Reversing C++

Dr. Jared DeMott
CHIEF HACKING OFFICER

@jareddemott www.vdalabs.com

Overview

Some C++ Specifics

Demo

C++ Details from Demo

```
int _tmain(int argc, _TCHAR* argv[]) {

  Parent_Class_One Obj1;

  Derived_Class_Two Obj2;

  cout << "This is correct: " << Obj1.SecondFunction(20) << endl;

  cout << "This isn't: " << Obj2.SecondFunction(20) << ".  Why?"<<
endl;

  system("pause");

  return 0;

}
```

Running the Program

```
cmd>VirtualFuncs.exe

This is correct: 1010

This isn't: 1010.  Why?
```

```cpp
class Parent_Class_One {
protected:
    double a;
public:
    Parent_Class_One(double = 2);
    double FirstFunction(double);
    double SecondFunction(double);
};
Parent_Class_One::Parent_Class_One(double val) { a = val;}
```

```
double Parent_Class_One::FirstFunction(double num) {
    return(num/2);

}

double Parent_Class_One::SecondFunction(double num) {   return(
FirstFunction(num) + 1000 );

}

class Derived_Class_Two : public Parent_Class_One {

public:

    virtual double FirstFunction(double num);

};

double Derived_Class_Two::FirstFunction(double num) {
    return(num/4);

}
```

Demo

Open both the fixed/incorrect in IDA pro

Was broken due to static bindings

Fixed to make the bindings dynamic

Compare the compiled programs in IDA

What’s something interesting?

Locate RTTI pointer, precede the vtable

Find each class name

Label virtual functions in vtable

C++

Simple classes look just like C structs

Function names my be “weird” due to name mangling

- Mangling is done to allow *function name overloading*
  - Same function name, different arguments

C++ objects

- Use of ***this*** in Visual Studio (ecx)
  - ‘*This*’ is a pointer to the object making the function call
  - First arg passed for g++

Object Construction

Allocate space for the object

- *New* is called prior to constructor
  - Size parameter reveals size of object
  - Result passed to constructor as ‘this’

