HexType: Efficient Detection of Type Confusion Errors for C++

Yuseok Jeon Priyam Biswas Scott A. Carr
Byoungyoung Lee Mathias Payer

Motivation

- C++ is a popular programming language
  - Google Chrome, Firefox, and Java Virtual Machine
- Type confusion bugs are emerging attack vectors
  - Google Chrome (CVE-2017-5023)
  - Adobe Flash (CVE-2017-2095)
- Existing sanitizers are incomplete and impractical

Outline

- Motivation
- **Type Confusion Problem**
- HexType
- Conclusion

C++ Casting Operators

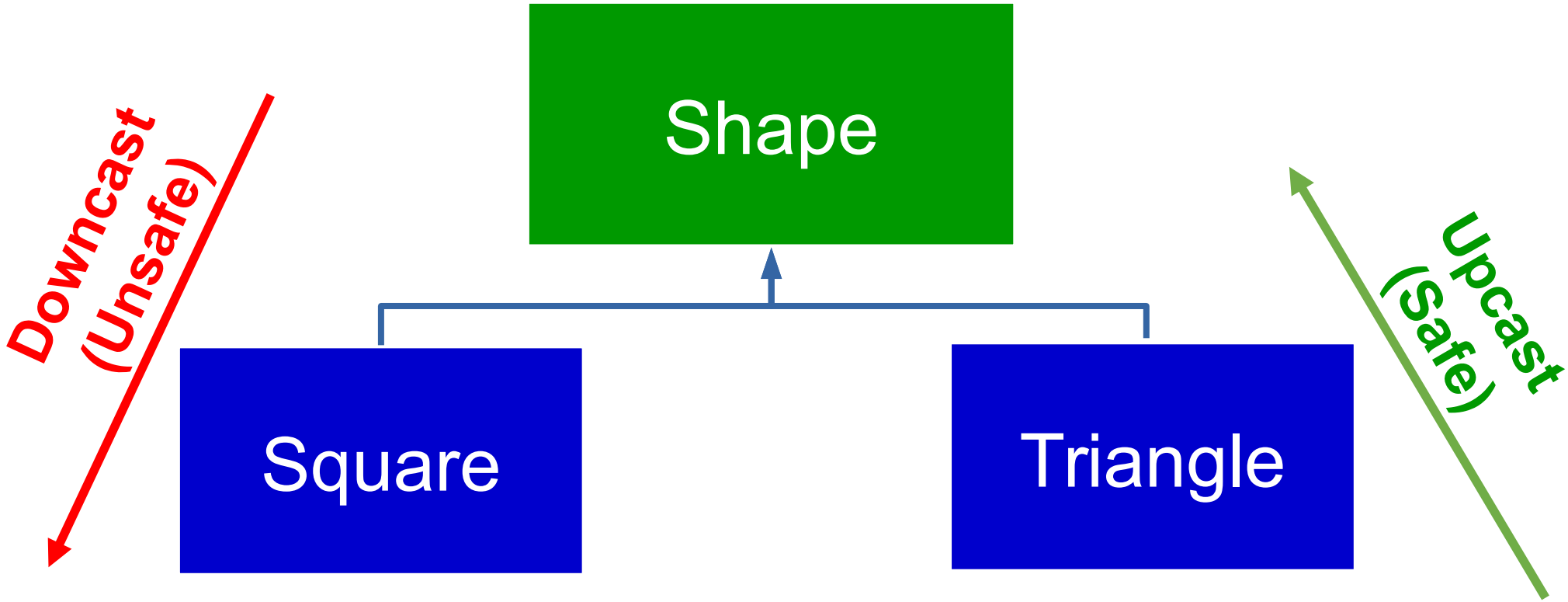
`static_cast<ToClass>(Object)`

- Compile time check
- No runtime type information

`dynamic_cast<ToClass>(Object)`

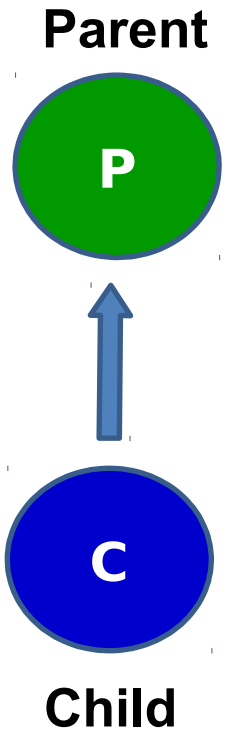
- Runtime check
- Requires Runtime Type Information (RTTI)
- Not used in performance critical code

