

Reverse02

1 找出生命值变量的地址

DosBoxDeb打开, mount后先运行gb



再运行游戏

进入游戏后先打开Address Analysis, 搜索1, 死一次搜索2, 再死一次搜索3, 然后查看一下搜出来的地址



这时剩的地址还比较多, 把最后一条命也用掉, 再来一次, 进入游戏先搜1



地址变少了, 但还不够少, 死一次搜2



左右跑一跑, 再搜2



再死一次, 搜3



左右跑一跑, 再搜3, 地址没有减少, 把最后一条命用掉, 重新开始游戏搜1, 地址还是没有减少, 死一次搜2, 这时地址已经只剩4条了



再死一次搜3, 还剩2条可疑地址



把第一条地址放到Game Table里, 把剩下的命用掉, 重新开始游戏, 在Game Table里把 1000:9966 锁住



返回游戏, 再死一次, 发现生命值虽然减少了, 但动一动又马上变回3

说明 1000:9966 应该就是生命值变量的地址

2 找到掉血指令并计算PSP偏移

alt-pause进入debug模式, 输入 bpm 1000:9966 设置硬件断点

```

DOSBox Debugger
---<Register Overview>---
EAX=00000002  ESI=00000000  DS=145F  ES=145F  FS=0000  GS=0000  SS=145F Real
EBX=0000AD08  EDI=0000938E  CS=0487  EIP=00000270  C1 Z0 S1 00 A1 P1 D0 I1 T0
ECK=00000000  EBP=00000025                                IOPL3  CPL0
EDX=000003C4  ESP=00008E52                                3366292147
---<Data Overview Scroll: page up/down>---
0000:0000  C6 0F 87 04 08 00 70 00 02 00 08 00 1F 08 25 02  .....p.....%.
0000:0010  08 00 70 00 60 10 00 F0 60 10 00 F0 60 10 00 F0  ..p. ....
0000:0020  2A 51 87 04 E0 10 87 04 55 FF 00 F0 60 10 00 F0  *Q.....U.....
0000:0030  60 10 00 F0 60 10 00 F0 80 10 00 F0 60 10 00 F0  .....
0000:0040  AE 4F 87 04 20 11 00 F0 40 11 00 F0 E1 09 25 02  .0. ....e.....%.
0000:0050  C0 11 00 F0 E0 11 00 F0 00 12 00 F0 40 12 00 F0  .....e...
0000:0060  E0 12 00 F0 E0 12 00 F0 60 12 00 F0 60 10 00 F0  .....
0000:0070  5E 0A 25 02 A4 F0 00 F0 60 10 00 F0 00 05 00 C0  ^.%.....
---<Code Overview Scroll: up/down>---
0487:0270  72F7          jc  00000269 <$-9>          <up>
0487:0272  A00500        mov al,[0005]             ds:[0005]=0000
0487:0275  80260500EF    and byte [0005],EF       ds:[0005]=0000
0487:027A  A810          test al,10
0487:027C  74DA          je  00000258 <$-26>        <no jmp>
0487:027E  800E050080    or  byte [0005],80       ds:[0005]=0000
0487:0283  8B1EE4AC      mov bx,[ACE4]            ds:[ACE4]=5364
0487:0287  837F1200      cmp word [bx+12],0000     ds:[AD1A]=AD99
0487:028B  7475          je  00000302 <$+75>        <no jmp>
0487:028D  E891D5        call FFFFD821 <$-2a6f>
-> bpm 1000:9966_
---<Variable Overview>---
---<OutPut/Input Scroll: home/end>---
3366231526: PIC:0 mask F8
3366272637: PIT:PIT 0 Timer at 18.2068 Hz mode 3
3366273782: PIT:PIT 0 Timer at 73.0088 Hz mode 3
3366273785: PIC:0 mask F8
***! TYPE HELP (+ENTER) TO GET AN OVERVIEW OF ALL COMMANDS !***