Call super (base) class constructor if applicable

- Reveals object hierarchy

Copy address of vtable into vtable pointer field

Initialize class members

Vtables

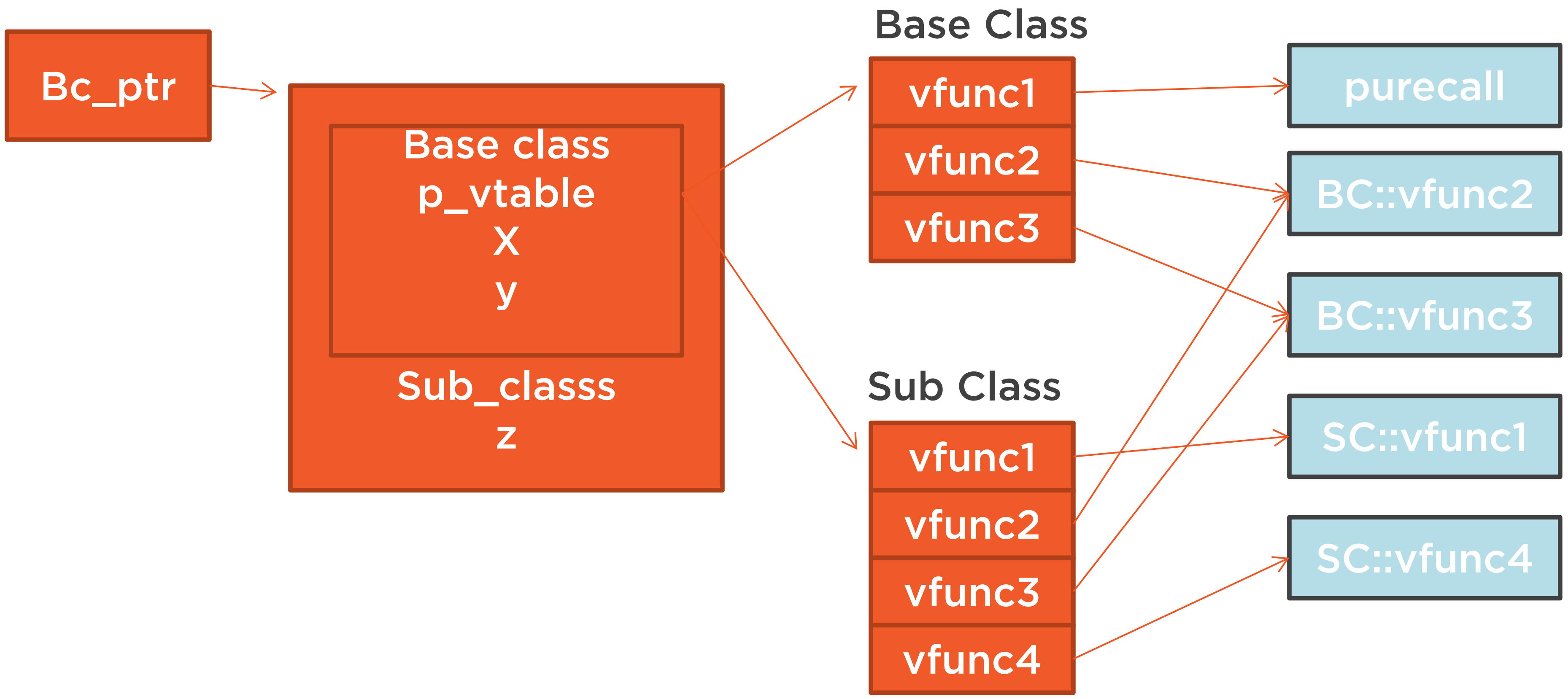
Polymorphism

- If class contains virtual functions
- Table of function pointers
  - One entry per virtual function
  - Pure virtual slot points to error function
    - If used in abstract superclass

Objects with virtual functions get a pointer to their vtable

- True for gcc and Visual Studio
- First member of their “struct”

vtable


Bc_ptr
Base class
p_vtable
X
y
Sub_classs
z
Base Class
vfunc1
vfunc2
vfunc3
Sub Class
vfunc1
vfunc2
vfunc3
vfunc4
purecall
BC::vfunc2
BC::vfunc3
SC::vfunc1
SC::vfunc4