Type Confusion Problem

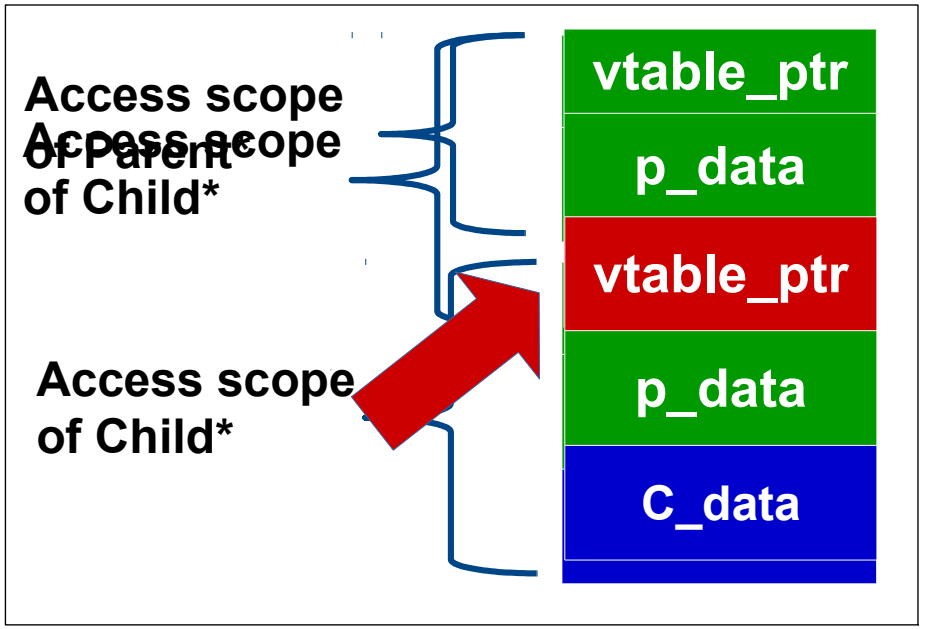


- Upcast: from a derived class to its parent class
- Downcast: from a parent class to a derived class

Type Confusion Example




```
class Parent {
    int p_data;
    virtual void print (void){ };
};

class Child: public Parent {
    int c_data;
    void print (void) override{ };
};
```



