecx 中
3.   if( input == middle ) return 0;
4.   else if (input < middle) middle = *( array + 1 ) return 2 * fun ( input );
5.   else middle = *(array + 2)return 2 * fun ( input ) + 1;
6.  }
```

而注意到3 = 2*1 + 1，1 = 2*0 + 1，0 = 0，也就是说，返回值等于3，代表有两次递归。次内层递归返回值为1，即最里层递归返回值为0。也就是两次都调用了input > middle情况。根据二叉树的图片，%eax的值应该是0x6b，即107。也就是最终的答案是107。

于是：

```
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/bomb11$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
I can see Russia from my house!
Phase 1 defused. How about the next one?
3 4 10 33 136 685
That's number 2.  Keep going!
2 v 555
Halfway there!
12 43 SecretSYSU
So you got that one.  Try this one.
421111
Good work!  On to the next...
1 3 5 2 6 4
Curses, you've found the secret phase!
But finding it and solving it are quite different...
107
Wow! You've defused the secret stage!
Congratulations! You've defused the bomb!
Your instructor has been notified and will verify your solution.
```

完结撒花!

五. 实验心得

这次实验创下了两个记录,一个是我上大学以来耗时最长的实验——国庆前为了拆完这个我连着三个晚上两点才睡,另一个是我写过的最长的实验报告(加上这反汇编的代码那就更gg了)。这次实验不算特别难,但是需要我们学习的东西特别多,我觉得虽然这个过程是极其艰苦的,但是也从中学到了非常多的东西。比如大一上学期学的gdb的一点皮毛,经过这次实验我重新回顾并加深了理解。而且为了在Ubuntu下拆弹,我也努力地配置好了环境。最重要的还是,我稍微理解了一点点汇编,知道了汇编的博大精深,也明白了接下来的任务的艰巨性。做这个实验的时候我才慢慢理解到选专业的时候大家都向我的头发投来关切眼神的原因。在这个实验中我其实也有一些失误,比如最开始不懂得在炸弹处设个断点,造成了我不断的bomb,在这方面的数据十分难看。而且很多gdb的命令我也是一知半解,比如TA上次问我关于如何一次性查看所有寄存器的值,我上网才知道info all-registers这个神奇的东西。得与失兼有,但这个实验让我感受到计组的魅力,特别是炸弹拆除时的快乐,让人乐在其中。所以我会更加用心地投入这门课程的学习,继续提高自己的能力,希望下次实验的时候可以少熬一点夜。

【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 d0 e0 ff ff       lea    -0x1f30(%ebx),%eax
 80488fc:   50                      push   %eax
 80488fd:   8d 83 70 e0 ff ff       lea    -0x1f90(%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 08 a1 04 08         push   $0x804a108
 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 86 06 00 00         call   80490c7 <initialize_bomb>
 8048a41:   83 ec 0c              sub    $0xc,%esp
 8048a44:   68 8c a1 04 08         push   $0x804a18c
 8048a49:   e8 72 fd ff ff         call   80487c0 <puts@plt>
 8048a4e:   c7 04 24 c8 a1 04 08    movl   $0x804a1c8,(%esp)
 8048a55:   e8 66 fd ff ff         call   80487c0 <puts@plt>
 8048a5a:   e8 ac 08 00 00         call   804930b <read_line>
 8048a5f:   89 04 24              mov    %eax,(%esp)
 8048a62:   e8 f3 00 00 00         call   8048b5a <phase_1>
 8048a67:   e8 b0 09 00 00         call   804941c <phase_defused>
 8048a6c:   c7 04 24 f4 a1 04 08    movl   $0x804a1f4,(%esp)
 8048a73:   e8 48 fd ff ff         call   80487c0 <puts@plt>
 8048a78:   e8 8e 08 00 00         call   804930b <read_line>
 8048a7d:   89 04 24              mov    %eax,(%esp)
 8048a80:   e8 f8 00 00 00         call   8048b7d <phase_2>
 8048a85:   e8 92 09 00 00         call   804941c <phase_defused>
 8048a8a:   c7 04 24 41 a1 04 08    movl   $0x804a141,(%esp)
 8048a91:   e8 2a fd ff ff         call   80487c0 <puts@plt>
 8048a96:   e8 70 08 00 00         call   804930b <read_line>
 8048a9b:   89 04 24              mov    %eax,(%esp)
 8048a9e:   e8 47 01 00 00         call   8048bea <phase_3>
```

```
8048aa3:   e8 74 09 00 00         call   804941c <phase_defused>
8048aa8:   c7 04 24 5f a1 04 08    movl   $0x804a15f,(%esp)
8048aaf:   e8 0c fd ff ff         call   80487c0 <puts@plt>
8048ab4:   e8 52 08 00 00         call   804930b <read_line>
8048ab9:   89 04 24               mov    %eax,(%esp)
8048abc:   e8 98 02 00 00         call   8048d59 <phase_4>
8048ac1:   e8 56 09 00 00         call   804941c <phase_defused>
8048ac6:   c7 04 24 20 a2 04 08    movl   $0x804a220,(%esp)
8048acd:   e8 ee fc ff ff         call   80487c0 <puts@plt>
8048ad2:   e8 34 08 00 00         call   804930b <read_line>
8048ad7:   89 04 24               mov    %eax,(%esp)
8048ada:   e8 e8 02 00 00         call   8048dc7 <phase_5>
8048adf:   e8 38 09 00 00         call   804941c <phase_defused>
8048ae4:   c7 04 24 6e a1 04 08    movl   $0x804a16e,(%esp)
8048aeb:   e8 d0 fc ff ff         call   80487c0 <puts@plt>
8048af0:   e8 16 08 00 00         call   804930b <read_line>
8048af5:   89 04 24               mov    %eax,(%esp)
8048af8:   e8 16 03 00 00         call   8048e13 <phase_6>
8048afd:   e8 1a 09 00 00         call   804941c <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 0a a1 04 08         push   $0x804a10a
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 27 a1 04 08         push   $0x804a127
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 44 a2 04 08         push   $0x804a244
 8048b65:   ff 75 08              pushl  0x8(%ebp)
 8048b68:   e8 f0 04 00 00         call   804905d <strings_not_equal> //
字符串比较函数
 8048b6d:   83 c4 10              add    $0x10,%esp
 8048b70:   85 c0                 test   %eax,%eax
 8048b72:   75 02                 jne    8048b76 <phase_1+0x1c>
 8048b74:   c9                    leave
 8048b75:   c3                    ret
 8048b76:   e8 16 07 00 00         call   8049291 <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 36 07 00 00         call   80492d1 <read_six_numbers>
 8048b9b:   83 c4 10              add    $0x10,%esp
 8048b9e:   83 7d dc 03           cmpl   $0x3,-0x24(%ebp)
 8048ba2:   75 07                 jne    8048bab <phase_2+0x2e>
 8048ba4:   bb 01 00 00 00         mov    $0x1,%ebx
 8048ba9:   eb 0f                 jmp    8048bba <phase_2+0x3d>
 8048bab:   e8 e1 06 00 00         call   8049291 <explode_bomb>
 8048bb0:   eb f2                 jmp    8048ba4 <phase_2+0x27>
 8048bb2:   83 c3 01              add    $0x1,%ebx
 8048bb5:   83 fb 06              cmp    $0x6,%ebx
 8048bb8:   74 1a                 je     8048bd4 <phase_2+0x57>
 8048bba:   8b 44 9d d8           mov    -0x28(%ebp,%ebx,4),%eax
 8048bbe:   89 45 d4              mov    %eax,-0x2c(%ebp)
 8048bc1:   83 c0 01              add    $0x1,%eax
 8048bc4:   0f af c3              imul   %ebx,%eax
 8048bc7:   39 44 9d dc           cmp    %eax,-0x24(%ebp,%ebx,4)
 8048bcb:   74 e5                 je     8048bb2 <phase_2+0x35>
```

```
 8048bcd:   e8 bf 06 00 00         call   8049291 <explode_bomb>
 8048bd2:   eb de                  jmp    8048bb2 <phase_2+0x35>
 8048bd4:   8b 45 f4               mov    -0xc(%ebp),%eax
 8048bd7:   65 33 05 14 00 00 00    xor    %gs:0x14,%eax
 8048bde:   75 05                  jne    8048be5 <phase_2+0x68>
 8048be0:   8b 5d fc               mov    -0x4(%ebp),%ebx
 8048be3:   c9                     leave
 8048be4:   c3                     ret
 8048be5:   e8 a6 fb ff ff         call   8048790 <__stack_chk_fail@plt>

08048bea <phase_3>:
 8048bea:   55                     push   %ebp
 8048beb:   89 e5                  mov    %esp,%ebp
 8048bed:   83 ec 24               sub    $0x24,%esp
 8048bf0:   65 a1 14 00 00 00      mov    %gs:0x14,%eax
 8048bf6:   89 45 f4               mov    %eax,-0xc(%ebp)
 8048bf9:   31 c0                  xor    %eax,%eax
 8048bfb:   8d 45 f0               lea    -0x10(%ebp),%eax
 8048bfe:   50                     push   %eax
 8048bff:   8d 45 eb               lea    -0x15(%ebp),%eax
 8048c02:   50                     push   %eax
 8048c03:   8d 45 ec               lea    -0x14(%ebp),%eax
 8048c06:   50                     push   %eax
 8048c07:   68 8a a2 04 08         push   $0x804a28a
 8048c0c:   ff 75 08               pushl  0x8(%ebp)
 8048c0f:   e8 fc fb ff ff         call   8048810 <__isoc99_sscanf@plt>
 8048c14:   83 c4 20               add    $0x20,%esp
 8048c17:   83 f8 02               cmp    $0x2,%eax
 8048c1a:   7e 14                  jle    8048c30 <phase_3+0x46>
 8048c1c:   83 7d ec 07            cmpl   $0x7,-0x14(%ebp)
 8048c20:   0f 87 b4 00 00 00      ja     8048cda <phase_3+0xf0>
 8048c26:   8b 45 ec               mov    -0x14(%ebp),%eax
 8048c29:   ff 24 85 a0 a2 04 08    jmp    *0x804a2a0(,%eax,4)
 8048c30:   e8 5c 06 00 00         call   8049291 <explode_bomb>
 8048c35:   eb e5                  jmp    8048c1c <phase_3+0x32>
 8048c37:   b8 70 00 00 00         mov    $0x70,%eax
 8048c3c:   81 7d f0 36 01 00 00    cmpl   $0x136,-0x10(%ebp)
 8048c43:   0f 84 9b 00 00 00      je     8048ce4 <phase_3+0xfa>
 8048c49:   e8 43 06 00 00         call   8049291 <explode_bomb>
 8048c4e:   b8 70 00 00 00         mov    $0x70,%eax
 8048c53:   e9 8c 00 00 00         jmp    8048ce4 <phase_3+0xfa>
 8048c58:   b8 76 00 00 00         mov    $0x76,%eax
 8048c5d:   81 7d f0 2b 02 00 00    cmpl   $0x22b,-0x10(%ebp)
 8048c64:   74 7e                  je     8048ce4 <phase_3+0xfa>
```

```
 8048c66:   e8 26 06 00 00         call   8049291 <explode_bomb>
 8048c6b:   b8 76 00 00 00         mov    $0x76,%eax
 8048c70:   eb 72                  jmp    8048ce4 <phase_3+0xfa>
 8048c72:   b8 6b 00 00 00         mov    $0x6b,%eax
 8048c77:   81 7d f0 a8 02 00 00    cmpl   $0x2a8,-0x10(%ebp)
 8048c7e:   74 64                  je     8048ce4 <phase_3+0xfa>
 8048c80:   e8 0c 06 00 00         call   8049291 <explode_bomb>
 8048c85:   b8 6b 00 00 00         mov    $0x6b,%eax
 8048c8a:   eb 58                  jmp    8048ce4 <phase_3+0xfa>
 8048c8c:   b8 68 00 00 00         mov    $0x68,%eax
 8048c91:   81 7d f0 cb 00 00 00    cmpl   $0xcb,-0x10(%ebp)
 8048c98:   74 4a                  je     8048ce4 <phase_3+0xfa>
 8048c9a:   e8 f2 05 00 00         call   8049291 <explode_bomb>
 8048c9f:   b8 68 00 00 00         mov    $0x68,%eax
 8048ca4:   eb 3e                  jmp    8048ce4 <phase_3+0xfa>
 8048ca6:   b8 70 00 00 00         mov    $0x70,%eax
 8048cab:   81 7d f0 fe 00 00 00    cmpl   $0xfe,-0x10(%ebp)
 8048cb2:   74 30                  je     8048ce4 <phase_3+0xfa>
 8048cb4:   e8 d8 05 00 00         call   8049291 <explode_bomb>
 8048cb9:   b8 70 00 00 00         mov    $0x70,%eax
 8048cbe:   eb 24                  jmp    8048ce4 <phase_3+0xfa>
 8048cc0:   b8 61 00 00 00         mov    $0x61,%eax
 8048cc5:   81 7d f0 a5 02 00 00    cmpl   $0x2a5,-0x10(%ebp)
 8048ccc:   74 16                  je     8048ce4 <phase_3+0xfa>
 8048cce:   e8 be 05 00 00         call   8049291 <explode_bomb>
 8048cd3:   b8 61 00 00 00         mov    $0x61,%eax
 8048cd8:   eb 0a                  jmp    8048ce4 <phase_3+0xfa>
 8048cda:   e8 b2 05 00 00         call   8049291
<explode_bomb>

                                                                  >
 8048cdf:   b8 62 00 00 00         mov    $0x62,%eax
 8048ce4:   38 45 eb               cmp    %al,-0x15(%ebp)
 8048ce7:   74 05                  je     8048cee <phase_3+0x104>
 8048ce9:   e8 a3 05 00 00         call   8049291 <explode_bomb>
 8048cee:   8b 45 f4               mov    -0xc(%ebp),%eax
 8048cf1:   65 33 05 14 00 00 00    xor    %gs:0x14,%eax
 8048cf8:   75 02                  jne    8048cfc <phase_3+0x112>
 8048cfa:   c9                     leave
 8048cfb:   c3                     ret
 8048cfc:   e8 8f fa ff ff         call   8048790 <__stack_chk_fail@plt>

08048d01 <func4>:
 8048d01:   55                     push   %ebp
```

```
8048d02:   89 e5              mov    %esp,%ebp
8048d04:   53                 push   %ebx
8048d05:   83 ec 04           sub    $0x4,%esp
8048d08:   8b 45 08           mov    0x8(%ebp),%eax
8048d0b:   8b 55 0c           mov    0xc(%ebp),%edx
8048d0e:   8b 4d 10           mov    0x10(%ebp),%ecx
8048d11:   29 d1              sub    %edx,%ecx
8048d13:   89 cb              mov    %ecx,%ebx
