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.

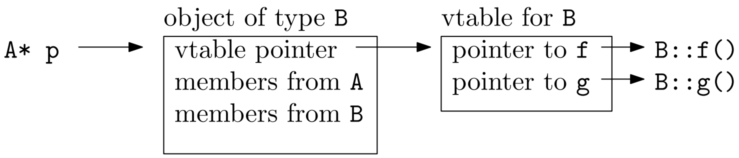
```
class A
{
  virtual void f()  { cout << "A::f" << endl; };
  virtual void g()  { cout << "A::g" << endl; };
};
class B : public A
{
  virtual void f()  { cout << "B::f" << endl; };
  virtual void g()  { cout << "B::g" << endl; };
};

void test(A* p)
{
  p->f();    // prints B::f
}
int main()
{
  test(new B());
  return 0;
}
```

Technical realization

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

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



- 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. . .