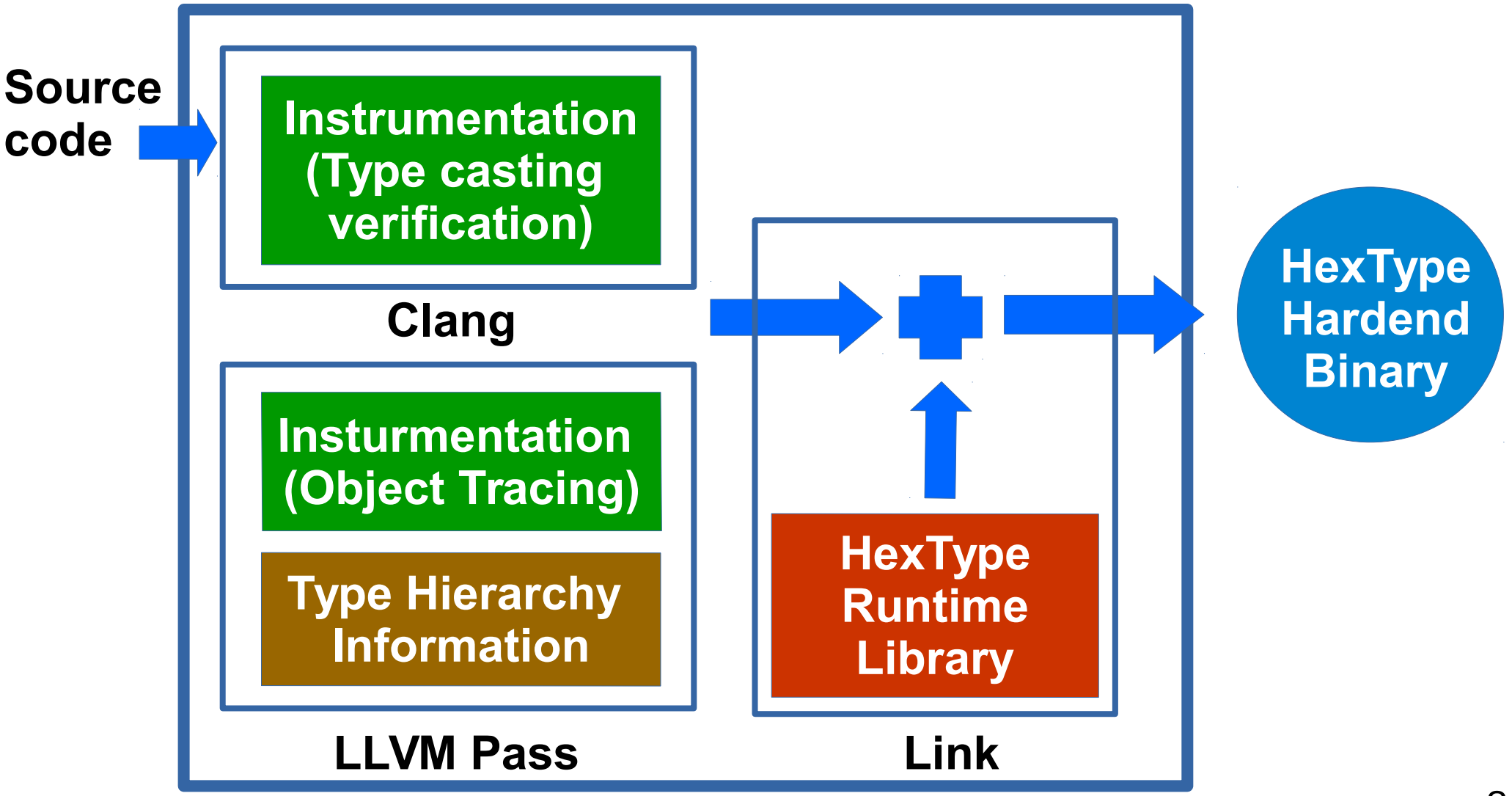

```
Parent *Pptr = new Parent;
Child *Cptr = static_cast<Child*>(Pptr);
Cptr→c_data = 0x12345678;
```

Memory Corruption !
Type Confusion Problem !

Outline

- Motivation
- Type Confusion Problem
- **HexType**
  - **Design**
  - Optimization
  - Coverage
- Conclusion

HexType Overview



Type Casting Verification

[Source code]

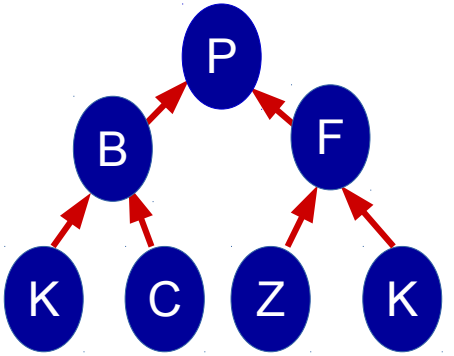
```
……….
B *pB = new B;
Update_objTypeMap(pB, B);

C *pC = static_cast<C*>(pB);
Verify_type_casting(pB, C);
```

