HexType: fast type safety for C++ programs

Yuseok Jeon, Hui Peng, Mathias Payer

Motivation

- C++ is used in many areas because of its modularity and performance.
- Type-casting converts a pointer from one object type into another.
- Down-casting (converting a base class pointer to a derived class pointer) has critical security implications.
- This vulnerability class has recently received increasing attention and is known as type confusion (unsafe down-casting).
- Several existing solutions are severely limited by both high runtime performance and low coverage (e.g., UBSAN only handles type-casting between polymorphic classes, a small subset of all casts).

Type Confusion Attack

Parent

Child

P

P

C

C

class P {
    int p_data;
};

class C: public P {
    int c_data;
};
P *Pptr = new P;
C *Cptr = static_cast<C*>(Pptr);
Cptr->c_data; (Type confusion!!)

HexType Architecture

- We introduce a practical technique that has low runtime performance overhead and broad coverage, covering all type casts in an application.
- The source for high runtime overhead of existing approaches is the combination of expensive class relation checks and tracking type information for different memory areas.
- We devise and apply various optimization methods to reduce overhead for class relation checking and tracking type information.

Type Confusion Detection

Previous approaches have limitations to find type confusion vulnerability successfully regarding overhead and coverage.

- Thus, we propose a novel approach with three advantages: (i) full coverage, checking the type information of all casts, (ii) a fast general type check that leverages an indexed per-object metadata table and local information at the current program location, and (iii) low tracking overhead by leveraging architectural features.

- We Plan to:
  - Apply various optimization methods
  - Handle reinterpret and dynamic cast