

# 虚表和虚函数以及纯虚函数的逆向分析

---

## 1.环境准备

---

1. 调试器x64dbg
2. ide VS2019
3. 系统 win10

## 2.正向代码

---

让我们先来看一下正向代码，本次实验围绕该正向代码展开：

```
#include<stdio>
class A {
public:
    int a=1, b=2;
    virtual void printA() {
        printf("A\n");
    }
    virtual void printB() = 0;
};

class B :public A {
public:
    int a=3, b=4;
    void printB() {
        printf("B\n");
    }
};

int main() {
    B* b =new B();
    b->printA();
    b->printB();
    return 0;
}
```

## 3.逆向分析

---

### 3.1寻找入口点



| 偏移   | 内容        |
|------|-----------|
| 0x0  | 虚表地址      |
| 0x4  | 父类第一个数据成员 |
| 0x8  | 父类第二个数据成员 |
| 0xC  | 子类第一个数据成员 |
| 0x10 | 子类第二个数据成员 |

### 3.2.2 memset

```

mov     edx, eax
mov     dword ptr ss:[esp], edx
mov     dword ptr ss:[esp+4], 0
mov     dword ptr ss:[esp+8], 14
mov     dword ptr ss:[ebp-10], ecx
mov     dword ptr ss:[ebp-14], eax
call    <虚表和函数纯函数分析-抽象类._memset>
mov     ecx, dword ptr ss:[ebp-14]
call    <虚表和函数纯函数分析-抽象类.public: __thiscall B::B(void)>
mov     ecx, dword ptr ss:[ebp-14]
mov     dword ptr ss:[ebp-C], ecx
mov     edx, dword ptr ss:[ebp-C]
mov     esi, edx
mov     edx, dword ptr ds:[edx]

```

| 地址       | 十六进制  | ASCII             |
|----------|---|-------------------|
| 00824D18 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....             |
| 00824D28 | 00 00 00 00 FD FD FD FD 8A F7 91 54 56 48 00 00 | ....yyyy.÷.TV     |
| 00824D38 | 58 47 82 00 C0 9F 82 00 60 3E 82 00 60 3E 82 00 | XG. .A. . > . .   |
| 00824D48 | 60 3E 82 00 60 3E 82 00 00 38 82 00 78 36 82 00 | > . . > . . ; . X |
| 00824D58 | 60 3E 82 00 00 00 00 00 9E F7 90 41 58 48 00 0C | > . . . . ÷. AX   |
| 00824D68 | 05 00 00 00 00 00 00 00 FF FF FF FF 00 00 00 00 | . . . . . yyyv    |

调用memset对对象空间初始化为0x0。

### 3.2.3 构造函数

```

mov     dword ptr ss:[esp], ecx
mov     dword ptr ss:[esp+4], 0
mov     dword ptr ss:[esp+8], 14
mov     dword ptr ss:[ebp-10], ecx
mov     dword ptr ss:[ebp-14], eax
call    <虚表和函数纯函数分析-抽象类._memset>
mov     ecx, dword ptr ss:[ebp-14]
call    <虚表和函数纯函数分析-抽象类.public: __thiscall B::B(void)>
mov     ecx, dword ptr ss:[ebp-14]
mov     dword ptr ss:[ebp-C], ecx
mov     edx, dword ptr ss:[ebp-C]
mov     esi, edx
mov     edx, dword ptr ds:[edx]
mov     ecx, esi
mov     dword ptr ss:[ebp-18], eax
call    dword ptr ds:[edx]
mov     eax, dword ptr ss:[ebp-C]
mov     ecx, dword ptr ds:[eax]
mov     dword ptr ss:[ebp-1C], ecx
mov     ecx, eax
mov     eax, dword ptr ss:[ebp-1C]

