

寻找产生子弹的CALL

首先我们要找到这个记录场上子弹数量的地址



通过CE过滤子弹数量，当子弹数为1时搜1，为2时搜2.....最终可以得到记录当前场上子弹数量的地址

| | | | |
|----------|----------|---------|----|
| 记录当前子弹数量 | 22C2E358 | 4 Bytes | 99 |
|----------|----------|---------|----|

找到子弹数量之后 寻找哪些地址改变了这个值

```
5 00436B91 - 01 47 10 - add [edi+10],eax
4 0043420C - FF 8F F0000000 - dec [edi+000000F0]
```

一个是减少的 一个是增加的 我们当然 选择增加的去观察了啊

上X32Dbg 观察函数调用了

| | | | |
|----------|------------------|------------------------------------|-------------|
| 00436B66 | 56 | push esi | |
| 00436B67 | E8 34DD2800 | call plantsvszombies.6C48A0 | |
| 00436B6C | 8B47 14 | mov eax,dword ptr ds:[edi+14] | |
| 00436B6F | C1E0 10 | shl eax,10 | |
| 00436B72 | 08C3 | or eax,ebx | |
| 00436B74 | 8986 90000000 | mov dword ptr ds:[esi+90],eax | |
| 00436B7A | B8 01000000 | mov eax,1 | |
| 00436B7F | 0147 14 | add dword ptr ds:[edi+14],eax | |
| 00436B82 | 83C4 0C | add esp,C | |
| 00436B85 | 817F 14 00000100 | cmp dword ptr ds:[edi+14],10000 | |
| 00436B8C | 75 03 | jne plantsvszombies.436B91 | |
| 00436B8E | 8947 14 | mov dword ptr ds:[edi+14],eax | |
| 00436B91 | 0147 10 | add dword ptr ds:[edi+10],eax | 436B91 子弹增加 |
| 00436B94 | 8B0D 18967700 | mov ecx,dword ptr ds:[779618] | |
| 00436B9A | 896E 08 | mov dword ptr ds:[esi+8],ebp | |
| 00436B9D | 896E 0C | mov dword ptr ds:[esi+C],ebp | |
| 00436BA0 | 896E 10 | mov dword ptr ds:[esi+10],ebp | |
| 00436BA3 | 896E 14 | mov dword ptr ds:[esi+14],ebp | |
| 00436BA6 | 8B46 18 | mov byte ptr ds:[esi+18],al | |
| 00436BA9 | C746 1C FFFFFFFF | mov dword ptr ds:[esi+1C],FFFFFFFF | |
| 00436BB0 | C746 20 801A0600 | mov dword ptr ds:[esi+20],61A80 | |
| 00436BB7 | 890E | mov dword ptr ds:[esi],ecx | |

| 地址 | 返回到 | 返回自 | 大小 | 注释 | 力 |
|----------|----------|----------|--------|--------------------------|------|
| 0019F93C | 00422E9F | 004368C9 | C | plantsvszombies.004368C9 | 用户模块 |
| 0019F948 | 004885A1 | 00422E9F | 4C | plantsvszombies.00422E9F | 用户模块 |
| 0019F994 | 00488E11 | 004885A1 | 34 | plantsvszombies.004885A1 | 用户模块 |
| 0019F9C8 | 00487282 | 00488E11 | 98 | plantsvszombies.00488E11 | 用户模块 |
| 0019FA60 | 0042BF05 | 00487282 | 4 | plantsvszombies.00487282 | 用户模块 |
| 0019FA64 | 025F4678 | 0042BF05 | 28C4A5 | plantsvszombies.0042BF05 | 用户模块 |
| 0042BF09 | 858B0000 | 025F4678 | 4 | 025F4678 | 用户模块 |
| 0042BF0D | 00000000 | 858B0000 | | 858B0000 | 用户模块 |

看调用堆栈

如何确定是哪个CALL?

有这么多调用 哪个才是我们要找的呢？

看参数，产生子弹至少需要哪两个参数？，答案呼之欲出：坐标，子弹的X坐标和Y坐标，于是我们的这个函数至少有2个参数


```

1 push 0
2 push 0
3 mov eax,0x00049EAD
4 dec eax
5 push eax
6 mov eax,0x1DC7CCD0
7 push 0x5b
8 push 0x6C
9 call 0x422e90

```

| | | | |
|----------|-------------|-------------------------------|-------------------|
| 0048B58C | 8B45 20 | mov eax,dword ptr ss:[ebp+20] | |
| 0048B58F | 8B5C24 40 | mov ebx,dword ptr ss:[esp+40] | |
| 0048B593 | 52 | push edx | 子弹: 子弹生成call 子弹种类 |
| 0048B594 | 53 | push ebx | 子弹行数 |
| 0048B595 | 48 | dec eax | |
| 0048B596 | 50 | push eax | 未知 |
| 0048B597 | 8B45 04 | mov eax,dword ptr ss:[ebp+4] | |
| 0048B59A | 57 | push edi | 像是坐标 |
| 0048B59B | 56 | push esi | 像是坐标 |
| 0048B59C | E8 EF78F9FF | call plantsvszombies.422E90 | 子弹增加2call 14 5个参数 |
| 0048B5A1 | 8B4C24 44 | mov ecx,dword ptr ss:[esp+44] | |