8048d15:   c1 eb 1f           shr    $0x1f,%ebx
8048d18:   01 cb              add    %ecx,%ebx
8048d1a:   d1 fb              sar    %ebx
8048d1c:   01 d3              add    %edx,%ebx

8048d1e:   39 c3              cmp    %eax,%ebx
8048d20:   7f 0b              jg     8048d2d <func4+0x2c> //分支 1

8048d22:   39 c3              cmp    %eax,%ebx
8048d24:   7c 1c              jl     8048d42 <func4+0x41> //分支 2

8048d26:   89 d8              mov    %ebx,%eax
8048d28:   8b 5d fc           mov    -0x4(%ebp),%ebx
8048d2b:   c9                 leave
8048d2c:   c3                 ret    //返回

8048d2d:   83 ec 04           sub    $0x4,%esp
8048d30:   8d 4b ff           lea    -0x1(%ebx),%ecx
8048d33:   51                 push   %ecx
8048d34:   52                 push   %edx
8048d35:   50                 push   %eax
8048d36:   e8 c6 ff ff ff     call   8048d01 <func4>
8048d3b:   83 c4 10           add    $0x10,%esp
8048d3e:   01 c3              add    %eax,%ebx
8048d40:   eb e4              jmp    8048d26 <func4+0x25>
8048d42:   83 ec 04           sub    $0x4,%esp
8048d45:   ff 75 10           pushl  0x10(%ebp)
8048d48:   8d 53 01           lea    0x1(%ebx),%edx
8048d4b:   52                 push   %edx
8048d4c:   50                 push   %eax
8048d4d:   e8 af ff ff ff     call   8048d01 <func4>
8048d52:   83 c4 10           add    $0x10,%esp
8048d55:   01 c3              add    %eax,%ebx
8048d57:   eb cd              jmp    8048d26 <func4+0x25>

08048d59 <phase_4>:
```

```
 8048d59:   55                    push   %ebp
 8048d5a:   89 e5                 mov    %esp,%ebp
 8048d5c:   83 ec 18              sub    $0x18,%esp
 8048d5f:   65 a1 14 00 00 00      mov    %gs:0x14,%eax
 8048d65:   89 45 f4              mov    %eax,-0xc(%ebp)
 8048d68:   31 c0                 xor    %eax,%eax
 8048d6a:   8d 45 f0              lea    -0x10(%ebp),%eax
 8048d6d:   50                    push   %eax
 8048d6e:   8d 45 ec              lea    -0x14(%ebp),%eax
 8048d71:   50                    push   %eax
 8048d72:   68 d1 a4 04 08         push   $0x804a4d1
 8048d77:   ff 75 08              pushl  0x8(%ebp)
 8048d7a:   e8 91 fa ff ff         call   8048810 <__isoc99_sscanf@plt>
 8048d7f:   83 c4 10              add    $0x10,%esp
 8048d82:   83 f8 02              cmp    $0x2,%eax
 8048d85:   75 06                 jne    8048d8d <phase_4+0x34>
 8048d87:   83 7d ec 0e            cmpl   $0xe,-0x14(%ebp)
 8048d8b:   76 05                 jbe    8048d92 <phase_4+0x39>
 8048d8d:   e8 ff 04 00 00         call   8049291 <explode_bomb>
 8048d92:   83 ec 04              sub    $0x4,%esp
 8048d95:   6a 0e                 push   $0xe
 8048d97:   6a 00                 push   $0x0
 8048d99:   ff 75 ec              pushl  -0x14(%ebp)
 8048d9c:   e8 60 ff ff ff         call   8048d01 <func4>
 8048da1:   83 c4 10              add    $0x10,%esp
 8048da4:   83 f8 2b              cmp    $0x2b,%eax
 8048da7:   75 06                 jne    8048daf <phase_4+0x56>
 8048da9:   83 7d f0 2b            cmpl   $0x2b,-0x10(%ebp)
 8048dad:   74 05                 je     8048db4 <phase_4+0x5b>
 8048daf:   e8 dd 04 00 00         call   8049291 <explode_bomb>
 8048db4:   8b 45 f4              mov    -0xc(%ebp),%eax
 8048db7:   65 33 05 14 00 00 00    xor    %gs:0x14,%eax
 8048dbe:   75 02                 jne    8048dc2 <phase_4+0x69>
 8048dc0:   c9                    leave
 8048dc1:   c3                    ret
 8048dc2:   e8 c9 f9 ff ff         call   8048790 <__stack_chk_fail@plt>

08048dc7 <phase_5>:
 8048dc7:   55                    push   %ebp
 8048dc8:   89 e5                 mov    %esp,%ebp
 8048dca:   53                    push   %ebx
 8048dcb:   83 ec 10              sub    $0x10,%esp
 8048dce:   8b 5d 08              mov    0x8(%ebp),%ebx
 8048dd1:   53                    push   %ebx
```

```
 8048dd2:   e8 64 02 00 00          call   804903b <string_length>
 8048dd7:   83 c4 10                add    $0x10,%esp
 8048dda:   83 f8 06                cmp    $0x6,%eax
 8048ddd:   75 2d                   jne    8048e0c <phase_5+0x45>
 8048ddf:   89 d8                   mov    %ebx,%eax
 8048de1:   83 c3 06                add    $0x6,%ebx
 8048de4:   b9 00 00 00 00          mov    $0x0,%ecx
 8048de9:   0f b6 10                movzbl (%eax),%edx
 8048dec:   83 e2 0f                and    $0xf,%edx
 8048def:   03 0c 95 c0 a2 04 08    add    0x804a2c0(,%edx,4),%ecx
 8048df6:   83 c0 01                add    $0x1,%eax
 8048df9:   39 d8                   cmp    %ebx,%eax
 8048dfb:   75 ec                   jne    8048de9 <phase_5+0x22>
 8048dfd:   83 f9 3a                cmp    $0x3a,%ecx
 8048e00:   74 05                   je     8048e07 <phase_5+0x40>
 8048e02:   e8 8a 04 00 00          call   8049291 <explode_bomb>
 8048e07:   8b 5d fc                mov    -0x4(%ebp),%ebx
 8048e0a:   c9                      leave
 8048e0b:   c3                      ret
 8048e0c:   e8 80 04 00 00          call   8049291 <explode_bomb>
 8048e11:   eb cc                   jmp    8048ddf <phase_5+0x18>

08048e13 <phase_6>:
 8048e13:   55                      push   %ebp
 8048e14:   89 e5                   mov    %esp,%ebp
 8048e16:   56                      push   %esi
 8048e17:   53                      push   %ebx
 8048e18:   83 ec 48                sub    $0x48,%esp
 8048e1b:   65 a1 14 00 00 00       mov    %gs:0x14,%eax
 8048e21:   89 45 f4                mov    %eax,-0xc(%ebp)
 8048e24:   31 c0                   xor    %eax,%eax
 8048e26:   8d 45 c4                lea    -0x3c(%ebp),%eax
 8048e29:   50                      push   %eax
 8048e2a:   ff 75 08                pushl  0x8(%ebp)
 8048e2d:   e8 9f 04 00 00          call   80492d1 <read_six_numbers> //
读入六个整数
 8048e32:   83 c4 10                add    $0x10,%esp
 8048e35:   be 00 00 00 00          mov    $0x0,%esi
 8048e3a:   8b 44 b5 c4             mov    -0x3c(%ebp,%esi,4),%eax

 8048e3e:   83 e8 01                sub    $0x1,%eax              //给每个整数
减 1
 8048e41:   83 f8 05                cmp    $0x5,%eax              //减 1 后与 5
比较
```

```
8048e44:   77 0c                 ja     8048e52 <phase_6+0x3f> //大于则跳转

8048e46:   83 c6 01               add    $0x1,%esi
8048e49:   83 fe 06               cmp    $0x6,%esi
8048e4c:   74 51                  je     8048e9f <phase_6+0x8c>

8048e4e:   89 f3                  mov    %esi,%ebx
8048e50:   eb 0f                  jmp    8048e61 <phase_6+0x4e>

8048e52:   e8 3a 04 00 00          call   8049291 <explode_bomb>
8048e57:   eb ed                  jmp    8048e46 <phase_6+0x33>

8048e59:   83 c3 01               add    $0x1,%ebx
8048e5c:   83 fb 05               cmp    $0x5,%ebx
8048e5f:   7f d9                  jg     8048e3a <phase_6+0x27>

8048e61:   8b 44 9d c4            mov    -0x3c(%ebp,%ebx,4),%eax
8048e65:   39 44 b5 c0            cmp    %eax,-0x40(%ebp,%esi,4) // 比较相邻两个数是否相等
8048e69:   75 ee                  jne    8048e59 <phase_6+0x46>

8048e6b:   e8 21 04 00 00          call   8049291 <explode_bomb>
8048e70:   eb e7                  jmp    8048e59 <phase_6+0x46>

8048e72:   8b 52 08               mov    0x8(%edx),%edx
8048e75:   83 c0 01               add    $0x1,%eax
8048e78:   39 c8                  cmp    %ecx,%eax
8048e7a:   75 f6                  jne    8048e72 <phase_6+0x5f>

8048e7c:   89 54 b5 dc            mov    %edx,-0x24(%ebp,%esi,4)
8048e80:   83 c3 01               add    $0x1,%ebx
8048e83:   83 fb 06               cmp    $0x6,%ebx
8048e86:   74 1e                  je     8048ea6 <phase_6+0x93>

8048e88:   89 de                  mov    %ebx,%esi
8048e8a:   8b 4c 9d c4            mov    -0x3c(%ebp,%ebx,4),%ecx
8048e8e:   b8 01 00 00 00          mov    $0x1,%eax
8048e93:   ba 54 c1 04 08          mov    $0x804c154,%edx

8048e98:   83 f9 01               cmp    $0x1,%ecx
8048e9b:   7f d5                  jg     8048e72 <phase_6+0x5f>
8048e9d:   eb dd                  jmp    8048e7c <phase_6+0x69>
```

```
8048e9f:   bb 00 00 00 00          mov    $0x0,%ebx
8048ea4:   eb e2                   jmp    8048e88 <phase_6+0x75>

8048ea6:   8b 5d dc                mov    -0x24(%ebp),%ebx
8048ea9:   8b 45 e0                mov    -0x20(%ebp),%eax
8048eac:   89 43 08                mov    %eax,0x8(%ebx)
8048eaf:   8b 55 e4                mov    -0x1c(%ebp),%edx
8048eb2:   89 50 08                mov    %edx,0x8(%eax)
8048eb5:   8b 45 e8                mov    -0x18(%ebp),%eax
8048eb8:   89 42 08                mov    %eax,0x8(%edx)
8048ebb:   8b 55 ec                mov    -0x14(%ebp),%edx
8048ebe:   89 50 08                mov    %edx,0x8(%eax)
8048ec1:   8b 45 f0                mov    -0x10(%ebp),%eax
8048ec4:   89 42 08                mov    %eax,0x8(%edx)
8048ec7:   c7 40 08 00 00 00 00    movl   $0x0,0x8(%eax)

8048ece:   be 05 00 00 00          mov    $0x5,%esi
8048ed3:   eb 08                   jmp    8048edd <phase_6+0xca>

8048ed5:   8b 5b 08                mov    0x8(%ebx),%ebx
8048ed8:   83 ee 01                sub    $0x1,%esi
8048edb:   74 10                   je     8048eed <phase_6+0xda>

8048edd:   8b 43 08                mov    0x8(%ebx),%eax
8048ee0:   8b 00                   mov    (%eax),%eax //偏移寻址，存储偏移后
下一个整数对应地址对应的值
8048ee2:   39 03                   cmp    %eax,(%ebx) //比较当前整数对应的地
址对应的值与下一整数对应地址对应的值
8048ee4:   7d ef                   jge    8048ed5 <phase_6+0xc2>

8048ee6:   e8 a6 03 00 00          call   8049291 <explode_bomb>
8048eeb:   eb e8                   jmp    8048ed5 <phase_6+0xc2>

8048eed:   8b 45 f4                mov    -0xc(%ebp),%eax
8048ef0:   65 33 05 14 00 00 00    xor    %gs:0x14,%eax
8048ef7:   75 07                   jne    8048f00 <phase_6+0xed>
8048ef9:   8d 65 f8                lea    -0x8(%ebp),%esp
8048efc:   5b                      pop    %ebx
8048efd:   5e                      pop    %esi
8048efe:   5d                      pop    %ebp
8048eff:   c3                      ret
8048f00:   e8 8b f8 ff ff          call   8048790 <__stack_chk_fail@plt>

08048f05 <fun7>:
```

```
8048f05:   55                  push   %ebp
8048f06:   89 e5               mov    %esp,%ebp
8048f08:   53                  push   %ebx
8048f09:   83 ec 04            sub    $0x4,%esp
8048f0c:   8b 55 08            mov    0x8(%ebp),%edx
8048f0f:   8b 4d 0c            mov    0xc(%ebp),%ecx

8048f12:   85 d2               test   %edx,%edx
8048f14:   74 3c               je     8048f52 <fun7+0x4d>

8048f16:   8b 1a               mov    (%edx),%ebx
8048f18:   39 cb               cmp    %ecx,%ebx
8048f1a:   7f 0e               jg     8048f2a <fun7+0x25>

8048f1c:   b8 00 00 00 00      mov    $0x0,%eax
8048f21:   39 cb               cmp    %ecx,%ebx
8048f23:   75 18               jne    8048f3d <fun7+0x38>

8048f25:   8b 5d fc            mov    -0x4(%ebp),%ebx
8048f28:   c9                  leave
8048f29:   c3                  ret

8048f2a:   83 ec 08            sub    $0x8,%esp
8048f2d:   51                  push   %ecx
8048f2e:   ff 72 04            pushl  0x4(%edx)

8048f31:   e8 cf ff ff ff      call   8048f05 <fun7>

8048f36:   83 c4 10            add    $0x10,%esp
8048f39:   01 c0               add    %eax,%eax
8048f3b:   eb e8               jmp    8048f25 <fun7+0x20>

8048f3d:   83 ec 08            sub    $0x8,%esp
8048f40:   51                  push   %ecx
