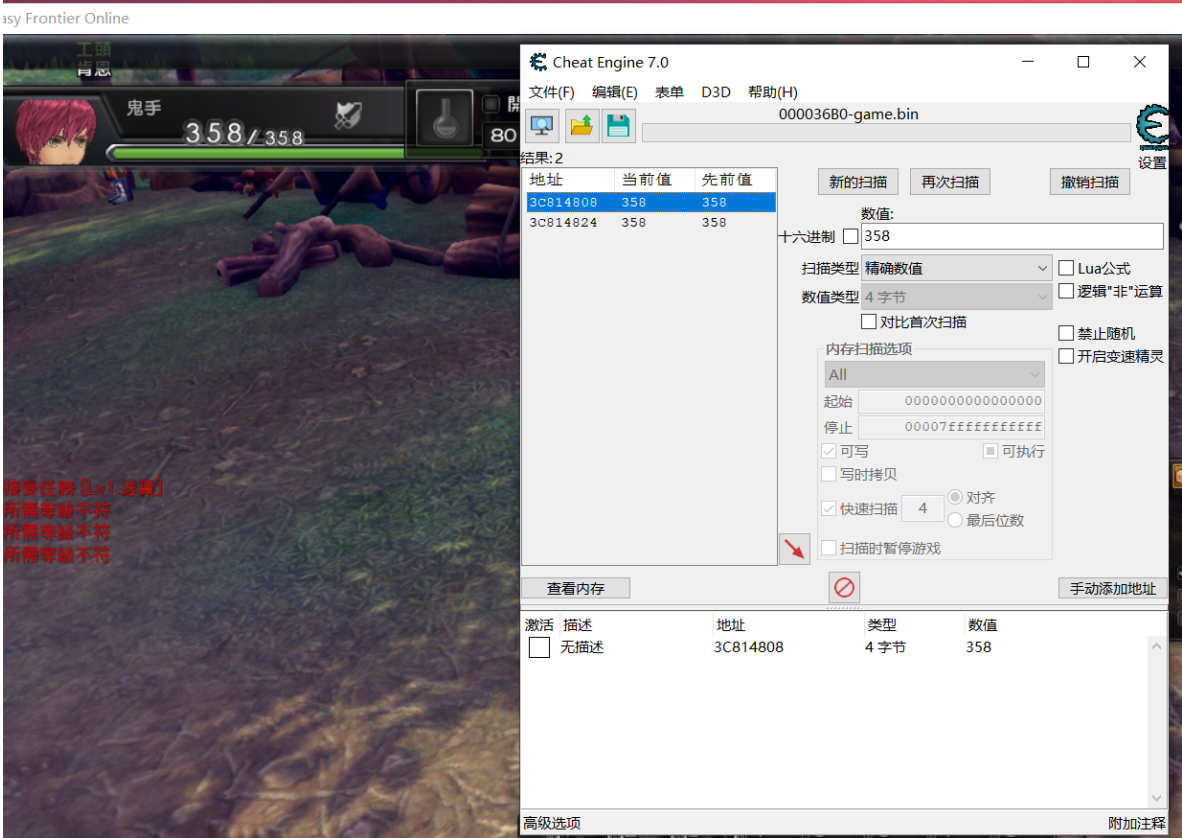


之前我们已经通过人物的血量找到了人物的属性数组，接下来学习一下链表。

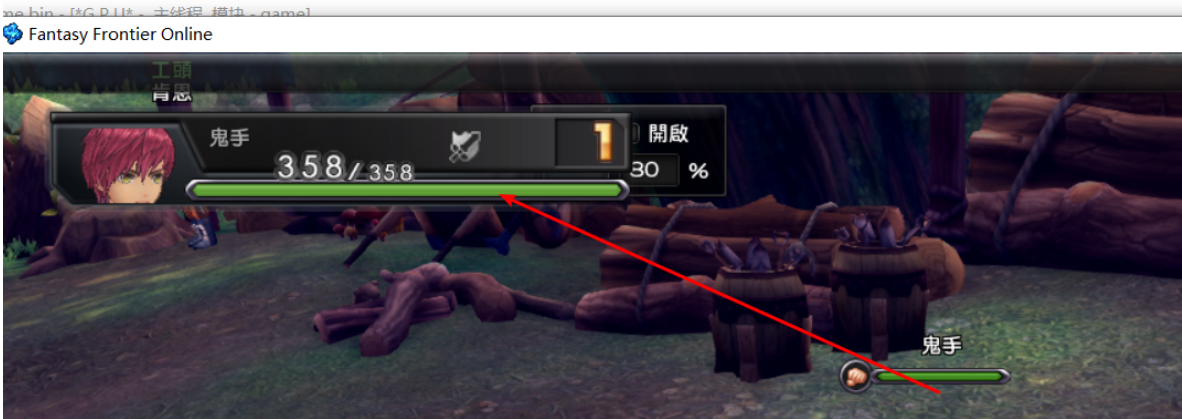
逆向周围对象链表

这一次要找的目标是人物周围的对象链表，包括人物周围的NPC和怪物等等。找这个数据结构的突破口有很多，可以通过人物本身，也可以通过NPC和怪物。

这里还是以人物血量为突破口，之前我们已经用人物血量去找到了属性的数组，但是同一个突破口通过不同的访问代码可以逆向分析出不同的数据。



和之前的方式一样，用CE找到当前的人物血量，然后在这个位置下硬件访问断点



点击当前的人物血量

OllyICE - game.bin - [*G.P.U* - 主线程, 模块 - game]

文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H) 工具 设置API断点>

暂停

地址	HEX 数据	反汇编	注释
00669A08	74 56	je short 00669A60	
00669A0A	68 C2FEE000	push 00E0FEC2	
00669A0F	8D4D D8	lea ecx, dword ptr [ebp-28]	
00669A12	E8 79AFDAFF	call 00414990	
00669A17	8B46 0C	mov eax, dword ptr [esi+C]	
00669A1A	DB40 08	fild dword ptr [eax+8]	
00669A1D	53	push ebx	
00669A1E	51	push ecx	
00669A1F	8B4E 10	mov ecx, dword ptr [esi+10]	
00669A22	DA70 24	fdiv dword ptr [eax+24]	
00669A25	8D55 D8	lea edx, dword ptr [ebp-28]	
00669A28	C745 FC 010000	mov dword ptr [ebp-4], 1	
00669A2F	D91C24	fstp dword ptr [esp]	
00669A32	52	push edx	
00669A33	57	push edi	
00669A34	E8 37BE0600	call 006D5870	

ebx=FFFFFFFF

地址	数值	注释
3C814808	00000166	

地址 00191E

断点断下, [eax+8]是人物的血量, 追eax的数据来源, eax来自[esi+0xC], 继续往上追esi

血量=[esi+0xC]+8

OllyICE - game.bin - [*G.P.U* - 主线程, 模块 - game]