Object to Type Mapping Table	
0x1232	Type1
0x2312	Type2
0x2333	Type3

[Runtime library]

```
Verify_type_casting (Ptr *SrcPtr, TypeInfo Dst) {
   TypeInfo Src = getSrcType(SrcPtr);
   isTypeConfusionCast(Src, Dst);
}
```

Type relation information



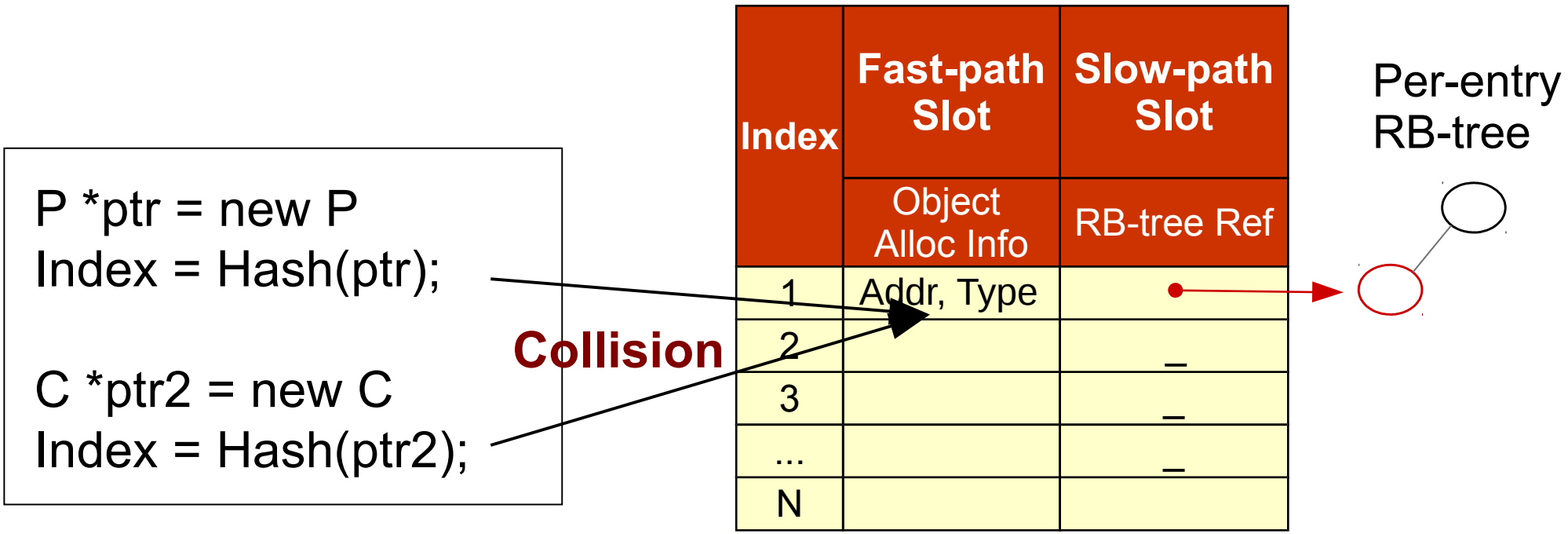
Outline

- Motivation
- Type Confusion Problem
- **HexType**
  - Design
  - **Optimization**
  - Coverage
- Conclusion

Previous Works

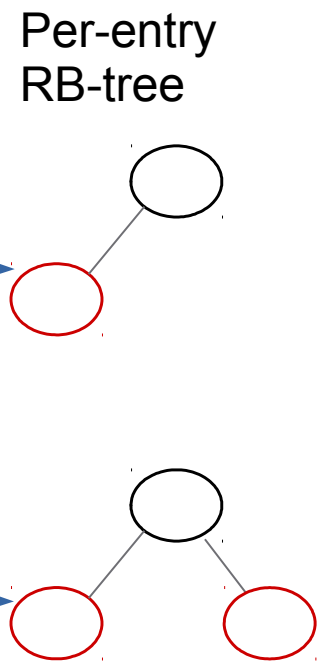
	Selective Object Tracing	Casting Replace	Compile time Verification	Data Structure
UBSAN	X	X	X	RTTI
CaVer (SEC 2015)	√	X	X	Shadow Memory + RB Tree
TypeSan (CCS 2016)	√	X	X	Shadow Memory
HexType (CCS 2017)	√	√	√	HashMap + RB Tree