8048f41:   ff 72 08            pushl  0x8(%edx)
8048f44:   e8 bc ff ff ff      call   8048f05 <fun7>
8048f49:   83 c4 10            add    $0x10,%esp
8048f4c:   8d 44 00 01         lea    0x1(%eax,%eax,1),%eax
8048f50:   eb d3               jmp    8048f25 <fun7+0x20>

8048f52:   b8 ff ff ff ff      mov    $0xffffffff,%eax
8048f57:   eb cc               jmp    8048f25 <fun7+0x20>

08048f59 <secret_phase>:
```

```
 8048f59:   55                    push   %ebp
 8048f5a:   89 e5                 mov    %esp,%ebp
 8048f5c:   53                    push   %ebx
 8048f5d:   83 ec 04              sub    $0x4,%esp
 8048f60:   e8 a6 03 00 00        call   804930b <read_line>
 8048f65:   83 ec 04              sub    $0x4,%esp
 8048f68:   6a 0a                 push   $0xa
 8048f6a:   6a 00                 push   $0x0
 8048f6c:   50                    push   %eax
 8048f6d:   e8 0e f9 ff ff        call   8048880 <strtol@plt>
 8048f72:   89 c3                 mov    %eax,%ebx
 8048f74:   8d 40 ff              lea    -0x1(%eax),%eax
 8048f77:   83 c4 10              add    $0x10,%esp
 8048f7a:   3d e8 03 00 00        cmp    $0x3e8,%eax
 8048f7f:   77 35                 ja     8048fb6 <secret_phase+0x5d>
 8048f81:   83 ec 08              sub    $0x8,%esp
 8048f84:   53                    push   %ebx
 8048f85:   68 a0 c0 04 08        push   $0x804c0a0
 8048f8a:   e8 76 ff ff ff        call   8048f05 <fun7>
 8048f8f:   83 c4 10              add    $0x10,%esp
 8048f92:   83 f8 03              cmp    $0x3,%eax
 8048f95:   74 05                 je     8048f9c <secret_phase+0x43>
 8048f97:   e8 f5 02 00 00        call   8049291 <explode_bomb>
 8048f9c:   83 ec 0c              sub    $0xc,%esp
 8048f9f:   68 64 a2 04 08        push   $0x804a264
 8048fa4:   e8 17 f8 ff ff        call   80487c0 <puts@plt>
 8048fa9:   e8 6e 04 00 00        call   804941c <phase_defused>
 8048fae:   83 c4 10              add    $0x10,%esp
 8048fb1:   8b 5d fc              mov    -0x4(%ebp),%ebx
 8048fb4:   c9                    leave
 8048fb5:   c3                    ret
 8048fb6:   e8 d6 02 00 00        call   8049291 <explode_bomb>
 8048fbb:   eb c4                 jmp    8048f81 <secret_phase+0x28>

08048fbd <sig_handler>:
 8048fbd:   55                    push   %ebp
 8048fbe:   89 e5                 mov    %esp,%ebp
 8048fc0:   83 ec 14              sub    $0x14,%esp
 8048fc3:   68 00 a3 04 08        push   $0x804a300
 8048fc8:   e8 f3 f7 ff ff        call   80487c0 <puts@plt>
 8048fcd:   c7 04 24 03 00 00 00    movl   $0x3,(%esp)
 8048fd4:   e8 97 f7 ff ff        call   8048770 <sleep@plt>
 8048fd9:   83 c4 08              add    $0x8,%esp
 8048fdc:   68 4d a4 04 08        push   $0x804a44d
```

```
 8048fe1:   6a 01                  push   $0x1
 8048fe3:   e8 58 f8 ff ff         call   8048840 <__printf_chk@plt>
 8048fe8:   83 c4 04               add    $0x4,%esp
 8048feb:   ff 35 e4 c7 04 08      pushl  0x804c7e4
 8048ff1:   e8 4a f7 ff ff         call   8048740 <fflush@plt>
 8048ff6:   c7 04 24 01 00 00 00   movl   $0x1,(%esp)
 8048ffd:   e8 6e f7 ff ff         call   8048770 <sleep@plt>
 8049002:   c7 04 24 55 a4 04 08   movl   $0x804a455,(%esp)
 8049009:   e8 b2 f7 ff ff         call   80487c0 <puts@plt>
 804900e:   c7 04 24 10 00 00 00   movl   $0x10,(%esp)
 8049015:   e8 c6 f7 ff ff         call   80487e0 <exit@plt>

0804901a <invalid_phase>:
 804901a:   55                     push   %ebp
 804901b:   89 e5                  mov    %esp,%ebp
 804901d:   83 ec 0c               sub    $0xc,%esp
 8049020:   ff 75 08               pushl  0x8(%ebp)
 8049023:   68 5d a4 04 08         push   $0x804a45d
 8049028:   6a 01                  push   $0x1
 804902a:   e8 11 f8 ff ff         call   8048840 <__printf_chk@plt>
 804902f:   c7 04 24 08 00 00 00   movl   $0x8,(%esp)
 8049036:   e8 a5 f7 ff ff         call   80487e0 <exit@plt>

0804903b <string_length>:
 804903b:   55                     push   %ebp
 804903c:   89 e5                  mov    %esp,%ebp
 804903e:   8b 55 08               mov    0x8(%ebp),%edx
 8049041:   80 3a 00               cmpb   $0x0,(%edx)
 8049044:   74 10                  je     8049056 <string_length+0x1b>
 8049046:   b8 00 00 00 00         mov    $0x0,%eax
 804904b:   83 c0 01               add    $0x1,%eax
 804904e:   80 3c 02 00            cmpb   $0x0,(%edx,%eax,1)
 8049052:   75 f7                  jne    804904b <string_length+0x10>
 8049054:   5d                     pop    %ebp
 8049055:   c3                     ret
 8049056:   b8 00 00 00 00         mov    $0x0,%eax
 804905b:   eb f7                  jmp    8049054 <string_length+0x19>

0804905d <strings_not_equal>:
 804905d:   55                     push   %ebp
 804905e:   89 e5                  mov    %esp,%ebp
 8049060:   57                     push   %edi
 8049061:   56                     push   %esi
 8049062:   53                     push   %ebx
```

```
 8049063:   8b 5d 08             mov    0x8(%ebp),%ebx
 8049066:   8b 75 0c             mov    0xc(%ebp),%esi
 8049069:   53                  push   %ebx
 804906a:   e8 cc ff ff ff        call   804903b <string_length>
 804906f:   89 c7               mov    %eax,%edi
 8049071:   89 34 24             mov    %esi,(%esp)
 8049074:   e8 c2 ff ff ff        call   804903b <string_length>
 8049079:   83 c4 04             add    $0x4,%esp
 804907c:   ba 01 00 00 00        mov    $0x1,%edx
 8049081:   39 c7               cmp    %eax,%edi
 8049083:   74 0a               je     804908f <strings_not_equal+0x32>
 8049085:   89 d0               mov    %edx,%eax
 8049087:   8d 65 f4             lea    -0xc(%ebp),%esp
 804908a:   5b                  pop    %ebx
 804908b:   5e                  pop    %esi
 804908c:   5f                  pop    %edi
 804908d:   5d                  pop    %ebp
 804908e:   c3                  ret
 804908f:   0f b6 03             movzbl (%ebx),%eax
 8049092:   84 c0               test   %al,%al
 8049094:   74 23               je     80490b9 <strings_not_equal+0x5c>
 8049096:   3a 06               cmp    (%esi),%al
 8049098:   75 26               jne    80490c0
<strings_not_equal+0x63>
 804909a:   83 c3 01             add    $0x1,%ebx
 804909d:   83 c6 01             add    $0x1,%esi
 80490a0:   0f b6 03             movzbl (%ebx),%eax
 80490a3:   84 c0               test   %al,%al
 80490a5:   74 0b               je     80490b2 <strings_not_equal+0x55>
 80490a7:   38 06               cmp    %al,(%esi)
 80490a9:   74 ef               je     804909a <strings_not_equal+0x3d>
 80490ab:   ba 01 00 00 00        mov    $0x1,%edx
 80490b0:   eb d3               jmp    8049085
<strings_not_equal+0x28>
 80490b2:   ba 00 00 00 00        mov    $0x0,%edx
 80490b7:   eb cc               jmp    8049085
<strings_not_equal+0x28>
 80490b9:   ba 00 00 00 00        mov    $0x0,%edx
 80490be:   eb c5               jmp    8049085
<strings_not_equal+0x28>
 80490c0:   ba 01 00 00 00        mov    $0x1,%edx
 80490c5:   eb be               jmp    8049085
<strings_not_equal+0x28>
```

```
080490c7 <initialize_bomb>:
 80490c7:   55                    push   %ebp
 80490c8:   89 e5                 mov    %esp,%ebp
 80490ca:   81 ec 20 20 00 00       sub    $0x2020,%esp
 80490d0:   65 a1 14 00 00 00       mov    %gs:0x14,%eax
 80490d6:   89 45 f4              mov    %eax,-0xc(%ebp)
 80490d9:   31 c0                 xor    %eax,%eax
 80490db:   68 bd 8f 04 08          push   $0x8048fbd
 80490e0:   6a 02                 push   $0x2
 80490e2:   e8 79 f6 ff ff          call   8048760 <signal@plt>
 80490e7:   8d 85 f4 df ff ff       lea    -0x200c(%ebp),%eax
 80490ed:   89 04 24              mov    %eax,(%esp)
 80490f0:   e8 2b 0d 00 00          call   8049e20 <init_driver>
 80490f5:   83 c4 10              add    $0x10,%esp
 80490f8:   85 c0                 test   %eax,%eax
 80490fa:   78 0e                 js     804910a <initialize_bomb+0x43>
 80490fc:   8b 45 f4              mov    -0xc(%ebp),%eax
 80490ff:   65 33 05 14 00 00 00    xor    %gs:0x14,%eax
 8049106:   75 24                 jne    804912c <initialize_bomb+0x65>
 8049108:   c9                    leave
 8049109:   c3                    ret
 804910a:   83 ec 04              sub    $0x4,%esp
 804910d:   8d 85 f4 df ff ff       lea    -0x200c(%ebp),%eax
 8049113:   50                    push   %eax
 8049114:   68 6e a4 04 08          push   $0x804a46e
 8049119:   6a 01                 push   $0x1
 804911b:   e8 20 f7 ff ff          call   8048840 <__printf_chk@plt>
 8049120:   c7 04 24 08 00 00 00    movl   $0x8,(%esp)
 8049127:   e8 b4 f6 ff ff          call   80487e0 <exit@plt>
 804912c:   e8 5f f6 ff ff          call   8048790 <__stack_chk_fail@plt>

08049131 <initialize_bomb_solve>:
 8049131:   55                    push   %ebp
 8049132:   89 e5                 mov    %esp,%ebp
 8049134:   5d                    pop    %ebp
 8049135:   c3                    ret

08049136 <blank_line>:
 8049136:   55                    push   %ebp
 8049137:   89 e5                 mov    %esp,%ebp
 8049139:   56                    push   %esi
 804913a:   53                    push   %ebx
 804913b:   8b 75 08              mov    0x8(%ebp),%esi
 804913e:   0f b6 1e              movzbl (%esi),%ebx
```

```
 8049141:   84 db                test   %bl,%bl
 8049143:   74 1b                je     8049160 <blank_line+0x2a>
 8049145:   e8 66 f7 ff ff        call   80488b0 <__ctype_b_loc@plt>
 804914a:   83 c6 01              add    $0x1,%esi
 804914d:   0f be db              movsbl %bl,%ebx
 8049150:   8b 00                 mov    (%eax),%eax
 8049152:   f6 44 58 01 20        testb  $0x20,0x1(%eax,%ebx,2)
 8049157:   75 e5                 jne    804913e <blank_line+0x8>
 8049159:   b8 00 00 00 00        mov    $0x0,%eax
 804915e:   eb 05                 jmp    8049165 <blank_line+0x2f>
 8049160:   b8 01 00 00 00        mov    $0x1,%eax
 8049165:   5b                    pop    %ebx
 8049166:   5e                    pop    %esi
 8049167:   5d                    pop    %ebp
 8049168:   c3                    ret

08049169 <skip>:
 8049169:   55                    push   %ebp
 804916a:   89 e5                 mov    %esp,%ebp
 804916c:   53                    push   %ebx
 804916d:   83 ec 04              sub    $0x4,%esp
 8049170:   83 ec 04              sub    $0x4,%esp
 8049173:   ff 35 f0 c7 04 08     pushl  0x804c7f0
 8049179:   6a 50                 push   $0x50
 804917b:   a1 ec c7 04 08        mov    0x804c7ec,%eax
 8049180:   8d 04 80              lea    (%eax,%eax,4),%eax
 8049183:   c1 e0 04              shl    $0x4,%eax
 8049186:   05 00 c8 04 08        add    $0x804c800,%eax
 804918b:   50                    push   %eax
 804918c:   e8 bf f5 ff ff        call   8048750 <fgets@plt>
 8049191:   89 c3                 mov    %eax,%ebx
 8049193:   83 c4 10              add    $0x10,%esp
 8049196:   85 c0                 test   %eax,%eax
 8049198:   74 10                 je     80491aa <skip+0x41>
 804919a:   83 ec 0c              sub    $0xc,%esp
 804919d:   50                    push   %eax
 804919e:   e8 93 ff ff ff        call   8049136 <blank_line>
 80491a3:   83 c4 10              add    $0x10,%esp
 80491a6:   85 c0                 test   %eax,%eax
 80491a8:   75 c6                 jne    8049170 <skip+0x7>