文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H) 工具 设置API断点>

暂停

地址	HEX 数据	反汇编	注释
006698E0	50	push eax	
006698E1	64:8925 00000000	mov dword ptr fs:[0], esp	
006698E8	83EC 38	sub esp, 38	
006698EB	56	push esi	
006698EC	8BF1	mov esi, ecx	
006698EE	80BE 68020000	cmp byte ptr [esi+268], 0	
006698F5	57	push edi	
006698F6	74 0C	je short 00669904	
006698F8	E8 435F0600	call 006CF840	
006698FD	C686 68020000	mov byte ptr [esi+268], 0	
00669904	8BCE	mov ecx, esi	
00669906	E8 C54C0600	call 006CE5D0	
0066990B	833D 744BF800	cmp dword ptr [F84B74], 0	
00669912	75 05	jnz short 00669919	

esi来自于ecx, 返回上层找ecx

地址	HEX 数据	反汇编	注释
006EC0B2	898F A8010000	mov dword ptr [edi+1A8], ecx	
006EC0B8	8B50 08	mov edx, dword ptr [eax+8]	
006EC0BB	8997 AC010000	mov dword ptr [edi+1AC], edx	
006EC0C1	8B76 08	mov esi, dword ptr [esi+8]	
006EC0C4	83C6 30	add esi, 30	
006EC0C7	B9 09000000	mov ecx, 9	
006EC0CC	81C7 B0010000	add edi, 1B0	
006EC0D2	F3:A5	rep movs dword ptr es:[edi], dword ptr [esi]	
006EC0D4	8BCB	mov ecx, ebx	
006EC0D6	E8 F5D7F7FF	call 006698D0	
006EC0DB	8BCB	mov ecx, ebx	
006EC0DD	E8 1EDEFFFF	call 006E9F00	
006EC0E2	83BB E4010000	cmp dword ptr [ebx+1E4], 0	
006EC0E9	5F	pop edi	