```

回车

```

---<OutPut/Input Scroll: home/end>---
3366272637: PIT:PIT 0 Timer at 18.2068 Hz mode 3
3366273782: PIT:PIT 0 Timer at 73.0088 Hz mode 3
3366273785: PIC:0 mask F8
***! TYPE HELP (+ENTER) TO GET AN OVERVIEW OF ALL COMMANDS !***
DEBUG: Set memory breakpoint at 1000:9966

```

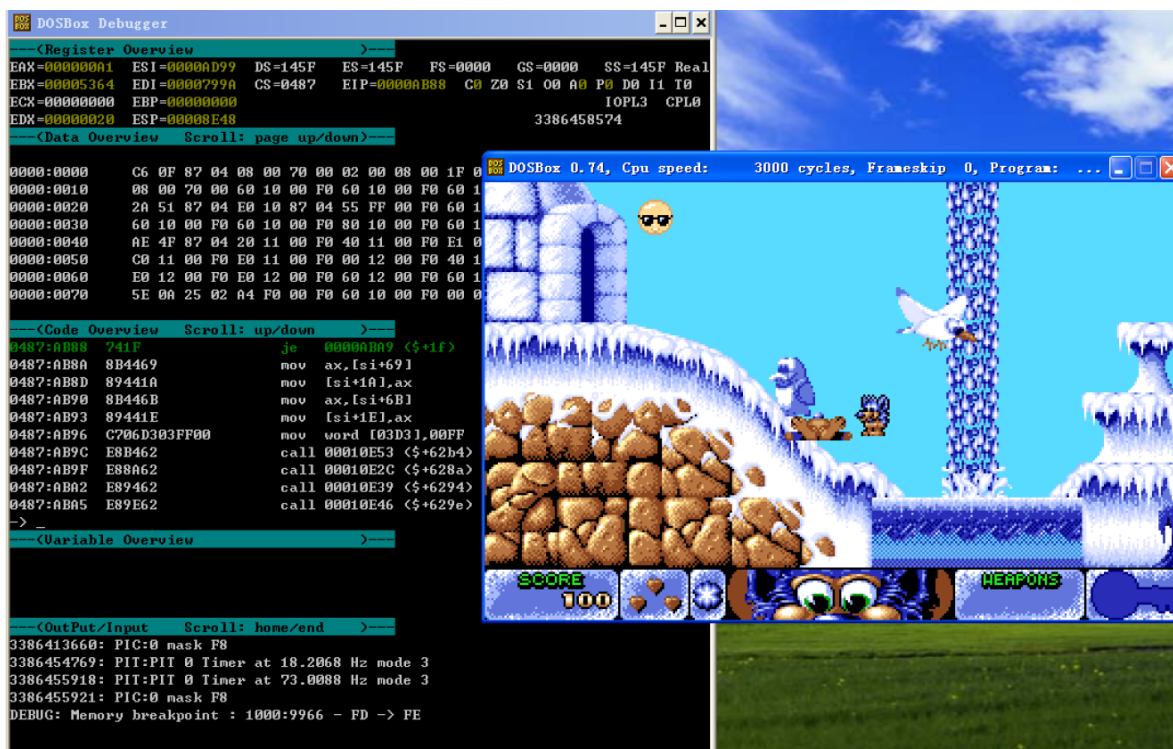
F5继续运行, 首次断点触发

```

---<OutPut/Input Scroll: home/end>---
3366273782: PIT:PIT 0 Timer at 73.0088 Hz mode 3
3366273785: PIC:0 mask F8
***! TYPE HELP (+ENTER) TO GET AN OVERVIEW OF ALL COMMANDS !***
DEBUG: Set memory breakpoint at 1000:9966
DEBUG: Memory breakpoint : 1000:9966 - 00 -> FD

