Eric Mullen
Spring 2017
Lecture 26: Vtable Hijacking
Administirivia

- HW7 due Thurs at Midnight
  - If you have late days you might as well use them (no more than 2 at once though)
- HW5 grades back, we’re working madly on HW6b
- Final Review Session 4:30pm-6:30pm on Tues 6/6 (in CSE 403)
- Final Exam 2:30pm on Wed 6/7 in THIS ROOM
Course Evaluations

• Please fill them out!
• Your **honest feedback** helps me learn to teach better
• Things you say about me can and will be used (by me) in job applications next fall
• This is the first class I’ve taught, I’m sure I have much to learn
• Link is on course website, and here:  
  https://uw.iasystem.org/survey/178403
Inheritance

class A {
    virtual void msg() {
        cout << "A";
    }
    int f;
};
class B : public A {
    void msg() {
        cout << "B";
    }
};

A* x;
if (rand()%2) {
    x = new B();
} else {
    x = new A();
}
x->msg();
class A {
  virtual void msg() {
    cout << "A"; 
  }
  int f;
};

class B : public A {
  void msg() {
    cout << "B"; 
  }
};

vtables

A
  A::msg
  ... 
  vtable
  int f 

B
  B::msg
  ... 
  vtable
  int f
making a call

A::msg
...

EVIL::doEvil
...

A
vtable
int f

Evil
vtable
int f
Use after Free

- Memory allocators reuse memory when they can

```c
C1* x = new C1();
delete x;
C2* y = new C2();
x->method(); //could call C2 method
```
Not just what if?

- This led to a zero day exploit in Chrome in 2012
  - Assuming an already compromised tab process, could execute arbitrary code as the browser kernel
  - Used vtable hijacking along with a use-after-free bug
How can we fix it?

- Any ideas?
How can we fix it?

• There’s an invariant being violated here: when making a virtual call, the vtable pointer is one of the allowed vtable pointers

• Just check that it’s the one of the correct pointers right before making the call

How can we fix it?

A::toString

...
Which vtable pointers are OK?

Static Type Classes

A

B C

D E F
Performance

- Evaluated on the open source part of Chrome, called Chromium

- 2.1% performance overhead overall

- 7.5% memory overhead overall
Downsides

• Can you think of any downsides to this approach?