 80491aa:   89 d8                 mov    %ebx,%eax
 80491ac:   8b 5d fc              mov    -0x4(%ebp),%ebx
 80491af:   c9                    leave
 80491b0:   c3                    ret
```

```
080491b1 <send_msg>:
 80491b1:   55                    push   %ebp
 80491b2:   89 e5                 mov    %esp,%ebp
 80491b4:   57                    push   %edi
 80491b5:   56                    push   %esi
 80491b6:   53                    push   %ebx
 80491b7:   81 ec 1c 40 00 00       sub    $0x401c,%esp
 80491bd:   65 a1 14 00 00 00       mov    %gs:0x14,%eax
 80491c3:   89 45 e4              mov    %eax,-0x1c(%ebp)
 80491c6:   31 c0                 xor    %eax,%eax
 80491c8:   8b 1d ec c7 04 08       mov    0x804c7ec,%ebx
 80491ce:   8d 54 9b fb           lea    -0x5(%ebx,%ebx,4),%edx
 80491d2:   c1 e2 04              shl    $0x4,%edx
 80491d5:   81 c2 00 c8 04 08       add    $0x804c800,%edx
 80491db:   b9 ff ff ff ff          mov    $0xffffffff,%ecx
 80491e0:   89 d7                 mov    %edx,%edi
 80491e2:   f2 ae                 repnz scas %es:(%edi),%al
 80491e4:   89 c8                 mov    %ecx,%eax
 80491e6:   f7 d0                 not    %eax
 80491e8:   83 c0 63              add    $0x63,%eax
 80491eb:   3d 00 20 00 00          cmp    $0x2000,%eax
 80491f0:   77 64                 ja     8049256 <send_msg+0xa5>
 80491f2:   83 7d 08 00             cmpl   $0x0,0x8(%ebp)
 80491f6:   b8 88 a4 04 08          mov    $0x804a488,%eax
 80491fb:   b9 90 a4 04 08          mov    $0x804a490,%ecx
 8049200:   0f 44 c1              cmove  %ecx,%eax
 8049203:   52                    push   %edx
 8049204:   53                    push   %ebx
 8049205:   50                    push   %eax
 8049206:   ff 35 a0 c5 04 08       pushl  0x804c5a0
 804920c:   68 99 a4 04 08          push   $0x804a499
 8049211:   68 00 20 00 00          push   $0x2000
 8049216:   6a 01                 push   $0x1
 8049218:   8d 9d e4 bf ff ff       lea    -0x401c(%ebp),%ebx
 804921e:   53                    push   %ebx
 804921f:   e8 9c f6 ff ff          call   80488c0 <__sprintf_chk@plt>
 8049224:   83 c4 20              add    $0x20,%esp
 8049227:   8d 85 e4 df ff ff       lea    -0x201c(%ebp),%eax
 804922d:   50                    push   %eax
 804922e:   6a 00                 push   $0x0
 8049230:   53                    push   %ebx
 8049231:   68 a0 c1 04 08          push   $0x804c1a0
 8049236:   e8 bf 0d 00 00          call   8049ffa <driver_post>
```

```
 804923b:   83 c4 10              add    $0x10,%esp
 804923e:   85 c0                 test   %eax,%eax
 8049240:   78 2f                 js     8049271 <send_msg+0xc0>
 8049242:   8b 45 e4              mov    -0x1c(%ebp),%eax
 8049245:   65 33 05 14 00 00 00    xor    %gs:0x14,%eax
 804924c:   75 3e                 jne    804928c <send_msg+0xdb>
 804924e:   8d 65 f4              lea    -0xc(%ebp),%esp
 8049251:   5b                    pop    %ebx
 8049252:   5e                    pop    %esi
 8049253:   5f                    pop    %edi
 8049254:   5d                    pop    %ebp
 8049255:   c3                    ret
 8049256:   83 ec 08              sub    $0x8,%esp
 8049259:   68 38 a3 04 08          push   $0x804a338
 804925e:   6a 01                 push   $0x1
 8049260:   e8 db f5 ff ff          call   8048840 <__printf_chk@plt>
 8049265:   c7 04 24 08 00 00 00    movl   $0x8,(%esp)
 804926c:   e8 6f f5 ff ff          call   80487e0 <exit@plt>
 8049271:   83 ec 0c              sub    $0xc,%esp
 8049274:   8d 85 e4 df ff ff       lea    -0x201c(%ebp),%eax
 804927a:   50                    push   %eax
 804927b:   e8 40 f5 ff ff          call   80487c0 <puts@plt>
 8049280:   c7 04 24 00 00 00 00    movl   $0x0,(%esp)
 8049287:   e8 54 f5 ff ff          call   80487e0 <exit@plt>
 804928c:   e8 ff f4 ff ff          call   8048790 <__stack_chk_fail@plt>

08049291 <explode_bomb>:
 8049291:   55                    push   %ebp
 8049292:   89 e5                 mov    %esp,%ebp
 8049294:   83 ec 14              sub    $0x14,%esp
 8049297:   68 a5 a4 04 08          push   $0x804a4a5
 804929c:   e8 1f f5 ff ff          call   80487c0 <puts@plt>
 80492a1:   c7 04 24 ae a4 04 08    movl   $0x804a4ae,(%esp)
 80492a8:   e8 13 f5 ff ff          call   80487c0 <puts@plt>
 80492ad:   c7 04 24 00 00 00 00    movl   $0x0,(%esp)
 80492b4:   e8 f8 fe ff ff          call   80491b1 <send_msg>
 80492b9:   c7 04 24 5c a3 04 08    movl   $0x804a35c,(%esp)
 80492c0:   e8 fb f4 ff ff          call   80487c0 <puts@plt>
 80492c5:   c7 04 24 08 00 00 00    movl   $0x8,(%esp)
 80492cc:   e8 0f f5 ff ff          call   80487e0 <exit@plt>

080492d1 <read_six_numbers>:
 80492d1:   55                    push   %ebp
 80492d2:   89 e5                 mov    %esp,%ebp
```

```
 80492d4:   83 ec 08              sub    $0x8,%esp
 80492d7:   8b 45 0c              mov    0xc(%ebp),%eax
 80492da:   8d 50 14              lea    0x14(%eax),%edx
 80492dd:   52                   push   %edx
 80492de:   8d 50 10              lea    0x10(%eax),%edx
 80492e1:   52                   push   %edx
 80492e2:   8d 50 0c              lea    0xc(%eax),%edx
 80492e5:   52                   push   %edx
 80492e6:   8d 50 08              lea    0x8(%eax),%edx
 80492e9:   52                   push   %edx
 80492ea:   8d 50 04              lea    0x4(%eax),%edx
 80492ed:   52                   push   %edx
 80492ee:   50                   push   %eax
 80492ef:   68 c5 a4 04 08         push   $0x804a4c5
 80492f4:   ff 75 08              pushl  0x8(%ebp)
 80492f7:   e8 14 f5 ff ff         call   8048810 <__isoc99_sscanf@plt>
 80492fc:   83 c4 20              add    $0x20,%esp
 80492ff:   83 f8 05              cmp    $0x5,%eax
 8049302:   7e 02                 jle    8049306 <read_six_numbers+0x35>
 8049304:   c9                   leave
 8049305:   c3                   ret
 8049306:   e8 86 ff ff ff         call   8049291 <explode_bomb>

0804930b <read_line>:
 804930b:   55                   push   %ebp
 804930c:   89 e5                 mov    %esp,%ebp
 804930e:   57                   push   %edi
 804930f:   56                   push   %esi
 8049310:   53                   push   %ebx
 8049311:   83 ec 0c              sub    $0xc,%esp
 8049314:   e8 50 fe ff ff         call   8049169 <skip>
 8049319:   85 c0                 test   %eax,%eax
 804931b:   74 53                 je     8049370 <read_line+0x65>
 804931d:   8b 15 ec c7 04 08      mov    0x804c7ec,%edx
 8049323:   8d 1c 92              lea    (%edx,%edx,4),%ebx
 8049326:   c1 e3 04              shl    $0x4,%ebx
 8049329:   81 c3 00 c8 04 08      add    $0x804c800,%ebx
 804932f:   b9 ff ff ff ff         mov    $0xffffffff,%ecx
 8049334:   b8 00 00 00 00         mov    $0x0,%eax
 8049339:   89 df                 mov    %ebx,%edi
 804933b:   f2 ae                 repnz scas %es:(%edi),%al
 804933d:   89 ce                 mov    %ecx,%esi
 804933f:   f7 d6                 not    %esi
 8049341:   89 f1                 mov    %esi,%ecx
```

```
8049343:   83 e9 01              sub    $0x1,%ecx
8049346:   83 f9 4e              cmp    $0x4e,%ecx
8049349:   0f 8f 95 00 00 00      jg     80493e4 <read_line+0xd9>
804934f:   8d 04 92              lea    (%edx,%edx,4),%eax
8049352:   c1 e0 04              shl    $0x4,%eax
8049355:   c6 84 01 ff c7 04 08    movb   $0x0,0x804c7ff(%ecx,%eax,1)
804935c:   00
804935d:   83 c2 01              add    $0x1,%edx
8049360:   89 15 ec c7 04 08      mov    %edx,0x804c7ec
8049366:   89 d8                 mov    %ebx,%eax
8049368:   8d 65 f4              lea    -0xc(%ebp),%esp
804936b:   5b                    pop    %ebx
804936c:   5e                    pop    %esi
804936d:   5f                    pop    %edi
804936e:   5d                    pop    %ebp
804936f:   c3                    ret
8049370:   a1 e0 c7 04 08         mov    0x804c7e0,%eax
8049375:   39 05 f0 c7 04 08      cmp    %eax,0x804c7f0
804937b:   74 1e                 je     804939b <read_line+0x90>
804937d:   83 ec 0c              sub    $0xc,%esp
8049380:   68 f5 a4 04 08         push   $0x804a4f5
8049385:   e8 26 f4 ff ff         call   80487b0 <getenv@plt>
804938a:   83 c4 10              add    $0x10,%esp
804938d:   85 c0                 test   %eax,%eax
804938f:   74 23                 je     80493b4 <read_line+0xa9>
8049391:   83 ec 0c              sub    $0xc,%esp
8049394:   6a 00                 push   $0x0
8049396:   e8 45 f4 ff ff         call   80487e0 <exit@plt>
804939b:   83 ec 0c              sub    $0xc,%esp
804939e:   68 d7 a4 04 08         push   $0x804a4d7
80493a3:   e8 18 f4 ff ff         call   80487c0 <puts@plt>
80493a8:   c7 04 24 08 00 00 00    movl   $0x8,(%esp)
80493af:   e8 2c f4 ff ff         call   80487e0 <exit@plt>
80493b4:   a1 e0 c7 04 08         mov    0x804c7e0,%eax
80493b9:   a3 f0 c7 04 08         mov    %eax,0x804c7f0
80493be:   e8 a6 fd ff ff         call   8049169 <skip>
80493c3:   85 c0                 test   %eax,%eax
80493c5:   0f 85 52 ff ff ff      jne    804931d <read_line+0x12>
80493cb:   83 ec 0c              sub    $0xc,%esp
80493ce:   68 d7 a4 04 08         push   $0x804a4d7
80493d3:   e8 e8 f3 ff ff         call   80487c0 <puts@plt>
80493d8:   c7 04 24 00 00 00 00    movl   $0x0,(%esp)
80493df:   e8 fc f3 ff ff         call   80487e0 <exit@plt>
80493e4:   83 ec 0c              sub    $0xc,%esp
```

```
 80493e7:   68 00 a5 04 08          push   $0x804a500
 80493ec:   e8 cf f3 ff ff          call   80487c0 <puts@plt>
 80493f1:   a1 ec c7 04 08          mov    0x804c7ec,%eax
 80493f6:   8d 50 01                lea    0x1(%eax),%edx
 80493f9:   89 15 ec c7 04 08       mov    %edx,0x804c7ec
 80493ff:   6b c0 50                imul   $0x50,%eax,%eax
 8049402:   05 00 c8 04 08          add    $0x804c800,%eax
 8049407:   ba 1b a5 04 08          mov    $0x804a51b,%edx
 804940c:   b9 04 00 00 00          mov    $0x4,%ecx
 8049411:   89 c7                   mov    %eax,%edi
 8049413:   89 d6                   mov    %edx,%esi
 8049415:   f3 a5                   rep movsl %ds:(%esi),%es:(%edi)
 8049417:   e8 75 fe ff ff          call   8049291 <explode_bomb>

0804941c <phase_defused>:
 804941c:   55                      push   %ebp
 804941d:   89 e5                   mov    %esp,%ebp
 804941f:   83 ec 74                sub    $0x74,%esp
 8049422:   65 a1 14 00 00 00       mov    %gs:0x14,%eax
 8049428:   89 45 f4                mov    %eax,-0xc(%ebp)
 804942b:   31 c0                   xor    %eax,%eax
 804942d:   6a 01                   push   $0x1
 804942f:   e8 7d fd ff ff          call   80491b1 <send_msg>
 8049434:   83 c4 10                add    $0x10,%esp
 8049437:   83 3d ec c7 04 08 06    cmpl   $0x6,0x804c7ec
 804943e:   74 12                   je     8049452 <phase_defused+0x36>
 8049440:   8b 45 f4                mov    -0xc(%ebp),%eax
 8049443:   65 33 05 14 00 00 00    xor    %gs:0x14,%eax
 804944a:   0f 85 81 00 00 00       jne    80494d1 <phase_defused+0xb5>
 8049450:   c9                      leave
 8049451:   c3                      ret
 8049452:   83 ec 0c                sub    $0xc,%esp
 8049455:   8d 45 a4                lea    -0x5c(%ebp),%eax
 8049458:   50                      push   %eax
 8049459:   8d 45 a0                lea    -0x60(%ebp),%eax
