

# Altering the vtable

Or: how polymorphic can an object be?

Or: unexpected and malicious aspect-oriented programming

Stefan Huber

Chaostreff Salzburg

1. April 2011

# Motivation

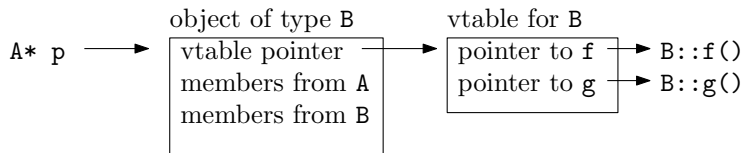
- ▶ Dynamic polymorphism is supported in C++ via virtual member functions.

```
1 class A
2 {
3     virtual void f() { cout << "A::f" << endl; };
4     virtual void g() { cout << "A::g" << endl; };
5 };
6 class B : public A
7 {
8     virtual void f() { cout << "B::f" << endl; };
9     virtual void g() { cout << "B::g" << endl; };
10 };
11
12 void test(A* p)
13 {
14     p->f();    // prints B::f
15 }
16 int main()
17 {
18     test(new B());
19     return 0;
20 }
```

# Technical realization

- ▶ A polymorphic object contains a pointer to the “vtable” — a table of function pointers to the functions to call at runtime.

```
1 void test(A* p)
2 {
3   p->f();    // prints B::f
4 }
```



- ▶ Fun: can we modify the vtable in order to inject malicious code?
  - ▶ No: vtable lies in read-only segment in memory
  - ▶ But we can change the vtable pointer!
  - ▶ Idea: copy the vtable and alter the copy.
- ▶ Code demo...