Optimized ObjTypeMap



Optimized ObjTypeMap

Index	Fast-path Slot			Slow-path Slot
	Allocated Object Ref	Hashvalue for Object Name	Type Relationship Information Ref	RB-tree Ref
1	0x417000	2341234	0x51723D	•
2	0x41563C	1312321	0x51724D	-
3	0x41723D	7231234	0x51724D	-
...	-	-	-	-
N	0x41563E	4232123	0x51623D	•



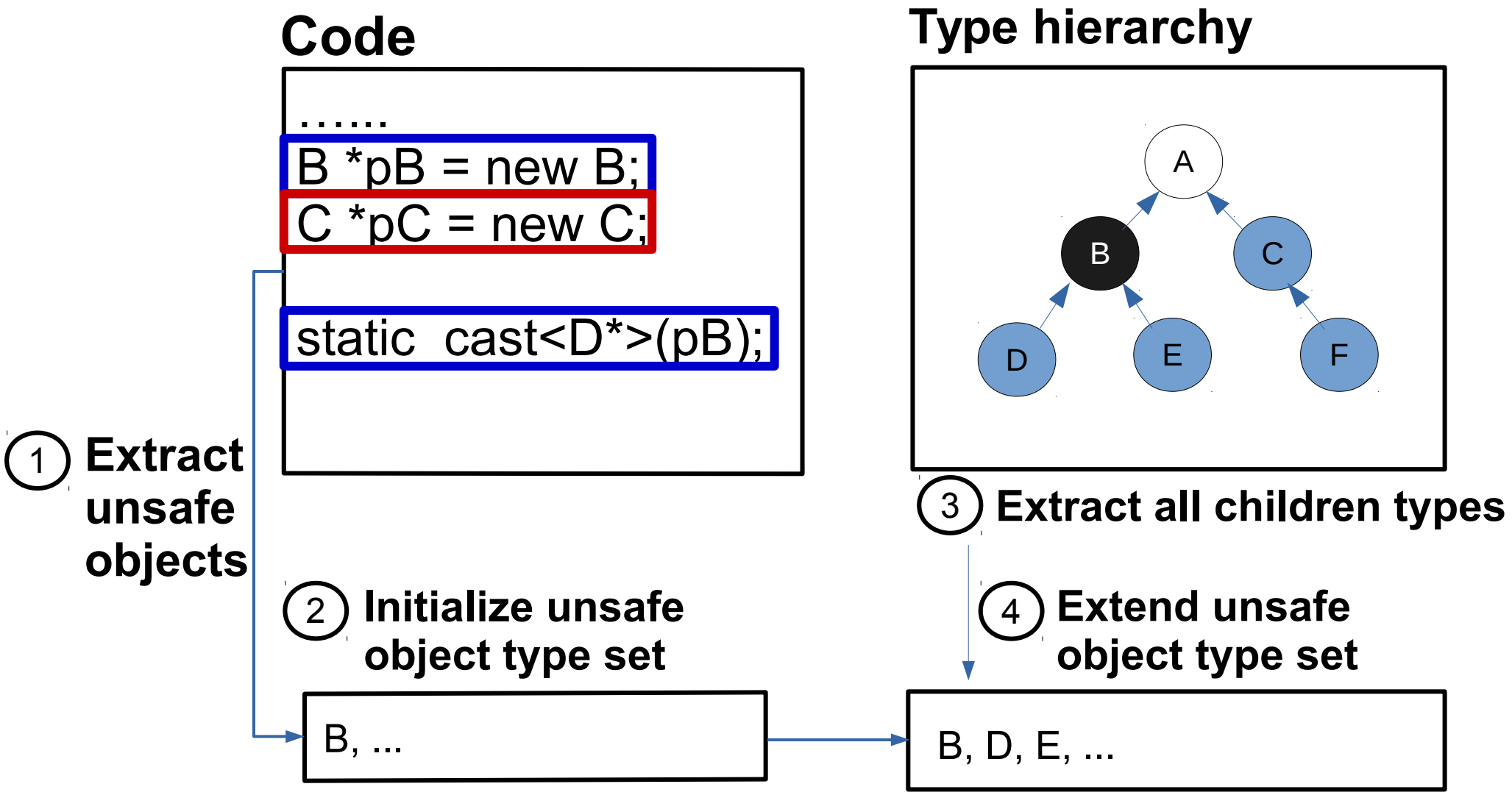
Optimized ObjTypeMap

Program	Benchmark	Allocated objects			Fast-path Hit ratio (%) (update)	Fast-path hit ratio (%) (lookup)
		stack	heap	global		
SPEC CPU 2006	Omnetpp	1m	478m	0.001m	100.00	100.00
	Xalancbmk	3,150m	45m	0.003m	100.00	100.00
	DealII	497m	283m	0m	99.99	100.00
	Soplex	21m	639m	197m	99.69	100.00
	Povray	-	-	-	-	-
	Astar	-	-	-	-	-
	Namd	-	-	-	-	-
Firefox	Octane	593m	7m	0.125m	98.82	98.65
	Drom-js	2,775m	11m	0.125m	99.65	98.43
	Drom-dom	34,900m	607m	0.125m	99.71	94.10

Selective Object Tracing

- Unsafe objects: casting related objects
  - need to keep track of them
- Safe objects: never used for casting
  - do not need to keep track of them

Collect Unsafe Object Set



Only Tracing Unsafe Objects

Program	Benchmark	# of objects	Safe objects ratio (%)
SPEC CPU 2006	Omnetpp	480m	54.77
	Xalancbmk	3,196m	99.42
	DealII	781m	83.81
	Soplex	858m	97.87
	Povray	6,550m	100.00
	Astar	28m	100.00