 804945c:   50                      push   %eax
 804945d:   8d 45 9c                lea    -0x64(%ebp),%eax
 8049460:   50                      push   %eax
 8049461:   68 2b a5 04 08          push   $0x804a52b
 8049466:   68 f0 c8 04 08          push   $0x804c8f0
 804946b:   e8 a0 f3 ff ff          call   8048810 <__isoc99_sscanf@plt>
 8049470:   83 c4 20                add    $0x20,%esp
 8049473:   83 f8 03                cmp    $0x3,%eax
 8049476:   74 1e                   je     8049496 <phase_defused+0x7a>
```

```
 8049478:   83 ec 0c             sub    $0xc,%esp
 804947b:   68 e0 a3 04 08        push   $0x804a3e0
 8049480:   e8 3b f3 ff ff        call   80487c0 <puts@plt>
 8049485:   c7 04 24 0c a4 04 08    movl   $0x804a40c,(%esp)
 804948c:   e8 2f f3 ff ff        call   80487c0 <puts@plt>
 8049491:   83 c4 10             add    $0x10,%esp
 8049494:   eb aa                jmp    8049440 <phase_defused+0x24>

 8049496:   83 ec 08             sub    $0x8,%esp
 8049499:   68 34 a5 04 08        push   $0x804a534
 804949e:   8d 45 a4             lea    -0x5c(%ebp),%eax

 80494a1:   50                   push   %eax
 80494a2:   e8 b6 fb ff ff        call   804905d <strings_not_equal>
 80494a7:   83 c4 10             add    $0x10,%esp
 80494aa:   85 c0                test   %eax,%eax
 80494ac:   75 ca                jne    8049478 <phase_defused+0x5c>
 80494ae:   83 ec 0c             sub    $0xc,%esp
 80494b1:   68 80 a3 04 08        push   $0x804a380
 80494b6:   e8 05 f3 ff ff        call   80487c0 <puts@plt>
 80494bb:   c7 04 24 a8 a3 04 08    movl   $0x804a3a8,(%esp)
 80494c2:   e8 f9 f2 ff ff        call   80487c0 <puts@plt>
 80494c7:   e8 8d fa ff ff        call   8048f59 <secret_phase>    //here
is secret
 80494cc:   83 c4 10             add    $0x10,%esp
 80494cf:   eb a7                jmp    8049478 <phase_defused+0x5c>
 80494d1:   e8 ba f2 ff ff        call   8048790 <__stack_chk_fail@plt>

080494d6 <sigalrm_handler>:
 80494d6:   55                   push   %ebp
 80494d7:   89 e5                mov    %esp,%ebp
 80494d9:   83 ec 08             sub    $0x8,%esp
 80494dc:   6a 00                push   $0x0
 80494de:   68 8c a5 04 08        push   $0x804a58c
 80494e3:   6a 01                push   $0x1
 80494e5:   ff 35 c0 c7 04 08      pushl  0x804c7c0
 80494eb:   e8 70 f3 ff ff        call   8048860 <__fprintf_chk@plt>
 80494f0:   c7 04 24 01 00 00 00    movl   $0x1,(%esp)
 80494f7:   e8 e4 f2 ff ff        call   80487e0 <exit@plt>

080494fc <rio_readlineb>:
 80494fc:   55                   push   %ebp
 80494fd:   89 e5                mov    %esp,%ebp
```

```
80494ff:   57                   push   %edi
8049500:   56                   push   %esi
8049501:   53                   push   %ebx
8049502:   83 ec 1c              sub    $0x1c,%esp
8049505:   89 d7                mov    %edx,%edi
8049507:   83 f9 01              cmp    $0x1,%ecx
804950a:   76 7d                jbe    8049589 <rio_readlineb+0x8d>
804950c:   89 c3                mov    %eax,%ebx
804950e:   8d 44 0a ff           lea    -0x1(%edx,%ecx,1),%eax
8049512:   89 45 e0              mov    %eax,-0x20(%ebp)
8049515:   c7 45 e4 01 00 00 00    movl   $0x1,-0x1c(%ebp)
804951c:   8d 73 0c              lea    0xc(%ebx),%esi
804951f:   eb 0a                jmp    804952b <rio_readlineb+0x2f>
8049521:   e8 0a f3 ff ff         call   8048830 <__errno_location@plt>
8049526:   83 38 04              cmpl   $0x4,(%eax)
8049529:   75 67                jne    8049592 <rio_readlineb+0x96>
804952b:   8b 43 04              mov    0x4(%ebx),%eax
804952e:   85 c0                test   %eax,%eax
8049530:   7f 23                jg     8049555 <rio_readlineb+0x59>
8049532:   83 ec 04              sub    $0x4,%esp
8049535:   68 00 20 00 00         push   $0x2000
804953a:   56                   push   %esi
804953b:   ff 33                pushl  (%ebx)
804953d:   e8 ee f1 ff ff         call   8048730 <read@plt>
8049542:   89 43 04              mov    %eax,0x4(%ebx)
8049545:   83 c4 10              add    $0x10,%esp
8049548:   85 c0                test   %eax,%eax
804954a:   78 d5                js     8049521 <rio_readlineb+0x25>
804954c:   85 c0                test   %eax,%eax
804954e:   74 49                je     8049599 <rio_readlineb+0x9d>
8049550:   89 73 08              mov    %esi,0x8(%ebx)
8049553:   eb d6                jmp    804952b <rio_readlineb+0x2f>
8049555:   8b 4b 08              mov    0x8(%ebx),%ecx
8049558:   0f b6 11              movzbl (%ecx),%edx
804955b:   83 c1 01              add    $0x1,%ecx
804955e:   89 4b 08              mov    %ecx,0x8(%ebx)
8049561:   83 e8 01              sub    $0x1,%eax
8049564:   89 43 04              mov    %eax,0x4(%ebx)
8049567:   83 c7 01              add    $0x1,%edi
804956a:   88 57 ff              mov    %dl,-0x1(%edi)
804956d:   80 fa 0a              cmp    $0xa,%dl
8049570:   74 09                je     804957b <rio_readlineb+0x7f>
8049572:   83 45 e4 01           addl   $0x1,-0x1c(%ebp)
8049576:   3b 7d e0              cmp    -0x20(%ebp),%edi
```

```
 8049579:   75 b0                jne    804952b <rio_readlineb+0x2f>
 804957b:   c6 07 00             movb   $0x0,(%edi)
 804957e:   8b 45 e4             mov    -0x1c(%ebp),%eax
 8049581:   8d 65 f4             lea    -0xc(%ebp),%esp
 8049584:   5b                   pop    %ebx
 8049585:   5e                   pop    %esi
 8049586:   5f                   pop    %edi
 8049587:   5d                   pop    %ebp
 8049588:   c3                   ret
 8049589:   c7 45 e4 01 00 00 00    movl   $0x1,-0x1c(%ebp)
 8049590:   eb e9                jmp    804957b <rio_readlineb+0x7f>
 8049592:   b8 ff ff ff ff         mov    $0xffffffff,%eax
 8049597:   eb 05                jmp    804959e <rio_readlineb+0xa2>
 8049599:   b8 00 00 00 00         mov    $0x0,%eax
 804959e:   85 c0                test   %eax,%eax
 80495a0:   75 0f                jne    80495b1 <rio_readlineb+0xb5>
 80495a2:   83 7d e4 01            cmpl   $0x1,-0x1c(%ebp)
 80495a6:   75 d3                jne    804957b <rio_readlineb+0x7f>
 80495a8:   c7 45 e4 00 00 00 00    movl   $0x0,-0x1c(%ebp)
 80495af:   eb cd                jmp    804957e <rio_readlineb+0x82>
 80495b1:   c7 45 e4 ff ff ff ff    movl   $0xffffffff,-0x1c(%ebp)
 80495b8:   eb c4                jmp    804957e <rio_readlineb+0x82>

080495ba <submitr>:
 80495ba:   55                   push   %ebp
 80495bb:   89 e5                mov    %esp,%ebp
 80495bd:   57                   push   %edi
 80495be:   56                   push   %esi
 80495bf:   53                   push   %ebx
 80495c0:   81 ec 60 a0 00 00       sub    $0xa060,%esp
 80495c6:   8b 75 08             mov    0x8(%ebp),%esi
 80495c9:   8b 45 10             mov    0x10(%ebp),%eax
 80495cc:   89 85 ac 5f ff ff       mov    %eax,-0xa054(%ebp)
 80495d2:   8b 45 14             mov    0x14(%ebp),%eax
 80495d5:   89 85 a8 5f ff ff       mov    %eax,-0xa058(%ebp)
 80495db:   8b 45 18             mov    0x18(%ebp),%eax
 80495de:   89 85 a4 5f ff ff       mov    %eax,-0xa05c(%ebp)
 80495e4:   8b 5d 1c             mov    0x1c(%ebp),%ebx
 80495e7:   8b 45 20             mov    0x20(%ebp),%eax
 80495ea:   89 85 a0 5f ff ff       mov    %eax,-0xa060(%ebp)
 80495f0:   65 a1 14 00 00 00       mov    %gs:0x14,%eax
 80495f6:   89 45 e4             mov    %eax,-0x1c(%ebp)
 80495f9:   31 c0                xor    %eax,%eax
 80495fb:   c7 85 c4 5f ff ff 00    movl   $0x0,-0xa03c(%ebp)
```

```
8049602:   00 00 00
8049605:   6a 00                push   $0x0
8049607:   6a 01                push   $0x1
8049609:   6a 02                push   $0x2
804960b:   e8 40 f2 ff ff        call   8048850 <socket@plt>
8049610:   89 85 b0 5f ff ff      mov    %eax,-0xa050(%ebp)
8049616:   83 c4 10              add    $0x10,%esp
8049619:   85 c0                 test   %eax,%eax
804961b:   0f 88 30 01 00 00      js     8049751 <submitr+0x197>
8049621:   83 ec 0c              sub    $0xc,%esp
8049624:   56                   push   %esi
8049625:   e8 46 f2 ff ff        call   8048870 <gethostbyname@plt>
804962a:   83 c4 10              add    $0x10,%esp
804962d:   85 c0                 test   %eax,%eax
804962f:   0f 84 70 01 00 00      je     80497a5 <submitr+0x1eb>
8049635:   8d b5 c8 5f ff ff      lea    -0xa038(%ebp),%esi
804963b:   c7 85 ca 5f ff ff 00    movl   $0x0,-0xa036(%ebp)
8049642:   00 00 00
8049645:   c7 85 ce 5f ff ff 00    movl   $0x0,-0xa032(%ebp)
804964c:   00 00 00
804964f:   c7 85 d2 5f ff ff 00    movl   $0x0,-0xa02e(%ebp)
8049656:   00 00 00
8049659:   66 c7 85 d6 5f ff ff    movw   $0x0,-0xa02a(%ebp)
8049660:   00 00
8049662:   66 c7 85 c8 5f ff ff    movw   $0x2,-0xa038(%ebp)
8049669:   02 00
804966b:   6a 0c                push   $0xc
804966d:   ff 70 0c              pushl  0xc(%eax)
8049670:   8b 40 10              mov    0x10(%eax),%eax
8049673:   ff 30                 pushl  (%eax)
8049675:   8d 85 cc 5f ff ff      lea    -0xa034(%ebp),%eax
804967b:   50                   push   %eax
804967c:   e8 4f f1 ff ff        call   80487d0 <__memmove_chk@plt>
8049681:   0f b7 45 0c           movzwl 0xc(%ebp),%eax
8049685:   66 c1 c8 08           ror    $0x8,%ax
8049689:   66 89 85 ca 5f ff ff    mov    %ax,-0xa036(%ebp)
8049690:   83 c4 0c              add    $0xc,%esp
8049693:   6a 10                push   $0x10
8049695:   56                   push   %esi
8049696:   ff b5 b0 5f ff ff      pushl  -0xa050(%ebp)
804969c:   e8 ef f1 ff ff        call   8048890 <connect@plt>
80496a1:   83 c4 10              add    $0x10,%esp
80496a4:   85 c0                 test   %eax,%eax
80496a6:   0f 88 70 01 00 00      js     804981c <submitr+0x262>
```

```
80496ac:   ba ff ff ff ff        mov    $0xffffffff,%edx
80496b1:   b8 00 00 00 00        mov    $0x0,%eax
80496b6:   89 d1                 mov    %edx,%ecx
80496b8:   89 df                 mov    %ebx,%edi
80496ba:   f2 ae                 repnz scas %es:(%edi),%al
80496bc:   89 ce                 mov    %ecx,%esi
80496be:   f7 d6                 not    %esi
80496c0:   89 d1                 mov    %edx,%ecx
80496c2:   8b bd ac 5f ff ff     mov    -0xa054(%ebp),%edi
80496c8:   f2 ae                 repnz scas %es:(%edi),%al
80496ca:   89 8d b4 5f ff ff     mov    %ecx,-0xa04c(%ebp)
80496d0:   89 d1                 mov    %edx,%ecx
80496d2:   8b bd a8 5f ff ff     mov    -0xa058(%ebp),%edi
80496d8:   f2 ae                 repnz scas %es:(%edi),%al
80496da:   89 cf                 mov    %ecx,%edi
80496dc:   f7 d7                 not    %edi
80496de:   89 bd 9c 5f ff ff     mov    %edi,-0xa064(%ebp)
80496e4:   89 d1                 mov    %edx,%ecx
80496e6:   8b bd a4 5f ff ff     mov    -0xa05c(%ebp),%edi
80496ec:   f2 ae                 repnz scas %es:(%edi),%al
80496ee:   8b 95 9c 5f ff ff     mov    -0xa064(%ebp),%edx
80496f4:   2b 95 b4 5f ff ff     sub    -0xa04c(%ebp),%edx
80496fa:   29 ca                 sub    %ecx,%edx
80496fc:   8d 44 76 fd           lea    -0x3(%esi,%esi,2),%eax
8049700:   8d 44 02 7b           lea    0x7b(%edx,%eax,1),%eax
8049704:   3d 00 20 00 00        cmp    $0x2000,%eax
8049709:   0f 87 76 01 00 00     ja     8049885 <submitr+0x2cb>
804970f:   8d 95 e4 9f ff ff     lea    -0x601c(%ebp),%edx
8049715:   b9 00 08 00 00        mov    $0x800,%ecx