上一层的call又是什么呢？

```

1 push 0
2 push 1
3 push 0
4 push 1DCDC4B8 ;这个比较奇怪 不知道是什么 经过分析发现是植物结构体地址
5 call 48b190

```

这样调用同样可以生成子弹

寻找产生子弹的CALL 总结: call 48b190 和 call 0x422e90 都是可以的

让我们来找找植物数组

为什么要找到植物数组，前面我们找到的call有无法分析的参数，猜测可能是植物结构体地址，所以我们找植物数组进行验证

坚果强受伤法

```
005654C0 - add dword ptr [esi+40],-04
```

esi = ecx = eax

一路向上跟 找到关键函数

| | | | |
|----------|-------------|-----------------------------|----------|
| 00564E2D | 6A 00 | push 0 | |
| 00564E2F | 57 | push edi | |
| 00564E30 | E8 5BF0FFFF | call plantsvszombies.563E90 | 获取eax |
| 00564E35 | 85C0 | test eax,eax | |
| 00564E37 | 74 0E | je plantsvszombies.564E47 | |
| 00564E39 | 57 | push edi | |
| 00564E3A | 8BC8 | mov ecx,eax | |
| 00564E3C | E8 CF040000 | call plantsvszombies.565310 | 植物受伤call |
| 00564E41 | 5F | pop edi | |
| 00564E42 | 5B | pop ebx | |

可以看到 是一个函数生成植物的结构体地址

进去

| | | | |
|----------|-------------|-------------------------------|--------------------------|
| 00563EC0 | 8845 08 | mov eax,dword ptr ss:[ebp+8] | |
| 00563EC3 | 8850 04 | mov edx,dword ptr ds:[eax+4] | edx:"0zr", [eax+4]:"0zr" |
| 00563EC6 | 8D7424 0C | lea esi,dword ptr ss:[esp+c] | |
| 00563ECA | E8 0113EDFF | call plantsvszombies.4351D0 | 生成植物地址 |
| 00563ECF | 84C0 | test al,al | |
| 00563ED1 | 74 6D | je plantsvszombies.563F40 | |
| 00563ED3 | 884D 08 | mov ecx,dword ptr ss:[ebp+8] | |
| 00563ED6 | 8851 1C | mov edx,dword ptr ds:[ecx+1C] | edx:"0zr" |
| 00563ED9 | 8B7424 0C | mov esi,dword ptr ss:[esp+c] | 获取植物数组 |
| 00563EDD | 3B56 1C | cmp edx,dword ptr ds:[esi+1C] | 判断是不是我们的植物 |
| 00563EE0 | 75 DE | jne plantsvszombies.563EC0 | |
| 00563EE2 | 8D4424 10 | lea eax,dword ptr ss:[esp+10] | |

| | | | |
|----------|------------------|---------------------------------|------------------|
| 004351D0 | 57 | push edi | 植物数组guanjiancall |
| 004351D1 | BF 0000FFFF | mov edi,FFFFFF0000 | |
| 004351D6 | 8B06 | mov eax,dword ptr ds:[esi] | |
| 004351D8 | 85C0 | test eax,eax | |
| 004351DA | 75 08 | jne plantsvszombies.4351E4 | |
| 004351DC | 8882 C4000000 | mov eax,dword ptr ds:[edx+C4] | |
| 004351E2 | EB 05 | jmp plantsvszombies.4351E9 | |
| 004351E4 | 05 4C010000 | add eax,14C | |
| 004351E9 | 888A C8000000 | mov ecx,dword ptr ds:[edx+C8] | |
| 004351EF | 69C9 4C010000 | imul ecx,ecx,14C | |
| 004351F5 | 038A C4000000 | add ecx,dword ptr ds:[edx+C4] | |
| 004351F8 | 3BC1 | cmp eax,ecx | |
| 004351FD | 73 12 | jae plantsvszombies.435211 | |
| 004351FF | 90 | nop | |
| 00435200 | 8588 48010000 | test dword ptr ds:[eax+148],edi | |
| 00435206 | 75 13 | jne plantsvszombies.435218 | |
| 00435208 | 05 4C010000 | add eax,14C | |
| 0043520D | 3BC1 | cmp eax,ecx | |
| 0043520F | 72 EF | jb plantsvszombies.435200 | |
| 00435211 | C706 FFFFFFFF | mov dword ptr ds:[esi],FFFFFFFF | |
| 00435217 | 32C0 | xor al,al | |
| 00435219 | 5F | pop edi | |
| 0043521A | C3 | ret | |
| 0043521B | 8906 | mov dword ptr ds:[esi],eax | |
| 0043521D | 8088 41010000 00 | cmp byte ptr ds:[eax+141],0 | |
| 00435224 | 75 80 | jne plantsvszombies.4351D6 | |
| 00435226 | B0 01 | mov al,1 | |
| 00435228 | 5F | pop edi | |
| 00435229 | C3 | ret | |