	Namd	2m	100.00
Firefox	Octane	600m	44.11
	Drom-js	2,491m	40.26
	Drom-dom	37,538m	21.54

Replace Dynamic Cast

- Replaced dynamic_cast to use our fast check
- Evaluated SPEC CPU2006 C++ benchmarks
  - dealII performs a large number of dynamic_cast (206m)
  - reduced dealII's performance overhead by 4%

Compile Time Verification

```
class T : public S { … };

void safe_casting_ex() {
  S test1;
  S test2[1000];

  static_cast<T*>(&test1);
  static_cast<T*>(test2);

  S *local_obj_ptr = &test1;
  static_cast<T*>(local_obj_ptr);
}
```

Performance Overhead

Program	Benchmark	CaVer	TypeSan	HexType		
		%	%	%	X1	X2
SPEC CPU 2006	Omnetpp	-	49.13	9.69	NA	5.07
	Xalancbmk	29.60	41.35	1.25	23.68	33.08
	DealII	-	78.23	13.13	NA	5.96
	Soplex	20.00	1.16	0.76	26.32	1.53
	Povray	-	0.36	0.34	NA	1.06
	Astar	-	0.16	-0.37	NA	1.00
	Namd	-	26.73	0.80	NA	33.41
Firefox	Octane	45.00	19.37	30.87	1.45	-1.59
	Drom-js	40.00	25.18	25.89	1.55	-1.03
	Drom-dom	55.00	97.15	126.03	-2.29	-1.30

Outline

- Motivation
- Type Confusion Problem
- **HexType**
  - Design
  - Optimization
  - **Coverage**
- Conclusion

Increase Detection Coverage

- The state of the art still has low coverage
  - Firefox 10% ~ 45%

- Assume that objects are allocated individually
  - Malloc or C++‘s operator new