ecx又来源于ebx

血量=[ebx+0xC]+8

址	HEX 数据	反汇编	注释
5EBF1E	CC	int3	
5EBF1F	CC	int3	
5EBF20	55	push ebp	
5EBF21	8BEC	mov ebp, esp	
5EBF23	83EC 10	sub esp, 10	
5EBF26	53	push ebx	
5EBF27	8BD9	mov ebx, ecx	
5EBF29	8B4B 2C	mov ecx, dword ptr [ebx+2C]	
5EBF2C	8379 5C FF	cmp dword ptr [ecx+5C], -1	
5EBF30	74 2B	je short 006EBF5D	
5EBF32	D981 C0000000	fld dword ptr [ecx+C0]	
5EBF38	D9EE	fldz	
5EBF3A	DDE1	fucomp st(1)	
5EBF3C	DEFA	setcm	

ebx来源ecx

地址	HEX 数据	反汇编	注释
006D7D4E	74 22	je short 006D7D72	
006D7D50	3B77 04	cmp esi, dword ptr [edi+4]	
006D7D53	75 05	jnz short 006D7D5A	
006D7D55	E8 D126D4FF	call 0041A42B	
006D7D5A	8B4E 0C	mov ecx, dword ptr [esi+C]	
006D7D5D	8B11	mov edx, dword ptr [ecx]	
006D7D5F	8B42 08	mov eax, dword ptr [edx+8]	
006D7D62	FFD0	call eax	
006D7D64	3B77 04	cmp esi, dword ptr [edi+4]	
006D7D67	75 05	jnz short 006D7D6E	
006D7D69	E8 BD26D4FF	call 0041A42B	
006D7D6E	8B36	mov esi, dword ptr [esi]	

ecx来自[esi+C]

血量=[[esi+C]+0xC]+8

OllyICE - game.bin - [*G.P.U* - m主线程, 模块 - game]

暂停

地址	HEX 数据	反汇编	注释
006D7D4E	74 22	je short 006D7D72	
006D7D50	3B77 04	cmp esi, dword ptr [edi+4]	
006D7D53	75 05	jnz short 006D7D5A	
006D7D55	E8 D126D4FF	call 0041A42B	
006D7D5A	8B4E 0C	mov ecx, dword ptr [esi+C]	血量=[[esi+C]+0xC]+8
006D7D5D	8B11	mov edx, dword ptr [ecx]	
006D7D5F	8B42 08	mov eax, dword ptr [edx+8]	
006D7D62	FFD0	call eax	
006D7D64	3B77 04	cmp esi, dword ptr [edi+4]	
006D7D67	75 05	jnz short 006D7D6E	
006D7D69	E8 BD26D4FF	call 0041A42B	
006D7D6E	8B36	mov esi, dword ptr [esi]	
006D7D70	EB CE	jmp short 006D7D40	
006D7D72	5F	pop edi	
006D7D73	5E	pop esi	
006D7D74	5B	pop ebx	

ds:[0E5C8FCC]=296F4980
esi=0019FD74

地址	数值	注释
296F5888	0000008C	
296F588C	00000000	
296F5890	00000003	
296F5894	40000000	
296F5898	00000000	
296F589C	00000000	
296F58A0	00000000	
296F58A4	0000008C	
296F58A8	00000000	
296F58AC	00000000	
296F58B0	00000000	
296F58B4	00000000	
296F58B8	00000000	
296F58BC	40466667	
296F58C0	00000000	
296F58C4	00000005	
296F58C8	00000000	

地址	数值	注释
0019FD38	15734720	
0019FD3C	14F24C00	
0019FD40	00000000	
0019FD44	006D7E18	返回到 game.
0019FD48	06B10240	
0019FD4C	0067483C	返回到 game.
0019FD50	77039570	user32.Trans
0019FD54	069980B0	
0019FD58	00000000	
0019FD5C	00000000	
0019FD60	77039570	user32.Trans
0019FD64	0019FD80	
0019FD68	0065AD2C	返回到 game.
0019FD6C	7703C380	user32.PeekM
0019FD70	06B10240	
0019FD74	0019FECC	指向下一个 SI
0019FD78	00D8FE5E	SE处理程序

寄存器 (FPU)

