2021/05/12_x86逆向C++_第1课_类成员函数、构造析构的识别

笔记本： x86逆向-C++
创建时间： 2021/5/12 星期三 15:02
作者： ileemi

课前会议

阅读ARK工具源码：

- A盾
- OpenARK

游戏保护反外挂系统：TP（腾讯的内核级反外挂系统）

面向对象的目标：让代码好维护（让编译器做更多的事）。编译器做更多的事，逆向程序就可以从中获取到更多的信息。

类和结构体的区别：结构体权限默认为public。

数组和结构体的区别：数组访问成员时会出现寻址公式（成员类型、大小一致），结构体访问成员时会使用首地址+编译（不是绝对，成员类型、大小一般不一致）。

vs2019 sig文件的制作

根据运行库版本制作：多线程，多线程调试，多线程dll，多线程调试dll（动态库库函数不需要制作）。

使用 Flair 工具制作静态库所需的sig文件（老版本 Flair 工具不会从 .lib 文件中提取特征）：

定位项目中所使用到的 lib 文件（工程属性 --> VS++ 目录 --> 库目录），将其拷贝到一个独立的文件夹内。
提取特征：pcf *.lib vs2019_x86.pat
将特征文件转换为sig文件：sigmake -r vs2019_x86.pat vs2019_x86.sig

类成员函数的识别

main函数中定义类对象，会抬栈：

```
push    ebp
mov     ebp, esp
sub     esp, 208
push    ebx
```

调用约定

thiscall（vs编译器独有）：
第一个参数this指针，通过ecx传参（函数内部有对ecx的保存、恢复，以及直接使用ecx的值）
第二个参数通过入栈传参（**其它参数入栈**）

> 识别为成员函数的第一个要素就是函数的调用约定是否是thiscall。满足thiscall调用约定不一定就是类成员函数（存在模仿）。

类成员函数**只有一个参数** 和 fastcall 调用约定的函数（有类对象参数）识别上有点困难。
fastcall调用约定的函数参数个数为1时：

```
push    10
lea     ecx, [ebp+var_C]
call    SetAge
mov     [ebp+var_1C], 1
mov     [ebp+var_18], 2
lea     ecx, [ebp+var_1C]
call    sub_4111EA
lea     ecx, [ebp+var_C]
call    GetAge
push    eax             ; char
push    offset aD       ; "%d\r\n"
call    printf
```

此时的调用约定不太容易区分

fastcall调用约定的函数参数个数**超过1**时：

```
push    0Ah
lea     ecx, [ebp+var_C]
call    SetAge
mov     [ebp+var_1C], 1
mov     [ebp+var_18], 2
mov     edx, 3E7h
lea     ecx, [ebp+var_1C]
call    sub_4113C0
lea     ecx, [ebp+var_C]
call    GetAge
push    eax             ; char
push    offset aD       ; "%d\r\n"
call    printf
```

通过 edx, ecx 传参，确定函数的调用约定就是 _fastcall

> 使用同一个类对象的函数都可以解释为成员函数（函数内部有对ecx寄存器的直接使用）：

64位的软件在编写函数时给定调用约定，编译器不错参考（thiscall、sdtcall、fastcall没有区别）。

> 类似fastcall的定义，但是仅仅使用ecx传递this指针，其余参数通过栈传递，平栈传参方向和fastcall一样。当成员函数使用其它的调用约定时，第一个参数必须是this指针，不管是栈传递还是寄存器传递。用栈传递的时候，第一个参数是最后一个push（距离call最近的一个push就是this指针）的，用寄存器传递参数时，第一个参数是ecx。

fastcall：通过ecx、edx传参
第一个参数this指针，通过ecx传参（内部直接使用）
第二个参数不是通过push进行入栈传参，通过寄存器传递
从三个参数往后通过push进行入栈传参

构造函数

构造函数的识别：语法：不能有返回值，可以有参数，可以重载

1. 是成员函数
2. 类对象生命周期开始的时候第一个调用的成员函数
3. 调用约定一定是thiscall（调用约定修改了没用）
4. 一定会返回this指针（不同的成员函数不一定会返回）
5. 不包含堆对象（什么时候new对象什么时候调用构造，和第3点冲突）

> 出现和构造函数特点类似的成员函数可以将其还原成构造。

析构函数

析构函数：语法：不能有返回值，不能有参数，不可以重载

1. 是成员函数
2. 对象生命周期结束时调用
3. 返回值是void
4. 调用约定一定是thiscall

const 类成员对象必须初始化，且后续不能对其进行修改。const 还原不了。

const 类对象只能调用常成员函数不能调用成员函数 。非 const 类对象可以调用常成员函数以及成员函数。

堆对象的识别：找特征 new delete

> printf最后会调用标准输出缓冲区，会用到 stdout 宏(__acrt_iob_func(1))；同理，scanf 会用到 stdin 宏(__acrt_iob_func(0))：

```
_ACRTIMP_ALT FILE* __cdecl __acrt_iob_func(unsigned _Ix);

#define stdin  (__acrt_iob_func(0))
#define stdout (__acrt_iob_func(1))
#define stderr (__acrt_iob_func(2))
```

```
mov     [ebp+ArgList], eax
mov     eax, [ebp+ArgList]
push    eax             ; ArgList
push    0               ; Locale
mov     ecx, [ebp+Format]
push    ecx             ; Format
mov     esi, esp
push    1               ; Ix
call    ds:__acrt_iob_func
add     esp, 4
```