- Need to handle allocation using memory pools

Typecasting Coverage

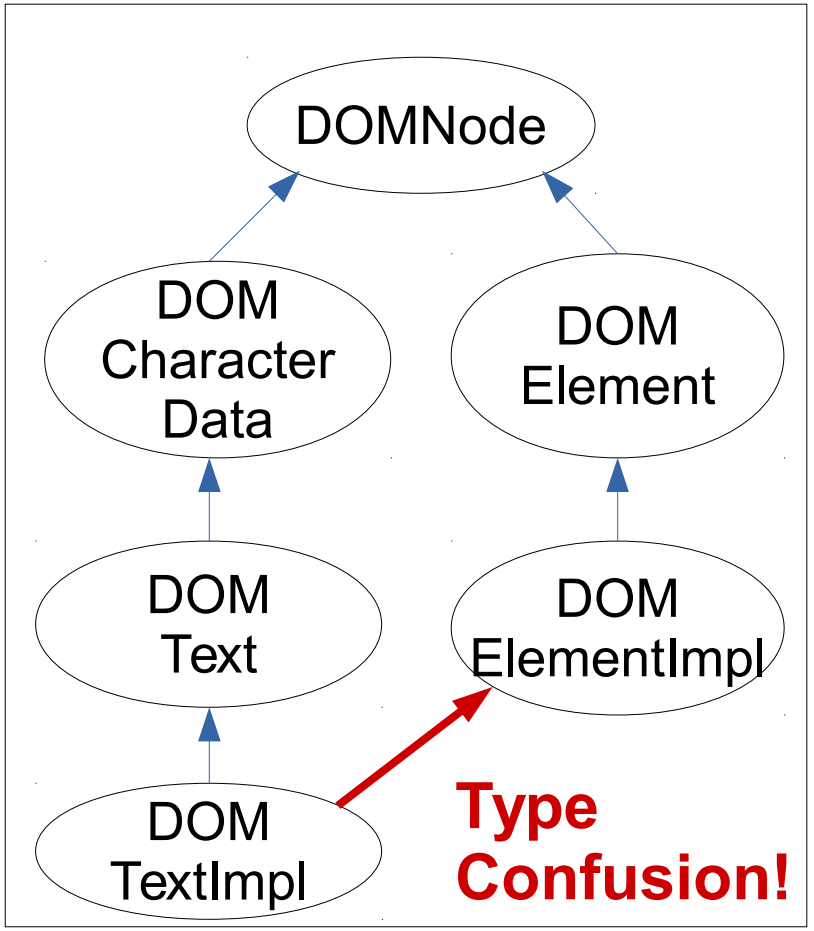
Program	Benchmark	# of casting	Type San	HexType	
			%	%	X
SPEC CPU 2006	Omnetpp	2,014m	1	1	1
	Xalancbmk	283m	0.9	1	1.1
	DealII	3,596m	1	1	1
	Soplex	0.209m	1	1	1
	Povray	0	-	-	-
	Astar	0	-	-	-
	Namd	0	-	-	-
Firefox	ff-octane	623m	0.1	0.7	7
	ff-drom-js	4,229m	0.2	0.8	4
	ff-drom-dom	10,786m	0.5	0.9	1.8

Newly Discovered Vulnerabilities

- Discovered four new type confusion vulnerabilities
  - Qt base library
  - Apache Xerces-C++

- Confirmed and patched by the developers

New Type Confusion Bug
(Apache Xerces-C++)