804971a:   b8 00 00 00 00        mov    $0x0,%eax
804971f:   89 d7                 mov    %edx,%edi
8049721:   f3 ab                 rep stos %eax,%es:(%edi)
8049723:   b9 ff ff ff ff        mov    $0xffffffff,%ecx
8049728:   89 df                 mov    %ebx,%edi
804972a:   f2 ae                 repnz scas %es:(%edi),%al
804972c:   89 ca                 mov    %ecx,%edx
804972e:   f7 d2                 not    %edx
8049730:   89 d1                 mov    %edx,%ecx
8049732:   83 e9 01              sub    $0x1,%ecx
8049735:   89 8d b4 5f ff ff     mov    %ecx,-0xa04c(%ebp)
804973b:   0f 84 3b 06 00 00     je     8049d7c <submitr+0x7c2>
8049741:   8d b5 e4 9f ff ff     lea    -0x601c(%ebp),%esi
8049747:   bf 01 00 00 00        mov    $0x1,%edi
804974c:   e9 cb 01 00 00        jmp    804991c <submitr+0x362>
```

```
8049751:   8b 85 a0 5f ff ff      mov    -0xa060(%ebp),%eax
8049757:   c7 00 45 72 72 6f      movl   $0x6f727245,(%eax)
804975d:   c7 40 04 72 3a 20 43   movl   $0x43203a72,0x4(%eax)
8049764:   c7 40 08 6c 69 65 6e   movl   $0x6e65696c,0x8(%eax)
804976b:   c7 40 0c 74 20 75 6e   movl   $0x6e752074,0xc(%eax)
8049772:   c7 40 10 61 62 6c 65   movl   $0x656c6261,0x10(%eax)
8049779:   c7 40 14 20 74 6f 20   movl   $0x206f7420,0x14(%eax)
8049780:   c7 40 18 63 72 65 61   movl   $0x61657263,0x18(%eax)
8049787:   c7 40 1c 74 65 20 73   movl   $0x73206574,0x1c(%eax)
804978e:   c7 40 20 6f 63 6b 65   movl   $0x656b636f,0x20(%eax)
8049795:   66 c7 40 24 74 00      movw   $0x74,0x24(%eax)
804979b:   b8 ff ff ff ff         mov    $0xffffffff,%eax
80497a0:   e9 f2 04 00 00         jmp    8049c97 <submitr+0x6dd>
80497a5:   8b 85 a0 5f ff ff      mov    -0xa060(%ebp),%eax
80497ab:   c7 00 45 72 72 6f      movl   $0x6f727245,(%eax)
80497b1:   c7 40 04 72 3a 20 44   movl   $0x44203a72,0x4(%eax)
80497b8:   c7 40 08 4e 53 20 69   movl   $0x6920534e,0x8(%eax)
80497bf:   c7 40 0c 73 20 75 6e   movl   $0x6e752073,0xc(%eax)
80497c6:   c7 40 10 61 62 6c 65   movl   $0x656c6261,0x10(%eax)
80497cd:   c7 40 14 20 74 6f 20   movl   $0x206f7420,0x14(%eax)
80497d4:   c7 40 18 72 65 73 6f   movl   $0x6f736572,0x18(%eax)
80497db:   c7 40 1c 6c 76 65 20   movl   $0x2065766c,0x1c(%eax)
80497e2:   c7 40 20 73 65 72 76   movl   $0x76726573,0x20(%eax)
80497e9:   c7 40 24 65 72 20 61   movl   $0x61207265,0x24(%eax)
80497f0:   c7 40 28 64 64 72 65   movl   $0x65726464,0x28(%eax)
80497f7:   66 c7 40 2c 73 73      movw   $0x7373,0x2c(%eax)
80497fd:   c6 40 2e 00            movb   $0x0,0x2e(%eax)
8049801:   83 ec 0c               sub    $0xc,%esp
8049804:   ff b5 b0 5f ff ff      pushl  -0xa050(%ebp)
804980a:   e8 91 f0 ff ff         call   80488a0 <close@plt>
804980f:   83 c4 10               add    $0x10,%esp
8049812:   b8 ff ff ff ff         mov    $0xffffffff,%eax
8049817:   e9 7b 04 00 00         jmp    8049c97 <submitr+0x6dd>
804981c:   8b 85 a0 5f ff ff      mov    -0xa060(%ebp),%eax
8049822:   c7 00 45 72 72 6f      movl   $0x6f727245,(%eax)
8049828:   c7 40 04 72 3a 20 55   movl   $0x55203a72,0x4(%eax)
804982f:   c7 40 08 6e 61 62 6c   movl   $0x6c62616e,0x8(%eax)
8049836:   c7 40 0c 65 20 74 6f   movl   $0x6f742065,0xc(%eax)
804983d:   c7 40 10 20 63 6f 6e   movl   $0x6e6f6320,0x10(%eax)
8049844:   c7 40 14 6e 65 63 74   movl   $0x7463656e,0x14(%eax)
804984b:   c7 40 18 20 74 6f 20   movl   $0x206f7420,0x18(%eax)
8049852:   c7 40 1c 74 68 65 20   movl   $0x20656874,0x1c(%eax)
8049859:   c7 40 20 73 65 72 76   movl   $0x76726573,0x20(%eax)
8049860:   66 c7 40 24 65 72      movw   $0x7265,0x24(%eax)
```

```
8049866:   c6 40 26 00           movb   $0x0,0x26(%eax)
804986a:   83 ec 0c              sub    $0xc,%esp
804986d:   ff b5 b0 5f ff ff      pushl  -0xa050(%ebp)
8049873:   e8 28 f0 ff ff         call   80488a0 <close@plt>
8049878:   83 c4 10              add    $0x10,%esp
804987b:   b8 ff ff ff ff         mov    $0xffffffff,%eax
8049880:   e9 12 04 00 00         jmp    8049c97 <submitr+0x6dd>
8049885:   8b 85 a0 5f ff ff      mov    -0xa060(%ebp),%eax
804988b:   c7 00 45 72 72 6f      movl   $0x6f727245,(%eax)
8049891:   c7 40 04 72 3a 20 52    movl   $0x52203a72,0x4(%eax)
8049898:   c7 40 08 65 73 75 6c    movl   $0x6c757365,0x8(%eax)
804989f:   c7 40 0c 74 20 73 74    movl   $0x74732074,0xc(%eax)
80498a6:   c7 40 10 72 69 6e 67    movl   $0x676e6972,0x10(%eax)
80498ad:   c7 40 14 20 74 6f 6f    movl   $0x6f6f7420,0x14(%eax)
80498b4:   c7 40 18 20 6c 61 72    movl   $0x72616c20,0x18(%eax)
80498bb:   c7 40 1c 67 65 2e 20    movl   $0x202e6567,0x1c(%eax)
80498c2:   c7 40 20 49 6e 63 72    movl   $0x72636e49,0x20(%eax)
80498c9:   c7 40 24 65 61 73 65    movl   $0x65736165,0x24(%eax)
80498d0:   c7 40 28 20 53 55 42    movl   $0x42555320,0x28(%eax)
80498d7:   c7 40 2c 4d 49 54 52    movl   $0x5254494d,0x2c(%eax)
80498de:   c7 40 30 5f 4d 41 58    movl   $0x58414d5f,0x30(%eax)
80498e5:   c7 40 34 42 55 46 00    movl   $0x465542,0x34(%eax)
80498ec:   83 ec 0c              sub    $0xc,%esp
80498ef:   ff b5 b0 5f ff ff      pushl  -0xa050(%ebp)
80498f5:   e8 a6 ef ff ff         call   80488a0 <close@plt>
80498fa:   83 c4 10              add    $0x10,%esp
80498fd:   b8 ff ff ff ff         mov    $0xffffffff,%eax
8049902:   e9 90 03 00 00         jmp    8049c97 <submitr+0x6dd>
8049907:   88 16                 mov    %dl,(%esi)
8049909:   8d 76 01              lea    0x1(%esi),%esi
804990c:   83 c3 01              add    $0x1,%ebx
804990f:   83 ad b4 5f ff ff 01    subl   $0x1,-0xa04c(%ebp)
8049916:   0f 84 60 04 00 00      je     8049d7c <submitr+0x7c2>
804991c:   0f b6 13              movzbl (%ebx),%edx
804991f:   8d 4a d6              lea    -0x2a(%edx),%ecx
8049922:   89 f8                 mov    %edi,%eax
8049924:   80 f9 0f              cmp    $0xf,%cl
8049927:   77 0d                 ja     8049936 <submitr+0x37c>
8049929:   b8 d9 ff 00 00         mov    $0xffd9,%eax
804992e:   d3 e8                 shr    %cl,%eax
8049930:   83 f0 01              xor    $0x1,%eax
8049933:   83 e0 01              and    $0x1,%eax
8049936:   84 c0                 test   %al,%al
8049938:   74 cd                 je     8049907 <submitr+0x34d>
```

```
804993a:   80 fa 5f             cmp    $0x5f,%dl
804993d:   74 c8                je     8049907 <submitr+0x34d>
804993f:   89 d0                mov    %edx,%eax
8049941:   83 e0 df             and    $0xffffffdf,%eax
8049944:   83 e8 41             sub    $0x41,%eax
8049947:   3c 19                cmp    $0x19,%al
8049949:   76 bc                jbe    8049907 <submitr+0x34d>
804994b:   80 fa 20             cmp    $0x20,%dl
804994e:   74 54                je     80499a4 <submitr+0x3ea>
8049950:   8d 42 e0             lea    -0x20(%edx),%eax
8049953:   3c 5f                cmp    $0x5f,%al
8049955:   76 09                jbe    8049960 <submitr+0x3a6>
8049957:   80 fa 09             cmp    $0x9,%dl
804995a:   0f 85 d1 03 00 00    jne    8049d31 <submitr+0x777>
8049960:   83 ec 0c             sub    $0xc,%esp
8049963:   0f b6 d2             movzbl %dl,%edx
8049966:   52                   push   %edx
8049967:   68 98 a6 04 08       push   $0x804a698
804996c:   6a 08                push   $0x8
804996e:   6a 01                push   $0x1
8049970:   8d 85 e4 df ff ff    lea    -0x201c(%ebp),%eax
8049976:   50                   push   %eax
8049977:   e8 44 ef ff ff       call   80488c0 <__sprintf_chk@plt>
804997c:   0f b6 85 e4 df ff ff movzbl -0x201c(%ebp),%eax
8049983:   88 06                mov    %al,(%esi)
8049985:   0f b6 85 e5 df ff ff movzbl -0x201b(%ebp),%eax
804998c:   88 46 01             mov    %al,0x1(%esi)
804998f:   0f b6 85 e6 df ff ff movzbl -0x201a(%ebp),%eax
8049996:   88 46 02             mov    %al,0x2(%esi)
8049999:   83 c4 20             add    $0x20,%esp
804999c:   8d 76 03             lea    0x3(%esi),%esi
804999f:   e9 68 ff ff ff       jmp    804990c <submitr+0x352>
80499a4:   c6 06 2b             movb   $0x2b,(%esi)
80499a7:   8d 76 01             lea    0x1(%esi),%esi
80499aa:   e9 5d ff ff ff       jmp    804990c <submitr+0x352>
80499af:   01 c6                add    %eax,%esi
80499b1:   29 c3                sub    %eax,%ebx
80499b3:   74 27                je     80499dc <submitr+0x422>
80499b5:   83 ec 04             sub    $0x4,%esp
80499b8:   53                   push   %ebx
80499b9:   56                   push   %esi
80499ba:   57                   push   %edi
80499bb:   e8 40 ee ff ff       call   8048800 <write@plt>
80499c0:   83 c4 10             add    $0x10,%esp
```

```
80499c3:   85 c0                test   %eax,%eax
80499c5:   7f e8                jg     80499af <submitr+0x3f5>
80499c7:   e8 64 ee ff ff        call   8048830 <__errno_location@plt>
80499cc:   83 38 04              cmpl   $0x4,(%eax)
80499cf:   0f 85 41 01 00 00      jne    8049b16 <submitr+0x55c>
80499d5:   b8 00 00 00 00         mov    $0x0,%eax
80499da:   eb d3                jmp    80499af <submitr+0x3f5>
80499dc:   8b bd b4 5f ff ff      mov    -0xa04c(%ebp),%edi
80499e2:   85 ff                test   %edi,%edi
80499e4:   0f 88 2c 01 00 00      js     8049b16 <submitr+0x55c>
80499ea:   8b 85 b0 5f ff ff      mov    -0xa050(%ebp),%eax
80499f0:   89 85 d8 5f ff ff      mov    %eax,-0xa028(%ebp)
80499f6:   c7 85 dc 5f ff ff 00    movl   $0x0,-0xa024(%ebp)
80499fd:   00 00 00
8049a00:   8d 85 e4 5f ff ff      lea    -0xa01c(%ebp),%eax
8049a06:   89 85 e0 5f ff ff      mov    %eax,-0xa020(%ebp)
8049a0c:   b9 00 20 00 00         mov    $0x2000,%ecx
8049a11:   8d 95 e4 7f ff ff      lea    -0x801c(%ebp),%edx
8049a17:   8d 85 d8 5f ff ff      lea    -0xa028(%ebp),%eax
8049a1d:   e8 da fa ff ff         call   80494fc <rio_readlineb>
8049a22:   85 c0                test   %eax,%eax
8049a24:   0f 8e 59 01 00 00      jle    8049b83 <submitr+0x5c9>
8049a2a:   83 ec 0c              sub    $0xc,%esp
8049a2d:   8d 85 e4 df ff ff      lea    -0x201c(%ebp),%eax
8049a33:   50                   push   %eax
8049a34:   8d 85 c4 5f ff ff      lea    -0xa03c(%ebp),%eax
8049a3a:   50                   push   %eax
8049a3b:   8d 85 e4 bf ff ff      lea    -0x401c(%ebp),%eax
8049a41:   50                   push   %eax
8049a42:   68 9f a6 04 08         push   $0x804a69f
8049a47:   8d 85 e4 7f ff ff      lea    -0x801c(%ebp),%eax
8049a4d:   50                   push   %eax
8049a4e:   e8 bd ed ff ff         call   8048810 <__isoc99_sscanf@plt>
8049a53:   8b 85 c4 5f ff ff      mov    -0xa03c(%ebp),%eax
8049a59:   83 c4 20              add    $0x20,%esp
8049a5c:   3d c8 00 00 00         cmp    $0xc8,%eax
8049a61:   0f 85 9d 01 00 00      jne    8049c04 <submitr+0x64a>
8049a67:   8d 9d e4 7f ff ff      lea    -0x801c(%ebp),%ebx
8049a6d:   bf b0 a6 04 08         mov    $0x804a6b0,%edi
8049a72:   b9 03 00 00 00         mov    $0x3,%ecx
8049a77:   89 de                mov    %ebx,%esi
8049a79:   f3 a6                repz cmpsb %es:(%edi),%ds:(%esi)
8049a7b:   0f 97 c0              seta   %al
8049a7e:   1c 00                sbb    $0x0,%al
```

```
8049a80:   84 c0                test   %al,%al
8049a82:   0f 84 b3 01 00 00    je     8049c3b <submitr+0x681>
8049a88:   b9 00 20 00 00       mov    $0x2000,%ecx
8049a8d:   89 da                mov    %ebx,%edx
8049a8f:   8d 85 d8 5f ff ff    lea    -0xa028(%ebp),%eax