```

忽略, F5继续运行, 死一次, 触发断点



向上翻查看断点触发处周围的指令

```

---(Code Overview Scroll: up/down)---
0487:AB81 8B1EE4AC      mov     bx,[ACE4]          ds:[ACE4]=5364
0487:AB85 FF4712      inc     word [bx+12]       ds:[5376]=FFFE
0487:AB88 741F      je      0000ABA9 <$+1f>   <no jmp>
0487:AB8A 8B4469      mov     ax,[si+69]        ds:[AE02]=00A7
0487:AB8D 89441A      mov     [si+1A],ax        ds:[ADB3]=00A7
0487:AB90 8B446B      mov     ax,[si+6B]        ds:[AE04]=02B7
0487:AB93 89441E      mov     [si+1E],ax        ds:[ADB7]=02B6
0487:AB96 C706D303FF00    mov     word [03D3],00FF  ds:[03D3]=00FF
0487:AB9C E8B462      call    00010E53 <$+62b4>
0487:AB9F E88A62      call    00010E2C <$+628a>
-> _

```

可以看到, 触发断点(掉血)的指令为:

1 | 0487:AB85 FF4712 inc word [bx+12]

这里掉血指令用的是 inc, 也就是生命值减少实际上是表示生命值的变量值增加, 与之前Address Analysis时生命值递减, 但搜索值增加的逻辑相符

dos mcbs 查看当前PSP段址

```

---(OutPut/Input Scroll: home/end)---
3386458574: MISC: 40D0      54880      0477      FIRE
3386458574: MISC: 4E37      334960     0477
3386458574: MISC:Upper memory:
3386458574: MISC: 9FFF      196608     0008 <DOS>   SC
3386458574: MISC: D000      65520      0000 <free>

```

当前PSP的段址为 0477h

```

---(Register Overview)---
EAX=00000005 ESI=00000000 DS=145F ES=145F FS=0000 GS=0000 SS=145F Real
EBX=0000AD08 EDI=0000938E CS=0487 EIP=0000026C C1 Z0 S1 00 A1 P0 D0 I1 T0
ECX=00000000 EBP=0000000F                                IOPL3 CPL0
EDX=000003C4 ESP=00008E52                                3404987503

```

当前CS为 0487h

段址的差: CS - PSP = 10h

要修改的指令的地址 = (PSP+10h):AB85h

3 编写Buster驻留程序

写汇编代码, 构造新的int8h中断, 将游戏中掉血的那条指令废除

```
1  ;myfire.asm
2  code segment
3  assume cs:code, ds:code
4  ;my int_8h
5  int_8h:
6      cmp cs:fixed, 1
7      je goto_old_8h
8      push ax
9      push bx
10     push cx
11     push dx
12     push si
13     push di
14     push ds
15     push es
16     mov ah, 62h
17     int 21h      ;BX=psp
18     add bx, 10h
19     mov ds, bx
20
21     cmp byte ptr ds:[0AB85h], 0FFh
22     jne skip
23     cmp byte ptr ds:[0AB86h], 47h
24     jne skip
25     cmp byte ptr ds:[0AB87h], 12h
26     jne skip
27     mov byte ptr ds:[0AB85h], 90h
28     mov byte ptr ds:[0AB86h], 90h
29     mov byte ptr ds:[0AB87h], 90h
30
31     mov cs:fixed, 1
32 skip:
33     pop es
34     pop ds
35     pop di
36     pop si
37     pop dx
38     pop cx
39     pop bx
40     pop ax
41 goto_old_8h:
42     jmp dword ptr cs:[old_8h]
43 fixed db 0
44 old_8h dw 0,0      ;old vector of int_8h
45 ;End of int_8h
46
47 initialize:
48     push cs
49     pop ds          ;DS=CS
50     xor ax, ax
```