堆对象

New的构造函数识别：

- 申请的空间是否为空，为空就不执行构造。
- 使用构造函数的返回值

Debug：

```
mov     large fs:0, eax
mov     ecx, offset unk_41E01F
call    j_@__CheckForDebuggerJustMyCode@4 ; __CheckForDebuggerJustMyCode(x)
push    0Ch             ; Size
call    j_??2@YAPAXI@Z  ; operator new(uint)
add     esp, 4
mov     [ebp+var_EC], eax
mov     [ebp+var_4], 0
cmp     [ebp+var_EC], 0
jz      short loc_411F97
push    0Ch             ; Size
mov     ecx, [ebp+var_EC]
call    sub_41107D      ; 检查溢出
mov     ecx, [ebp+var_EC]
call    sub_41129E      ; 构造函数
mov     [ebp+var_F4], eax ; 保存构造函数的返回值(this指针)
jmp     short loc_411FA1
; ---------------------------------------------------------------------------

loc_411F97:                             ; CODE XREF: _main_0+65↑j
mov     [ebp+var_F4], 0

loc_411FA1:                             ; CODE XREF: _main_0+85↑j
mov     eax, [ebp+var_F4]
mov     [ebp+var_E0], eax
mov     [ebp+var_4], 0FFFFFFFFh
mov     ecx, [ebp+var_E0]
```

申请空间

判断堆空间是否申请成功

调用构造

Release：

```
                mov     eax, ___security_cookie
                xor     eax, ebp
                push    eax
                lea     eax, [ebp+var_C]
                mov     large fs:0, eax                 申请空间
                push    0Ch             ; Size
                call    ??2@YAPAXI@Z    ; operator new(uint)
                mov     [ebp+Block], eax
                mov     ecx, eax
;   try {
                mov     [ebp+var_4], 0                  类成员初始化
                call    sub_4011D0
                call    sub_401050                      构造
                mov     esi, eax
                push    ecx
                mov     ecx, esi
                call    sub_4010E0
                mov     ecx, esi
                call    sub_401100
```

Delete 的构造函数识别：

- 判断在调用析构（SEH）
- 调用的是析构代理
- 析构代码带参数

代理析构（Debug）：传递的参数为1，还原代码 -- **delete pObj;**
显示调用析构（Debug）：传递的参数为0，还原代码 -- **pObj->~CTest();**

Debug：

```
                push    0                               参数为0，显示调用析构
                mov     ecx, [ebp+var_14]
                call    proxy_Construct
                mov     eax, [ebp+var_14]
                mov     [ebp+var_F8], eax
                cmp     [ebp+var_F8], 0
                jz      short loc_415F00
                push    1                               参数为1，析构代理
                mov     ecx, [ebp+var_F8]
                call    proxy_Construct
                mov     [ebp+var_100], eax
                jmp     short loc_415F0A
; ---------------------------------------------------------------------------

loc_415F00:                             ; CODE XREF: _main_0+F9↑j
                mov     [ebp+var_100], 0

loc_415F0A:                             ; CODE XREF: _main_0+10E↑j
                xor     eax, eax
                mov     ecx, [ebp+var_C]
                  rep stosd
                  pop     ecx
                  mov     [ebp+var_8], ecx
                  mov     ecx, [ebp+var_8]
                  call    Deconstruction  ; 析构
                  mov     eax, [ebp+arg_0]
                  and     eax, 1          ; 判断最低位是否为1
                  jz      short loc_411E81
                  push    0Ch
                  mov     eax, [ebp+var_8]
                  push    eax             ; void *
                  call    delete          ; delete
                  add     esp, 8

loc_411E81:                             ; CODE XREF: proxy_Construct+31↑j
                  mov     eax, [ebp+var_8]
```

Release（没有开启内联）：

```
                push    0               ; char
                call    sub_4011F0
                test    esi, esi
                jz      short loc_4011B6
                push    1               ; char
                mov     ecx, esi        ; Block
                call    sub_4011F0

loc_4011B6:                             ; CODE XREF: _main+7B↑j
                xor     eax, eax
                mov     ecx, [ebp+var_C]
                mov     large fs:0, ecx
                pop     ecx
                pop     esi
                mov     esp, ebp
                pop     ebp
                retn
```

显示析构代理

析构代理