可以看到跟僵尸的很相似

植物数组长度为14c 首地址在edx+c4处存储

植物结构体

0x8 x坐标 相差0x4f

0xc y坐标 相差0x64

让我们来找找植物数组 总结：通过找到植物数组 经过比较 我们发现 call 48b190的第一个参数就是植物结构体

植物分身

我们试试拦截调用call 修改坐标相关寄存器看看会不会向其他方向发射子弹

| | | | |
|----------|-------------|------------------------------|-------------------|
| 0048B593 | 52 | push edx | 子弹, 子弹生成call 子弹种类 |
| 0048B594 | 53 | push ebx | 子弹行数 |
| 0048B595 | 48 | dec eax | |
| 0048B596 | 50 | push eax | 未知 |
| 0048B597 | 8B45 04 | mov eax,dword ptr ss:[ebp+4] | |
| 0048B59A | 57 | push edi | 坐标y |
| 0048B59B | 56 | push esi | 坐标x |
| 0048B59C | E8 EF78F9FF | call plantsvszombies.422E90 | 子弹增加2call 14 5个参数 |

记录下寄存器的值 使用代码注入工具

| | | |
|-----|----------|--------------------------|
| EAX | 0005390D | |
| EBX | 00000004 | |
| ECX | 00000244 | L'8' |
| EDX | 00000004 | |
| EBP | 2230C05C | |
| ESP | 0019F960 | |
| ESI | 00000232 | L'2' |
| EDI | 00000206 | L'2' |
| EIP | 00488593 | plantsvszombies.00488593 |



成果实现没有植物释放子弹

如此我们只要在发射子弹的合适地方进行HOOK循环调用发射子弹 修改y坐标的值 就可以做到子弹分身

| | | | |
|----------|-------------|--------------------------------|------------------|
| 00488E03 | 8B4B 1C | mov ecx, dword ptr ds:[ebx+1C] | |
| 00488E06 | 6A 00 | push 0 | 子弹: call 生成子弹2 |
| 00488E08 | 51 | push ecx | |
| 00488E09 | 6A 00 | push 0 | |
| 00488E0B | 53 | push ebx | |
| 00488E0C | E8 7F230000 | call plantsvszombies.488190 | 子弹: call 10 4个参数 |

我们选用上一层植物发射子弹的call 进行hook 可以直接获取植物数组的地址

当然 我们已经找到了植物数组的基址 也可以进行判断过后 (是否可以发射子弹) 进行自动发射子弹 也可以直接Hook这个当前这个函数 就是需要更多的处理

| | | | |
|----------|-------------|-----------------------------|------------------|
| 00488E06 | 6A 00 | push 0 | 子弹: call 生成子弹2 |
| 00488E08 | 51 | push ecx | 植物行数 |
| 00488E09 | 6A 00 | push 0 | |
| 00488E0B | 53 | push ebx | 植物结构体指针 |
| 00488E0C | E8 7F230000 | call plantsvszombies.488190 | 子弹: call 10 4个参数 |
| 00488E11 | 5F | pop edi | edi: "02r" |

发现调用完毕 ebx的值还是不变 同样指向发射子弹的植物结构体指针 我们在这个地方HOOK

```

1 //植物子弹分身
2 static DWORD g_dwCall = ZHIWU_ZIDAN_FASHE_CALL_BASE; //发射子弹call
3 static bool g_bIsZiDanFengSheng = false; //控制是否开始
4
5 #define ZHIWU_YPOSITON_FIRST 0x50 //第一个Y坐标
6 #define ZHIWU_YPOSITON_OFFSET 0x64 //Y坐标之间的偏移
7 void __declspec(naked) m_fZiDanFengSheng(void)
8 {
9     __asm
10    {
11        push ebp;
12        mov ebp, esp;
13        sub esp, 0x100;
14    }
15    PDWORD pdwZhiWu;
16    __asm
17    {
18        mov pdwZhiWu, ebx;
19    }
20    if (g_bIsZiDanFengSheng == false)
21    {
22        goto exit;
23    }
24
25    DWORD dwPositionYOld;
26    DWORD dwPositionYNew;
27    __asm
28    {
29        mov dword ptr dwPositionYNew, ZHIWU_YPOSITON_FIRST;
30    }
31
32    dwPositionYOld = *PDWORD((DWORD)pdwZhiWu + JIANGSHI_YPOSITON_OFFSET); //保
33    for (int i = 0; i < 5; i++, dwPositionYNew += ZHIWU_YPOSITON_OFFSET)
34    {
35        if (dwPositionYNew != dwPositionYOld) //同行不再发射子弹了
36        {
37            *PDWORD((DWORD)pdwZhiWu + JIANGSHI_YPOSITON_OFFSET) = dwPositionY
38
39            __asm
40            {
41                push 0
42                push i
43                push 0

```

```
44         push pdwZhiWu
45         call g_dwCall
46     } //发射子弹
47 }
48 }
49 *PDWORD((DWORD)pdwZhiWu + JIANGSHI_YPOSITON_OFFSET) = dwPositionYOld; //i
50 exit:
51     __asm
52     {
53         leave;
54         ret;
55     }
56 }
```