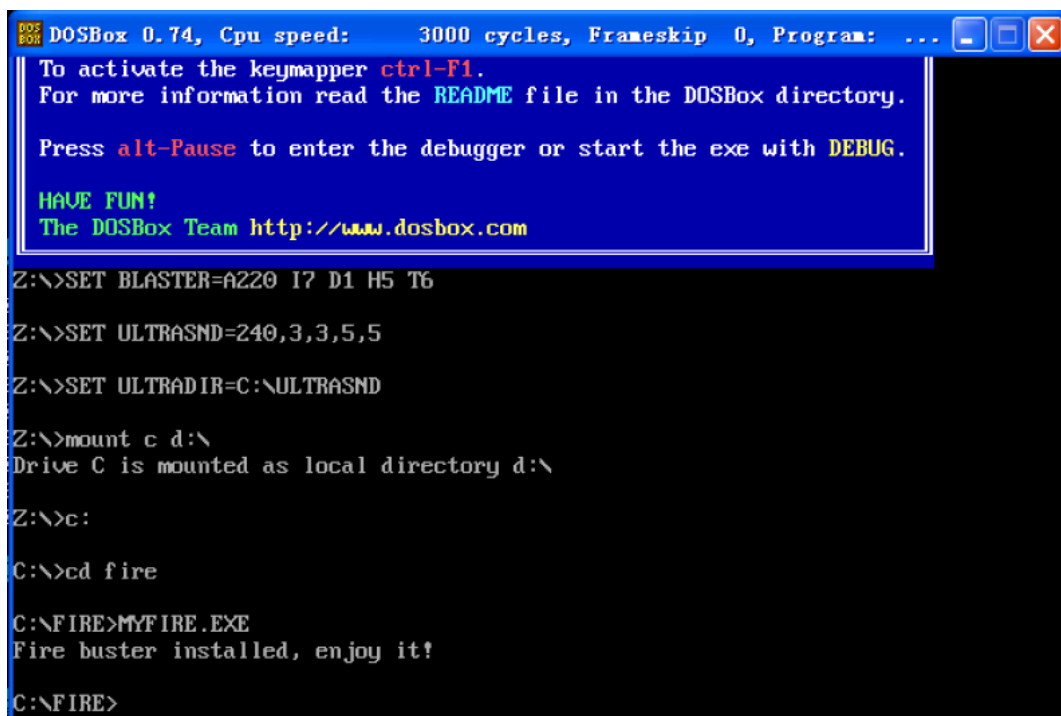
```

51     mov es, ax
52     mov bx, 8*4      ;ES:BX-> int_8h's vector
53     push es:[bx]
54     pop old_8h[0]
55     push es:[bx+2]
56     pop old_8h[2]    ;save old vector of int_8h
57     mov ax, offset int_8h
58     cli              ;disable interrupt when changing int_8h's vector
59     push ax
60     pop es:[bx]
61     push cs
62     pop es:[bx+2]    ;set vector of int_8h
63     sti              ;enable interrupt
64 install:
65     mov ah, 9
66     mov dx, offset install_msg
67     int 21h
68
69     mov dx, offset initialize    ;DX=len before label initialize
70     add dx, 100h                ;include PSP's len
71     add dx, 0Fh                 ;include remnant bytes
72     mov cl, 4
73     shr dx, cl                  ;DX=program's paragraph size to keep resident
74     mov ah, 31h
75     int 21h                     ;keep resident
76 install_msg db 'Fire buster installed, enjoy it!', 0Dh, 0Ah, '$'
77 code ends
78 end initialize

```

4 验证驻留程序的效果

先运行myfire.exe (由myfire.asm汇编链接而来)



```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

Press alt-Pause to enter the debugger or start the exe with DEBUG.

HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6
Z:\>SET ULTRASND=240,3,3,5,5
Z:\>SET ULTRADIR=C:\ULTRASND
Z:\>mount c d:\
Drive C is mounted as local directory d:\
Z:\>c:
C:\>cd fire
C:\FIRE>MYFIRE.EXE
Fire buster installed, enjoy it!
C:\FIRE>

```

再运行游戏, 死一次



但是生命值并没有减少



完成第一关



5 总结

这次作业主要内容是对DosBoxDeb的使用以及驻留程序的编写, 难点在于找出生命值变量的地址, 因为游戏中的生命值变量的值并不是简单的实际生命值, 而是实际生命值的相反数, 因此地址分析的时候要按照递增的顺序搜.

此外, 由于游戏中初始生命值有些少, 生命值用完后也可能还是不足以定位生命值变量的地址, 这时需要在中断分析过程的情况下多重复几次, 而且, 搜索的值并不一定要发生变化, 也可以做一些不影响生命值的操作然后搜索相同的值, 比如左右移动吃点金币后再次搜索上一次搜索的值.