- EAX 00197827
- ECX 0019FD74
- EDX 3DA5AF00
- EBX 1571BBC0
- ESP 0019FD38
- EBP 0019FD64
- ESI 0E5C8FC0
- EDI 14F25014
- EIP 006D7D5A game

C 1 ES 002B 32位
P 1 CS 0023 32位
A 0 SS 002B 32位
Z 0 DS 002B 32位
S 1 FS 0053 32位
T 0 GS 002B 32位
D 0

在这里查看一下血量的值，每一次断点断下，存储血量的地址和esi的值都会不断发生变化。

这个地方的地址一直发生变化说明是已经来到了一个数据结构，如果这个地方单纯的存放的是人物血量的话，就是基址+偏移的方式，不会发生改变。

继续往上追esi

OllyICE - game.bin - [G.P.U* - m主线程, 模块 - game]

文件(F) 查看(V) 调试(D) 插件(P) 选项(O) 窗口(W) 帮助(H) 工具 设置API断点>

暂停

地址	HEX 数据	反汇编	注释
006D7D2F	CC	int3	
006D7D30	8B41 08	mov eax, dword ptr [ecx+8]	
006D7D33	53	push ebx	
006D7D34	56	push esi	
006D7D35	8B30	mov esi, dword ptr [eax]	
006D7D37	57	push edi	
006D7D38	8D79 04	lea edi, dword ptr [ecx+4]	
006D7D3B	EB 03	jmp short 006D7D40	
006D7D3D	8D49 00	lea ecx, dword ptr [ecx]	
006D7D40	3BFF	cmp edi, edi	
006D7D42	8B5F 04	mov ebx, dword ptr [edi+4]	
006D7D45	74 05	je short 006D7D4C	
006D7D47	E8 DF26D4FF	call 0041A42B	
006D7D4C	3BF3	cmp esi, ebx	
006D7D4E	74 22	je short 006D7D72	
006D7D50	3B77 04	cmp esi, dword ptr [edi+4]	

寄存器 (FPU)

EAX 1571BB0C
ECX 14F25010
EDX 439E5FBA
EBX 00000000
ESP 0019FD3C
EBP 0019FD64
ESI 14F24C00
EDI 15734720
EIP 006D7D35 game.

C 0 ES 002B 32位
P 0 CS 0023 32位
A 0 SS 002B 32位
Z 0 DS 002B 32位
S 0 FS 0053 32位
T 0 GS 002B 32位
D 0

ds: [1571BB0C]=3DDF8A40
esi=14F24C00

地址	数值	注释
296F6288	00000001	
296F628C	00000000	
296F6290	00000001	
296F6294	40000000	
296F6298	00000000	
296F629C	00000000	
296F62A0	00000000	
296F62A4	00000001	
296F62A8	00000000	
296F62AC	00000000	
296F62B0	00000000	
296F62B4	00000000	
296F62B8	00000000	
296F62BC	3F000000	
296F62C0	00000000	
296F62C4	00000002	
296F62C8	00000000	

地址	数值	注释
0019FD3C	14F24C00	
0019FD40	00000000	
0019FD44	006D7E18	返回到 game.0
0019FD48	06B10240	
0019FD4C	0067483C	返回到 game.0
0019FD50	77039570	user32. Transl
0019FD54	069980B0	
0019FD58	00000000	
0019FD5C	00000000	
0019FD60	77039570	user32. Transl
0019FD64	0019FD80	
0019FD68	0065AD2C	返回到 game.0
0019FD6C	7703C380	user32. PeekMe
0019FD70	06B10240	
0019FD74	0019FECC	指向下一个 SE
0019FD78	00D8FE5E	SE处理程序
0019FD7C	FFFFFFFF	

Command: [eax]+0xC+0xC+8

esi来自[eax]