Object Destruction

Restore vtable pointer

- In case a subclass replaced it during construction

Perform code-specific local destruction tasks

If any data members are objects, call their destructor

- Free memory

Finally, call super class destructor

RTTI

Runtime Type Identification

- For *typeid* and *dynamic cast*

Utilized only for objects with virtual methods

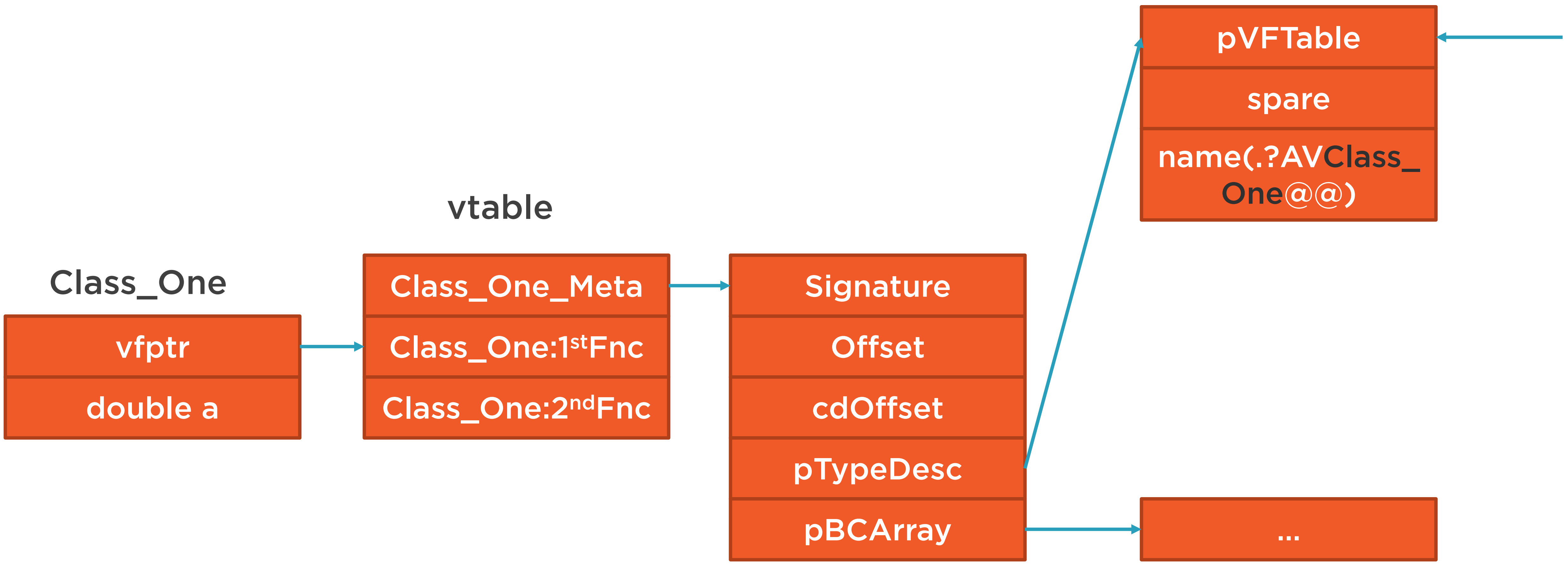
- If *typeid* isn't used, RTTI information could be stripped out of the binary

Pointer to RTTI proceeds p_vtable

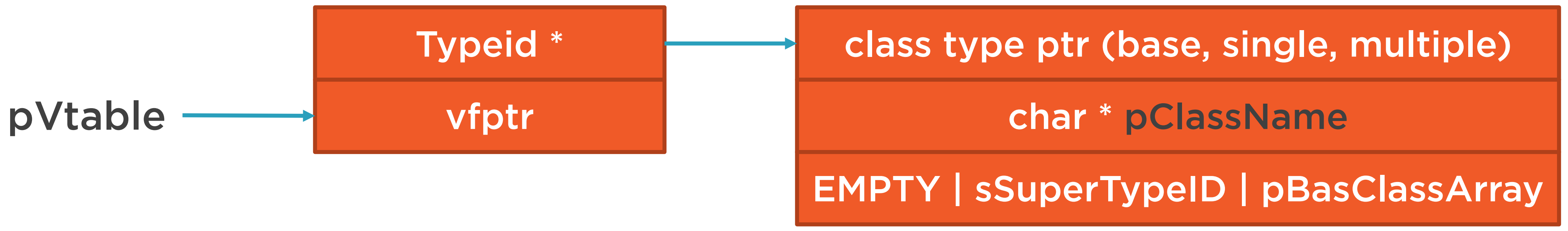
RTTI structures vary among compilers

- They all include information about original class name and super classes

Visual Studio RTTI Diagram



g++ RTTI


pVtable
Typeid *
vfptr
class type ptr (base, single, multiple)
char * pClassName
EMPTY | sSuperTypeID | pBasClassArray

Discovering Member Functions

Find Virtual functions via the vtable

Non-static members

- Visual Studio by recognizing that a function uses ecx ‘*this*’
- Any compiler by tracing the use of ‘*this*’ as a parameter

Static member functions

- More difficult since ‘this’ is not used

Summary

C++ Example Code

Classes and Members

RTTI

Next:

- Extending IDA