Virtual function table (*A.K.A vtable*) is the key to manage virtual function which is a key mechanism to support polymorphism in C++ language. *VTable* is used to support dynamic dispatch or runtime method binding in C++. At the same time it is a popular target of malicious programmers to execute their malicious behavior on the system. Attacker exploits the virtual function table mechanism to execute shellcode in the system. In this writing we'll see how attacker can execute virtual function table attack.

In C++, for each class with virtual function, the compiler will create one or more virtual function table. The number of created tables depend on the class inheritance hierarchy. Inside each instance of the class, a pointer is created. This pointer is called *virtual function pointer* or *vfptr* and it points to the VTable of the corresponding class. For example, take a look on the following class hierarchy, its object layout and the corresponding *VTable*.

```
class B1{
        int B1_i;
        int B1_j;
        virtual void B1_f1();
        virtual void B1_f2();
};

class B2{
        int B2_i;
        int B2_j;
        virtual void B2_f1();
        virtual void B2_f2();
};

class A : public B1, public B2{
        int A_i;
        int A_j;
        virtual void A_f1();
        virtual void A_f2();
};
```

Object Layout with VTable Pointer

As the figure dictates, the virtual function of the derived class and the first base class are normally placed together on the same *VTable*. During runtime, if there is a need to make a

virtual function call, system will first read out *vfptr* from the object. Using the *vfptr*, it will read out the target function from the *VTable* and finally call the target function.

Now from an attacker's perspective, there are 3 possible ways to execute a virtual function table attack.

1. *VTable* corruption attack: Modify the *VTable* content so it can point to some malicious function
2. *VTable* Inject Attack: Inject a fake *VTable* into the application and modify *vfptr* to point to this injected malicious *VTable*.
3. *VTable* Reuse Attack: Modify the *vfptr* to point to some already existing *VTable* or existing code or data in memory. It differs from *VTable* injection attack by its use of existing code, not attacker crafted code.

The attacker normally make use of some existing vulnerability in the software to execute any of these 3 attacks. Among these, *VTable* injection attack is most popular because it provides attacker more control of the attack.