血量=([[eax]+C)+0xC]+8

eax这个时候只有两个值在发生变化，而且偏移表达式的地址，不管断下多少次都没有发生改变。

从这个地方的变化说明esi的值不来源于eax，也就是说当前的代码不是顺序执行的，有可能是进到了循环里面。

那么就有必要分析一下当前的这段代码

006D7D34	56	push esi		
006D7D35	8B30	mov esi, dword ptr [eax]		
006D7D37	57	push edi		
006D7D38	8D79 04	lea edi, dword ptr [ecx+4]		
006D7D3B	EB 03	jmp short 006D7D40		
006D7D3D	8D49 00	lea ecx, dword ptr [ecx]		
006D7D40	3BFF	cmp edi, edi		循环头部
006D7D42	8B5F 04	mov ebx, dword ptr [edi+4]		
006D7D45	74 05	je short 006D7D4C		
006D7D47	E8 DF26D4FF	call 0041A42B		
006D7D4C	3BF3	cmp esi, ebx		
006D7D4E	74 22	je short 006D7D72		
006D7D50	3B77 04	cmp esi, dword ptr [edi+4]		
006D7D53	75 05	jnz short 006D7D5A		
006D7D55	E8 D126D4FF	call 0041A42B		
006D7D5A	8B4E 0C	mov ecx, dword ptr [esi+C]		血量=([esi+C]+0xC)+8
006D7D5D	8B11	mov edx, dword ptr [ecx]		
006D7D5F	8B42 08	mov eax, dword ptr [edx+8]		
006D7D62	FFD0	call eax		
006D7D64	3B77 04	cmp esi, dword ptr [edi+4]		
006D7D67	75 05	jnz short 006D7D6E		
006D7D69	E8 BD26D4FF	call 0041A42B		
006D7D6E	8B36	mov esi, dword ptr [esi]		
006D7D70	EB CE	jmp short 006D7D40		循环尾部

这里有一个向上的跳转是当前的循环尾部，jmp跳转的目标地址是循环头部，中间的代码相当于是循环体了。

3D	8D49 00	lea	ecx, dword ptr [ecx]	
40	3BFF	cmp	edi, edi	循环头部
42	8B5F 04	mov	ebx, dword ptr [edi+4]	
45	74 05	je	short 006D7D4C	
47	E8 DF26D4FF	call	0041A42B	
4C	3BF3	cmp	esi, ebx	
4E	74 22	je	short 006D7D72	
50	3B77 04	cmp	esi, dword ptr [edi+4]	
53	75 05	jnz	short 006D7D5A	
55	E8 D126D4FF	call	0041A42B	
5A	8B4E 0C	mov	ecx, dword ptr [esi+C]	血量=[[esi+C]+0xC]
5D	8B11	mov	edx, dword ptr [ecx]	
5F	8B42 08	mov	eax, dword ptr [edx+8]	
62	FFD0	call	eax	
64	3B77 04	cmp	esi, dword ptr [edi+4]	
67	75 05	jnz	short 006D7D6E	
69	E8 BD26D4FF	call	0041A42B	
6E	8B36	mov	esi, dword ptr [esi]	
70	EB CE	jmp	short 006D7D40	循环尾部
72	5F	pop	edi	

40=006D7D40

esi不来自循环体外的eax，也就说明是来自循环体内的[esi]，esi再往上找还是没有发现来源，说明当前的这一句代码一直在循环取值。

如果和数据结构联想到一起的话，很明显这个地方是一个单向链表，C++代码如下

```
class Node
{
public:
    Node<T> *next;
    T data;
```

上面这段代码就是通过对Node节点的循环遍历，来找到想要的目标节点。跳出循环体外

HEX	数据	反汇编	注释
30	8B41 08	mov eax, dword ptr [ecx+8]	
33	53	push ebx	
34	56	push esi	
35	8B30	mov esi, dword ptr [eax]	[eax]链表头
37	57	push edi	
38	8D79 04	lea edi, dword ptr [ecx+4]	
3B	EB 03	jmp short 006D7D40	
3D	8D49 00	lea ecx, dword ptr [ecx]	
40	3BFF	cmp edi, edi	循环头部

esi来源于[eax]，eax就是当前的链表头，这里可以对当前的链表进行取值

第一个对象血量：[[[[eax](#)]]+C]+0xC]+8
 第二个对象血量：[[[[[eax](#)]]]+C]+0xC]+8
 第三个对象血量：[[[[[[eax](#)]]]+C]+0xC]+8

我们已经找到了链表的数据结构，这个时候就没有必要继续往上追了，意义不大。剩余的对象属性各位可以自己去分析

总结

对于链表的逆向来说，需要时刻关注追数据过程中每一步的数据结构变化，特别是变化比较明显的那一种。识别链表的标识在于是否有汇编代码在遍历当前的链表不断进行取值，例如：

```
mov eax,[eax];
```

最后，附上Github地址，里面有游戏下载链接和相关工具，需要请自取：

<https://github.com/TonyChen56/GameReverseNote>