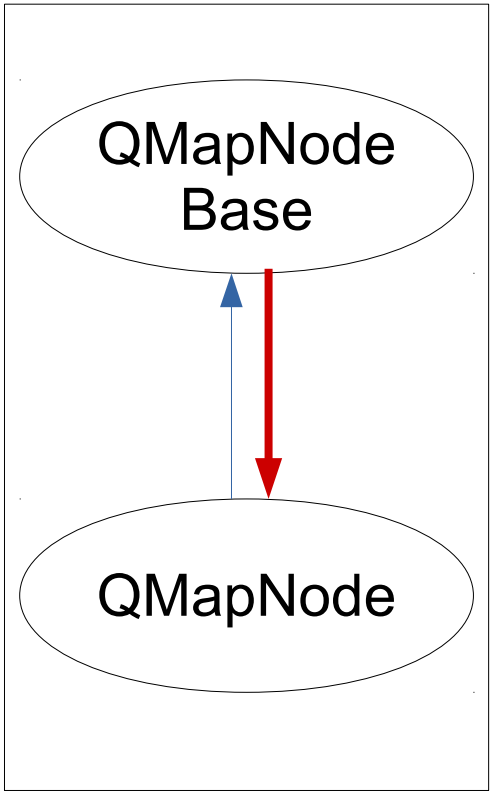


== HexType Type confusion Report ==

xercesc/dom/impl/DOMCasts.hpp Line: 111

[From] DOMTextImpl
[To] DOMElementImpl

New Type Confusion Bug (QT)



== Type confusion Report ==

/.../qt5/QtCore/qmap.h Line: 189

[From] QMapNodeBase
[To] QmapNode

Type Confusion!

Outline

- Motivation
- Type Confusion Problem
- HexType
- **Conclusion**

Conclusion

- HexType increases detection coverage
- Reduces overhead using several new optimizations
- Discovered four new type confusion bugs
- Open Source at https://github.com/HexHive/HexType

PURDUE
UNIVERSITY®

THANK YOU!

Q&A

Open Source at https://github.com/HexHive/HexType

Placement new / Reinterpret_cast handle

```
template<std::size_t Len, std::size_t Align>
struct aligned_storage {
    typedef struct {
        alignas(Align) unsigned char data[Len];
    } type;
};

template<class T, std::size_t N>
class static_vector
{
 size_t Size = sizeof(T);  size_t Align = alignof(T);
 typename std::aligned_storage<Size, Align>::type d[N];

 public: template<typename ...Args> void insert(Args&&… args)  {
    new(d+m_size) T(std::forward<args>…);
  }

  const T& operator[](std::size_t pos) const {
   return &reinterpret_cast<const T*>(d+pos);
  }
};
```

Type Confusion Report

```
P gObj; // declare global variable

int main() {
    .....
    // stack object testing
    P sObj;
    P* spObj = &sObj;
    // Type Confusion! (line 25)
    C* sTest = static_cast<C*>(spObj);

    // global object testing
    P* gpObj = &gObj;
    // Type Confusion! (line 29)
    C* gloTest = static_cast<C*>(gpObj);

    // heap object testing
    P* hpObj = new P;
    // Type Confusion! (line 33)
    C* hTest = static_cast<C*>(hpObj);
    .....
}
```

```
== HexType Type confusion Report ==
FileName : ./example.cpp Line: 25 Column 13
[From] class.P (hashValue: 3110715001)
[To] class.C (hashValue: 1037565863)
(Call Stack Info)
0x42203d: (__type_casting_verification+0x6ed)
0x4245b7: (main+0x77)
0x7f520087df45: (__libc_start_main+0xf5)
0x403274: (_start+0x29)

== HexType Type confusion Report ==
FileName : ./example.cpp Line: 29 Column 15
[From] class.P (hashValue: 3110715001)
[To] class.C (hashValue: 1037565863)
(Call Stack Info)
0x42203d: (__type_casting_verification+0x6ed)
0x42460d: (main+0xcd)
0x7f520087df45: (__libc_start_main+0xf5)
0x403274: (_start+0x29)

== HexType Type confusion Report ==
FileName : ./example.cpp Line: 33 Column 13
[From] class.P (hashValue: 3110715001)
[To] class.C (hashValue: 1037565863)
(Call Stack Info)
0x42203d: (__type_casting_verification+0x6ed)
0x42469e: (main+0x15e)
0x7f520087df45: (__libc_start_main+0xf5)
0x403274: (_start+0x29)
```