```

| 地址       | 十六进制  | ASCII           |
|----------|---|-----------------|
| 00824D18 | 04 40 EE 00 01 00 00 00 02 00 00 00 03 00 00 00 | .@i . .         |
| 00824D28 | 04 00 00 00 FD FD FD FD 8A F7 91 54 56 48 00 00 | ....yy          |
| 00824D38 | 58 47 82 00 C0 9F 82 00 60 3E 82 00 60 3E 82 00 | XG. .A. . > . . |
| 00824D48 | 60 3E 82 00 60 3E 82 00 00 38 82 00 78 36 82 00 | > . . > . .     |
| 00824D58 | 60 3E 82 00 00 00 00 00 9E F7 90 41 58 48 00 0C | > . . . .       |
| 00824D68 | 05 00 00 00 00 00 00 00 FF FF FF FF 00 00 00 00 | . . . . . yyyv  |

```

00EE106F C3 ret
00EE1070 55 push ebp
00EE1071 89E5 mov ebp,esp
00EE1073 83EC 0C sub esp,C
00EE1076 894D FC mov dword ptr ss:[ebp-4],ecx
00EE1079 8B45 FC mov eax,dword ptr ss:[ebp-4]
00EE107C 89C1 mov ecx,eax
00EE107E 8945 F8 mov dword ptr ss:[ebp-8],eax
00EE1081 E8 2A000000 call <虚表和虚函数纯虚函数分析-抽象类.public: __thiscall A::A(void)>
00EE1086 8D0D 0440EE00 lea ecx,dword ptr ds:[<const B::'vftable'>]
00EE108C 8B55 F8 mov edx,dword ptr ss:[ebp-8]
00EE108F 890A mov dword ptr ds:[edx],ecx
00EE1091 C742 0C 03000000 mov dword ptr ds:[edx+C],3
00EE1098 C742 10 04000000 mov dword ptr ds:[edx+10],4
00EE109F 8945 F4 mov dword ptr ss:[ebp-C],edx
00EE10A2 89D0 mov eax,edx
00EE10A4 83C4 0C add esp,C
00EE10A7 5D pop ebp
00EE10A8 C3 ret

```

初始化后，调用子类构造函数，子类构造函数里面又调用了父类的构造函数，从而进行全部的初始化。返回值为对象的地址。

### 3.2.4 虚函数和纯虚函数的调用

如图：

```

00000000 call <虚表和虚函数纯虚函数分析-抽象类.public: __thiscall
C mov ecx,dword ptr ss:[ebp-14]
4 mov dword ptr ss:[ebp-C],ecx
4 mov edx,dword ptr ss:[ebp-C]
mov esi,edx
mov edx,dword ptr ds:[edx]
mov ecx,esi
8 mov dword ptr ss:[ebp-18],eax
call dword ptr ds:[edx]
4 mov eax,dword ptr ss:[ebp-C]
4 mov ecx,dword ptr ds:[eax]
4 mov dword ptr ss:[ebp-1C],ecx
4 mov ecx,eax
4 mov eax,dword ptr ss:[ebp-1C]
4 call dword ptr ds:[eax+4]
4 xor eax,eax
4 add esp,24

```

| 地址       | 十六进制        |             |    |
|----------|-------------|-------------|----|
| 00EE4004 | E0 10 EE 00 | 00 11 EE 00 | 00 |
| 00EE4014 | 00 00 00 00 | 00 00 00 00 | 00 |
| 00EE4024 | 00 00 00 00 | 00 00 00 00 | 00 |
| 00EE4034 | 40 40 EE 00 | 60 40 EE 00 | 00 |
| 00EE4044 | 01 00 00 00 | 00 00 00 00 | 00 |

通过二次寻址，由对象地址取出首个4字节值为虚表数组的地址，再进行寻址，得到虚函数和纯虚函数的地址，分别调用。

内存布局：

| 偏移  | 内容      |
|-----|---------|
| 0x0 | 父类虚函数地址 |
| 0x4 | 纯虚函数地址  |

## 4.总结

虚表和虚函数以及纯虚函数分析总结：

- 1.对象变量首个4字节存放虚表地址
- 2.局部变量分配基类在子类之前
- 3.纯虚函数和虚函数在同一张虚表里面（此代码得出的猜想）
- 4.虚函数在纯虚函数之前（此代码得出的猜想）

class子类的构造过程

- 1.new分配空间，返回对象地址，之后memset初始化对象
- 2.利用此地址，调用构造函数默认的构造函数所有数据成员赋值
- 3.再对部分或者全部数据成员赋值
- 4.通过虚表调用虚函数和纯虚函数