8049a95:   e8 62 fa ff ff       call   80494fc <rio_readlineb>
8049a9a:   85 c0                test   %eax,%eax
8049a9c:   7f cf                jg     8049a6d <submitr+0x4b3>
8049a9e:   8b 85 a0 5f ff ff    mov    -0xa060(%ebp),%eax
8049aa4:   c7 00 45 72 72 6f    movl   $0x6f727245,(%eax)
8049aaa:   c7 40 04 72 3a 20 43 movl   $0x43203a72,0x4(%eax)
8049ab1:   c7 40 08 6c 69 65 6e movl   $0x6e65696c,0x8(%eax)
8049ab8:   c7 40 0c 74 20 75 6e movl   $0x6e752074,0xc(%eax)
8049abf:   c7 40 10 61 62 6c 65 movl   $0x656c6261,0x10(%eax)
8049ac6:   c7 40 14 20 74 6f 20 movl   $0x206f7420,0x14(%eax)
8049acd:   c7 40 18 72 65 61 64 movl   $0x64616572,0x18(%eax)
8049ad4:   c7 40 1c 20 68 65 61 movl   $0x61656820,0x1c(%eax)
8049adb:   c7 40 20 64 65 72 73 movl   $0x73726564,0x20(%eax)
8049ae2:   c7 40 24 20 66 72 6f movl   $0x6f726620,0x24(%eax)
8049ae9:   c7 40 28 6d 20 73 65 movl   $0x6573206d,0x28(%eax)
8049af0:   c7 40 2c 72 76 65 72 movl   $0x72657672,0x2c(%eax)
8049af7:   c6 40 30 00          movb   $0x0,0x30(%eax)
8049afb:   83 ec 0c             sub    $0xc,%esp
8049afe:   ff b5 b0 5f ff ff    pushl  -0xa050(%ebp)
8049b04:   e8 97 ed ff ff       call   80488a0 <close@plt>
8049b09:   83 c4 10             add    $0x10,%esp
8049b0c:   b8 ff ff ff ff       mov    $0xffffffff,%eax
8049b11:   e9 81 01 00 00       jmp    8049c97 <submitr+0x6dd>
8049b16:   8b 85 a0 5f ff ff    mov    -0xa060(%ebp),%eax
8049b1c:   c7 00 45 72 72 6f    movl   $0x6f727245,(%eax)
8049b22:   c7 40 04 72 3a 20 43 movl   $0x43203a72,0x4(%eax)
8049b29:   c7 40 08 6c 69 65 6e movl   $0x6e65696c,0x8(%eax)
8049b30:   c7 40 0c 74 20 75 6e movl   $0x6e752074,0xc(%eax)
8049b37:   c7 40 10 61 62 6c 65 movl   $0x656c6261,0x10(%eax)
8049b3e:   c7 40 14 20 74 6f 20 movl   $0x206f7420,0x14(%eax)
8049b45:   c7 40 18 77 72 69 74 movl   $0x74697277,0x18(%eax)
8049b4c:   c7 40 1c 65 20 74 6f movl   $0x6f742065,0x1c(%eax)
8049b53:   c7 40 20 20 74 68 65 movl   $0x65687420,0x20(%eax)
8049b5a:   c7 40 24 20 73 65 72 movl   $0x72657320,0x24(%eax)
8049b61:   c7 40 28 76 65 72 00 movl   $0x726576,0x28(%eax)
8049b68:   83 ec 0c             sub    $0xc,%esp
8049b6b:   ff b5 b0 5f ff ff    pushl  -0xa050(%ebp)
8049b71:   e8 2a ed ff ff       call   80488a0 <close@plt>
8049b76:   83 c4 10             add    $0x10,%esp
```

```
8049b79:   b8 ff ff ff ff          mov    $0xffffffff,%eax
8049b7e:   e9 14 01 00 00          jmp    8049c97 <submitr+0x6dd>
8049b83:   8b 85 a0 5f ff ff       mov    -0xa060(%ebp),%eax
8049b89:   c7 00 45 72 72 6f       movl   $0x6f727245,(%eax)
8049b8f:   c7 40 04 72 3a 20 43    movl   $0x43203a72,0x4(%eax)
8049b96:   c7 40 08 6c 69 65 6e    movl   $0x6e65696c,0x8(%eax)
8049b9d:   c7 40 0c 74 20 75 6e    movl   $0x6e752074,0xc(%eax)
8049ba4:   c7 40 10 61 62 6c 65    movl   $0x656c6261,0x10(%eax)
8049bab:   c7 40 14 20 74 6f 20    movl   $0x206f7420,0x14(%eax)
8049bb2:   c7 40 18 72 65 61 64    movl   $0x64616572,0x18(%eax)
8049bb9:   c7 40 1c 20 66 69 72    movl   $0x72696620,0x1c(%eax)
8049bc0:   c7 40 20 73 74 20 68    movl   $0x68207473,0x20(%eax)
8049bc7:   c7 40 24 65 61 64 65    movl   $0x65646165,0x24(%eax)
8049bce:   c7 40 28 72 20 66 72    movl   $0x72662072,0x28(%eax)
8049bd5:   c7 40 2c 6f 6d 20 73    movl   $0x73206d6f,0x2c(%eax)
8049bdc:   c7 40 30 65 72 76 65    movl   $0x65767265,0x30(%eax)
8049be3:   66 c7 40 34 72 00       movw   $0x72,0x34(%eax)
8049be9:   83 ec 0c                sub    $0xc,%esp
8049bec:   ff b5 b0 5f ff ff       pushl  -0xa050(%ebp)
8049bf2:   e8 a9 ec ff ff          call   80488a0 <close@plt>
8049bf7:   83 c4 10                add    $0x10,%esp
8049bfa:   b8 ff ff ff ff          mov    $0xffffffff,%eax
8049bff:   e9 93 00 00 00          jmp    8049c97 <submitr+0x6dd>
8049c04:   83 ec 08                sub    $0x8,%esp
8049c07:   8d 95 e4 df ff ff       lea    -0x201c(%ebp),%edx
8049c0d:   52                      push   %edx
8049c0e:   50                      push   %eax
8049c0f:   68 b0 a5 04 08          push   $0x804a5b0
8049c14:   6a ff                   push   $0xffffffff
8049c16:   6a 01                   push   $0x1
8049c18:   ff b5 a0 5f ff ff       pushl  -0xa060(%ebp)
8049c1e:   e8 9d ec ff ff          call   80488c0 <__sprintf_chk@plt>
8049c23:   83 c4 14                add    $0x14,%esp
8049c26:   ff b5 b0 5f ff ff       pushl  -0xa050(%ebp)
8049c2c:   e8 6f ec ff ff          call   80488a0 <close@plt>
8049c31:   83 c4 10                add    $0x10,%esp
8049c34:   b8 ff ff ff ff          mov    $0xffffffff,%eax
8049c39:   eb 5c                   jmp    8049c97 <submitr+0x6dd>
8049c3b:   b9 00 20 00 00          mov    $0x2000,%ecx
8049c40:   8d 95 e4 7f ff ff       lea    -0x801c(%ebp),%edx
8049c46:   8d 85 d8 5f ff ff       lea    -0xa028(%ebp),%eax
8049c4c:   e8 ab f8 ff ff          call   80494fc <rio_readlineb>
8049c51:   85 c0                   test   %eax,%eax
8049c53:   7e 5a                   jle    8049caf <submitr+0x6f5>
```

```
8049c55:   83 ec 08              sub    $0x8,%esp
8049c58:   8d 85 e4 7f ff ff       lea    -0x801c(%ebp),%eax
8049c5e:   50                    push   %eax
8049c5f:   8b b5 a0 5f ff ff       mov    -0xa060(%ebp),%esi
8049c65:   56                    push   %esi
8049c66:   e8 35 eb ff ff         call   80487a0 <strcpy@plt>
8049c6b:   83 c4 04              add    $0x4,%esp
8049c6e:   ff b5 b0 5f ff ff       pushl  -0xa050(%ebp)
8049c74:   e8 27 ec ff ff         call   80488a0 <close@plt>
8049c79:   bf b3 a6 04 08         mov    $0x804a6b3,%edi
8049c7e:   b9 03 00 00 00         mov    $0x3,%ecx
8049c83:   f3 a6                 repz cmpsb %es:(%edi),%ds:(%esi)
8049c85:   0f 97 c0              seta   %al
8049c88:   1c 00                 sbb    $0x0,%al
8049c8a:   83 c4 10              add    $0x10,%esp
8049c8d:   84 c0                 test   %al,%al
8049c8f:   0f 95 c0              setne  %al
8049c92:   0f b6 c0              movzbl %al,%eax
8049c95:   f7 d8                 neg    %eax
8049c97:   8b 7d e4              mov    -0x1c(%ebp),%edi
8049c9a:   65 33 3d 14 00 00 00    xor    %gs:0x14,%edi
8049ca1:   0f 85 3d 01 00 00       jne    8049de4 <submitr+0x82a>
8049ca7:   8d 65 f4              lea    -0xc(%ebp),%esp
8049caa:   5b                    pop    %ebx
8049cab:   5e                    pop    %esi
8049cac:   5f                    pop    %edi
8049cad:   5d                    pop    %ebp
8049cae:   c3                    ret
8049caf:   8b 85 a0 5f ff ff       mov    -0xa060(%ebp),%eax
8049cb5:   c7 00 45 72 72 6f       movl   $0x6f727245,(%eax)
8049cbb:   c7 40 04 72 3a 20 43    movl   $0x43203a72,0x4(%eax)
8049cc2:   c7 40 08 6c 69 65 6e    movl   $0x6e65696c,0x8(%eax)
8049cc9:   c7 40 0c 74 20 75 6e    movl   $0x6e752074,0xc(%eax)
8049cd0:   c7 40 10 61 62 6c 65    movl   $0x656c6261,0x10(%eax)
8049cd7:   c7 40 14 20 74 6f 20    movl   $0x206f7420,0x14(%eax)
8049cde:   c7 40 18 72 65 61 64    movl   $0x64616572,0x18(%eax)
8049ce5:   c7 40 1c 20 73 74 61    movl   $0x61747320,0x1c(%eax)
8049cec:   c7 40 20 74 75 73 20    movl   $0x20737574,0x20(%eax)
8049cf3:   c7 40 24 6d 65 73 73    movl   $0x7373656d,0x24(%eax)
8049cfa:   c7 40 28 61 67 65 20    movl   $0x20656761,0x28(%eax)
8049d01:   c7 40 2c 66 72 6f 6d    movl   $0x6d6f7266,0x2c(%eax)
8049d08:   c7 40 30 20 73 65 72    movl   $0x72657320,0x30(%eax)
8049d0f:   c7 40 34 76 65 72 00    movl   $0x726576,0x34(%eax)
8049d16:   83 ec 0c              sub    $0xc,%esp
```

```
8049d19:   ff b5 b0 5f ff ff      pushl  -0xa050(%ebp)
8049d1f:   e8 7c eb ff ff         call   80488a0 <close@plt>
8049d24:   83 c4 10               add    $0x10,%esp
8049d27:   b8 ff ff ff ff         mov    $0xffffffff,%eax
8049d2c:   e9 66 ff ff ff         jmp    8049c97 <submitr+0x6dd>
8049d31:   a1 e0 a5 04 08         mov    0x804a5e0,%eax
8049d36:   8b bd a0 5f ff ff      mov    -0xa060(%ebp),%edi
8049d3c:   89 07                  mov    %eax,(%edi)
8049d3e:   a1 1f a6 04 08         mov    0x804a61f,%eax
8049d43:   89 47 3f               mov    %eax,0x3f(%edi)
8049d46:   89 f8                  mov    %edi,%eax
8049d48:   8d 7f 04               lea    0x4(%edi),%edi
8049d4b:   83 e7 fc               and    $0xfffffffc,%edi
8049d4e:   29 f8                  sub    %edi,%eax
8049d50:   be e0 a5 04 08         mov    $0x804a5e0,%esi
8049d55:   29 c6                  sub    %eax,%esi
8049d57:   83 c0 43               add    $0x43,%eax
8049d5a:   c1 e8 02               shr    $0x2,%eax
8049d5d:   89 c1                  mov    %eax,%ecx
8049d5f:   f3 a5                  rep movsl %ds:(%esi),%es:(%edi)
8049d61:   83 ec 0c               sub    $0xc,%esp
8049d64:   ff b5 b0 5f ff ff      pushl  -0xa050(%ebp)
8049d6a:   e8 31 eb ff ff         call   80488a0 <close@plt>
8049d6f:   83 c4 10               add    $0x10,%esp
8049d72:   b8 ff ff ff ff         mov    $0xffffffff,%eax
8049d77:   e9 1b ff ff ff         jmp    8049c97 <submitr+0x6dd>
8049d7c:   8d 85 e4 9f ff ff      lea    -0x601c(%ebp),%eax
8049d82:   50                     push   %eax
8049d83:   ff b5 a4 5f ff ff      pushl  -0xa05c(%ebp)
8049d89:   ff b5 a8 5f ff ff      pushl  -0xa058(%ebp)
8049d8f:   ff b5 ac 5f ff ff      pushl  -0xa054(%ebp)
8049d95:   68 24 a6 04 08         push   $0x804a624
8049d9a:   68 00 20 00 00         push   $0x2000
8049d9f:   6a 01                  push   $0x1
8049da1:   8d bd e4 7f ff ff      lea    -0x801c(%ebp),%edi
8049da7:   57                     push   %edi
8049da8:   e8 13 eb ff ff         call   80488c0 <__sprintf_chk@plt>
8049dad:   b9 ff ff ff ff         mov    $0xffffffff,%ecx
8049db2:   b8 00 00 00 00         mov    $0x0,%eax
8049db7:   f2 ae                  repnz scas %es:(%edi),%al
8049db9:   89 cb                  mov    %ecx,%ebx
8049dbb:   f7 d3                  not    %ebx
8049dbd:   8d 7b ff               lea    -0x1(%ebx),%edi
8049dc0:   83 c4 20               add    $0x20,%esp
```

```
 8049dc3:   89 fb                mov    %edi,%ebx
 8049dc5:   8d b5 e4 7f ff ff      lea    -0x801c(%ebp),%esi
 8049dcb:   85 ff                test   %edi,%edi
 8049dcd:   0f 84 17 fc ff ff      je     80499ea <submitr+0x430>
 8049dd3:   89 bd b4 5f ff ff      mov    %edi,-0xa04c(%ebp)
 8049dd9:   8b bd b0 5f ff ff      mov    -0xa050(%ebp),%edi
 8049ddf:   e9 d1 fb ff ff        jmp    80499b5 <submitr+0x3fb>
 8049de4:   e8 a7 e9 ff ff        call   8048790 <__stack_chk_fail@plt>

08049de9 <init_timeout>:
 8049de9:   55                   push   %ebp
 8049dea:   89 e5                mov    %esp,%ebp
 8049dec:   53                   push   %ebx
 8049ded:   83 ec 04              sub    $0x4,%esp
 8049df0:   8b 5d 08              mov    0x8(%ebp),%ebx
 8049df3:   85 db                test   %ebx,%ebx
 8049df5:   74 24                je     8049e1b <init_timeout+0x32>
 8049df7:   83 ec 08              sub    $0x8,%esp
 8049dfa:   68 d6 94 04 08        push   $0x80494d6
 8049dff:   6a 0e                push   $0xe
 8049e01:   e8 5a e9 ff ff        call   8048760 <signal@plt>
 8049e06:   85 db                test   %ebx,%ebx
 8049e08:   b8 00 00 00 00        mov    $0x0,%eax
 8049e0d:   0f 48 d8              cmovs  %eax,%ebx
 8049e10:   89 1c 24              mov    %ebx,(%esp)
 8049e13:   e8 68 e9 ff ff        call   8048780 <alarm@plt>
 8049e18:   83 c4 10              add    $0x10,%esp
 8049e1b:   8b 5d fc              mov    -0x4(%ebp),%ebx
 8049e1e:   c9                   leave
 8049e1f:   c3                   ret

08049e20 <init_driver>:
 8049e20:   55                   push   %ebp
 8049e21:   89 e5                mov    %esp,%ebp
 8049e23:   57                   push   %edi
 8049e24:   56                   push   %esi
 8049e25:   53                   push   %ebx
 8049e26:   83 ec 34              sub    $0x34,%esp
 8049e29:   8b 75 08              mov    0x8(%ebp),%esi
 8049e2c:   65 a1 14 00 00 00      mov    %gs:0x14,%eax
 8049e32:   89 45 e4              mov    %eax,-0x1c(%ebp)
 8049e35:   31 c0                xor    %eax,%eax
 8049e37:   6a 01                push   $0x1
 8049e39:   6a 0d                push   $0xd
```

```
8049e3b:   e8 20 e9 ff ff        call   8048760 <signal@plt>
8049e40:   83 c4 08              add    $0x8,%esp
8049e43:   6a 01                 push   $0x1
8049e45:   6a 1d                 push   $0x1d
8049e47:   e8 14 e9 ff ff        call   8048760 <signal@plt>
8049e4c:   83 c4 08              add    $0x8,%esp
8049e4f:   6a 01                 push   $0x1
8049e51:   6a 1d                 push   $0x1d
8049e53:   e8 08 e9 ff ff        call   8048760 <signal@plt>
8049e58:   83 c4 0c              add    $0xc,%esp
8049e5b:   6a 00                 push   $0x0
8049e5d:   6a 01                 push   $0x1
8049e5f:   6a 02                 push   $0x2
8049e61:   e8 ea e9 ff ff        call   8048850 <socket@plt>
8049e66:   83 c4 10              add    $0x10,%esp
8049e69:   85 c0                 test   %eax,%eax
8049e6b:   0f 88 a0 00 00 00     js     8049f11 <init_driver+0xf1>
8049e71:   89 c3                 mov    %eax,%ebx
8049e73:   83 ec 0c              sub    $0xc,%esp
8049e76:   68 b6 a6 04 08        push   $0x804a6b6
8049e7b:   e8 f0 e9 ff ff        call   8048870 <gethostbyname@plt>
8049e80:   83 c4 10              add    $0x10,%esp
8049e83:   85 c0                 test   %eax,%eax
8049e85:   0f 84 d1 00 00 00     je     8049f5c <init_driver+0x13c>
8049e8b:   8d 7d d4              lea    -0x2c(%ebp),%edi
8049e8e:   c7 45 d6 00 00 00 00    movl   $0x0,-0x2a(%ebp)
8049e95:   c7 45 da 00 00 00 00    movl   $0x0,-0x26(%ebp)
8049e9c:   c7 45 de 00 00 00 00    movl   $0x0,-0x22(%ebp)
8049ea3:   66 c7 45 e2 00 00     movw   $0x0,-0x1e(%ebp)
8049ea9:   66 c7 45 d4 02 00     movw   $0x2,-0x2c(%ebp)
8049eaf:   6a 0c                 push   $0xc
8049eb1:   ff 70 0c              pushl  0xc(%eax)
8049eb4:   8b 40 10              mov    0x10(%eax),%eax
8049eb7:   ff 30                 pushl  (%eax)
8049eb9:   8d 45 d8              lea    -0x28(%ebp),%eax
8049ebc:   50                    push   %eax
8049ebd:   e8 0e e9 ff ff        call   80487d0 <__memmove_chk@plt>
8049ec2:   66 c7 45 d6 22 b9     movw   $0xb922,-0x2a(%ebp)
8049ec8:   83 c4 0c              add    $0xc,%esp
8049ecb:   6a 10                 push   $0x10
8049ecd:   57                    push   %edi
8049ece:   53                    push   %ebx
8049ecf:   e8 bc e9 ff ff        call   8048890 <connect@plt>
8049ed4:   83 c4 10              add    $0x10,%esp
```

```
8049ed7:   85 c0                test   %eax,%eax
8049ed9:   0f 88 e9 00 00 00      js     8049fc8 <init_driver+0x1a8>
8049edf:   83 ec 0c               sub    $0xc,%esp
8049ee2:   53                     push   %ebx
8049ee3:   e8 b8 e9 ff ff         call   80488a0 <close@plt>
8049ee8:   66 c7 06 4f 4b         movw   $0x4b4f,(%esi)
8049eed:   c6 46 02 00            movb   $0x0,0x2(%esi)
8049ef1:   83 c4 10               add    $0x10,%esp
8049ef4:   b8 00 00 00 00         mov    $0x0,%eax
8049ef9:   8b 55 e4               mov    -0x1c(%ebp),%edx
8049efc:   65 33 15 14 00 00 00   xor    %gs:0x14,%edx
8049f03:   0f 85 ec 00 00 00      jne    8049ff5 <init_driver+0x1d5>
8049f09:   8d 65 f4               lea    -0xc(%ebp),%esp
8049f0c:   5b                     pop    %ebx
8049f0d:   5e                     pop    %esi
8049f0e:   5f                     pop    %edi
8049f0f:   5d                     pop    %ebp
8049f10:   c3                     ret
8049f11:   c7 06 45 72 72 6f      movl   $0x6f727245,(%esi)
8049f17:   c7 46 04 72 3a 20 43   movl   $0x43203a72,0x4(%esi)
8049f1e:   c7 46 08 6c 69 65 6e   movl   $0x6e65696c,0x8(%esi)
8049f25:   c7 46 0c 74 20 75 6e   movl   $0x6e752074,0xc(%esi)
8049f2c:   c7 46 10 61 62 6c 65   movl   $0x656c6261,0x10(%esi)
8049f33:   c7 46 14 20 74 6f 20   movl   $0x206f7420,0x14(%esi)
8049f3a:   c7 46 18 63 72 65 61   movl   $0x61657263,0x18(%esi)
8049f41:   c7 46 1c 74 65 20 73   movl   $0x73206574,0x1c(%esi)
8049f48:   c7 46 20 6f 63 6b 65   movl   $0x656b636f,0x20(%esi)
8049f4f:   66 c7 46 24 74 00      movw   $0x74,0x24(%esi)
8049f55:   b8 ff ff ff ff         mov    $0xffffffff,%eax
8049f5a:   eb 9d                  jmp    8049ef9 <init_driver+0xd9>
8049f5c:   c7 06 45 72 72 6f      movl   $0x6f727245,(%esi)
8049f62:   c7 46 04 72 3a 20 44   movl   $0x44203a72,0x4(%esi)
8049f69:   c7 46 08 4e 53 20 69   movl   $0x6920534e,0x8(%esi)
8049f70:   c7 46 0c 73 20 75 6e   movl   $0x6e752073,0xc(%esi)
8049f77:   c7 46 10 61 62 6c 65   movl   $0x656c6261,0x10(%esi)
8049f7e:   c7 46 14 20 74 6f 20   movl   $0x206f7420,0x14(%esi)
8049f85:   c7 46 18 72 65 73 6f   movl   $0x6f736572,0x18(%esi)
8049f8c:   c7 46 1c 6c 76 65 20   movl   $0x2065766c,0x1c(%esi)
8049f93:   c7 46 20 73 65 72 76   movl   $0x76726573,0x20(%esi)
8049f9a:   c7 46 24 65 72 20 61   movl   $0x61207265,0x24(%esi)
8049fa1:   c7 46 28 64 64 72 65   movl   $0x65726464,0x28(%esi)
8049fa8:   66 c7 46 2c 73 73      movw   $0x7373,0x2c(%esi)
8049fae:   c6 46 2e 00            movb   $0x0,0x2e(%esi)
8049fb2:   83 ec 0c               sub    $0xc,%esp
```

```
 8049fb5:   53                   push   %ebx
 8049fb6:   e8 e5 e8 ff ff        call   80488a0 <close@plt>
 8049fbb:   83 c4 10              add    $0x10,%esp
 8049fbe:   b8 ff ff ff ff        mov    $0xffffffff,%eax
 8049fc3:   e9 31 ff ff ff        jmp    8049ef9 <init_driver+0xd9>
 8049fc8:   83 ec 0c              sub    $0xc,%esp
 8049fcb:   68 b6 a6 04 08        push   $0x804a6b6
 8049fd0:   68 70 a6 04 08        push   $0x804a670
 8049fd5:   6a ff                 push   $0xffffffff
 8049fd7:   6a 01                 push   $0x1
 8049fd9:   56                   push   %esi
 8049fda:   e8 e1 e8 ff ff        call   80488c0 <__sprintf_chk@plt>
 8049fdf:   83 c4 14              add    $0x14,%esp
 8049fe2:   53                   push   %ebx
 8049fe3:   e8 b8 e8 ff ff        call   80488a0 <close@plt>
 8049fe8:   83 c4 10              add    $0x10,%esp
 8049feb:   b8 ff ff ff ff        mov    $0xffffffff,%eax
 8049ff0:   e9 04 ff ff ff        jmp    8049ef9 <init_driver+0xd9>
 8049ff5:   e8 96 e7 ff ff        call   8048790 <__stack_chk_fail@plt>

08049ffa <driver_post>:
 8049ffa:   55                   push   %ebp
 8049ffb:   89 e5                mov    %esp,%ebp
 8049ffd:   53                   push   %ebx
 8049ffe:   83 ec 04              sub    $0x4,%esp
 804a001:   8b 55 08              mov    0x8(%ebp),%edx
 804a004:   8b 45 10              mov    0x10(%ebp),%eax
 804a007:   8b 5d 14              mov    0x14(%ebp),%ebx
 804a00a:   85 c0                 test   %eax,%eax
 804a00c:   75 17                 jne    804a025 <driver_post+0x2b>
 804a00e:   85 d2                 test   %edx,%edx
 804a010:   74 05                 je     804a017 <driver_post+0x1d>
 804a012:   80 3a 00              cmpb   $0x0,(%edx)
 804a015:   75 33                 jne    804a04a <driver_post+0x50>
 804a017:   66 c7 03 4f 4b         movw   $0x4b4f,(%ebx)
 804a01c:   c6 43 02 00           movb   $0x0,0x2(%ebx)
 804a020:   8b 5d fc              mov    -0x4(%ebp),%ebx
 804a023:   c9                   leave
 804a024:   c3                   ret
 804a025:   83 ec 04              sub    $0x4,%esp
 804a028:   ff 75 0c              pushl  0xc(%ebp)
 804a02b:   68 c5 a6 04 08        push   $0x804a6c5
 804a030:   6a 01                 push   $0x1
 804a032:   e8 09 e8 ff ff        call   8048840 <__printf_chk@plt>
```

```
 804a037:   66 c7 03 4f 4b          movw   $0x4b4f,(%ebx)
 804a03c:   c6 43 02 00             movb   $0x0,0x2(%ebx)
 804a040:   83 c4 10                add    $0x10,%esp
 804a043:   b8 00 00 00 00          mov    $0x0,%eax
 804a048:   eb d6                   jmp    804a020 <driver_post+0x26>
 804a04a:   83 ec 04                sub    $0x4,%esp
 804a04d:   53                      push   %ebx
 804a04e:   ff 75 0c                pushl  0xc(%ebp)
 804a051:   68 dc a6 04 08          push   $0x804a6dc
 804a056:   52                      push   %edx
 804a057:   68 f3 a6 04 08          push   $0x804a6f3
 804a05c:   68 b9 22 00 00          push   $0x22b9
 804a061:   68 b6 a6 04 08          push   $0x804a6b6
 804a066:   e8 4f f5 ff ff          call   80495ba <submitr>
 804a06b:   83 c4 20                add    $0x20,%esp
 804a06e:   eb b0                   jmp    804a020 <driver_post+0x26>

0804a070 <__libc_csu_init>:
 804a070:   55                      push   %ebp
 804a071:   57                      push   %edi
 804a072:   56                      push   %esi
 804a073:   53                      push   %ebx
 804a074:   e8 b7 e8 ff ff          call   8048930 <__x86.get_pc_thunk.bx>
 804a079:   81 c3 87 1f 00 00       add    $0x1f87,%ebx
 804a07f:   83 ec 0c                sub    $0xc,%esp
 804a082:   8b 6c 24 28             mov    0x28(%esp),%ebp
 804a086:   8d b3 10 ff ff ff       lea    -0xf0(%ebx),%esi
 804a08c:   e8 63 e6 ff ff          call   80486f4 <_init>
 804a091:   8d 83 0c ff ff ff       lea    -0xf4(%ebx),%eax
 804a097:   29 c6                   sub    %eax,%esi
 804a099:   c1 fe 02                sar    $0x2,%esi
 804a09c:   85 f6                   test   %esi,%esi
 804a09e:   74 25                   je     804a0c5 <__libc_csu_init+0x55>
 804a0a0:   31 ff                   xor    %edi,%edi
 804a0a2:   8d b6 00 00 00 00       lea    0x0(%esi),%esi
 804a0a8:   83 ec 04                sub    $0x4,%esp
 804a0ab:   55                      push   %ebp
 804a0ac:   ff 74 24 2c             pushl  0x2c(%esp)
 804a0b0:   ff 74 24 2c             pushl  0x2c(%esp)
 804a0b4:   ff 94 bb 0c ff ff ff    call   *-0xf4(%ebx,%edi,4)
 804a0bb:   83 c7 01                add    $0x1,%edi
 804a0be:   83 c4 10                add    $0x10,%esp
 804a0c1:   39 fe                   cmp    %edi,%esi
 804a0c3:   75 e3                   jne    804a0a8 <__libc_csu_init+0x38>
```

```
 804a0c5:   83 c4 0c             add    $0xc,%esp
 804a0c8:   5b                   pop    %ebx
 804a0c9:   5e                   pop    %esi
 804a0ca:   5f                   pop    %edi
 804a0cb:   5d                   pop    %ebp
 804a0cc:   c3                   ret
 804a0cd:   8d 76 00             lea    0x0(%esi),%esi

0804a0d0 <__libc_csu_fini>:
 804a0d0:   f3 c3                repz ret

Disassembly of section .fini:

0804a0d4 <_fini>:
 804a0d4:   53                   push   %ebx
 804a0d5:   83 ec 08             sub    $0x8,%esp
 804a0d8:   e8 53 e8 ff ff       call   8048930 <__x86.get_pc_thunk.bx>
 804a0dd:   81 c3 23 1f 00 00      add    $0x1f23,%ebx
 804a0e3:   83 c4 08             add    $0x8,%esp
 804a0e6:   5b                   pop    %ebx
 804a0e7:   c3                   ret
```