

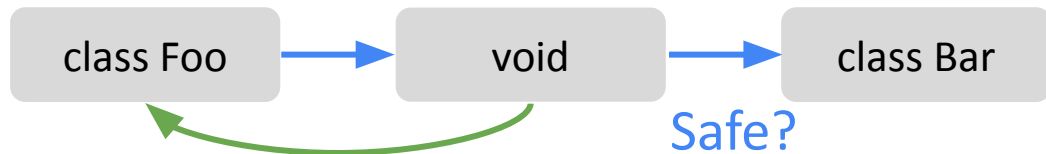
Sourcerer: channeling the `void`

Nicolas Badoux, Flavio Toffalini, and Mathias Payer

Motivation: Unrelated Type Confusion in C++

C++ allows for use of a “generic” type: void

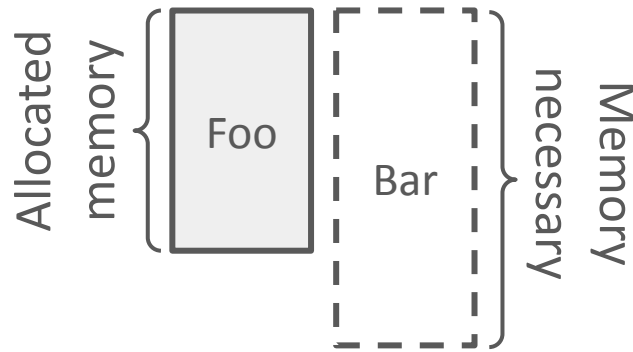
Cast to void* is always allowed



What can you do with the resulting pointer?

- Cannot dereference it
- **Have to** cast it back to the **original type**

Casting to another type breaks memory safety.



Sourcerer Solution

What: Sourcerer is a sanitizer to validate casts (i.e., detect invalid ones)

How: Add type information to **all** object destinations of a cast

What's new? Sourcerer covers **100% of the casts by design**

Technical details:

Inline information (RTTI) reduces the performance cost

Sourcer extends RTTI to all **classes, objects, unions, and w/ templates**

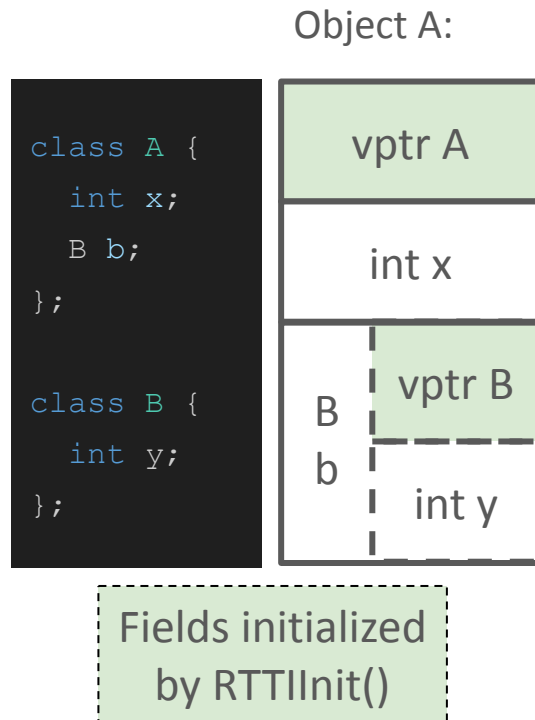
Sourcerer Design

We propose a new routine (RTTIInit) that initializes **only the type**

RTTIInit overcomes technical limitations that affect other C++ dialects (e.g., type++)

RTTIInit supports unions

- Calls to RTTIInit at each union member assignment
- A constructor would have overwritten data



Sourcerer Implementation

Ported the compiler of type++ to LLVM 19

Sourcerer allows for dialect simplification compared to original type++

RTTIInit implemented as dumbed down version of constructors

- Only retain the logic related to the RTTI

Evaluation: Performance

Almost an order-of-magnitude less overhead

SPEC CPU		Sourcerer	EffectiveSan
Runtime	Avg.	5.14%	49%
	Max.	16.99%	~440%
Memory	Max.	104.89%	-

Overhead caused by:

- Type checks: in particular failing ones (when benchmarking)
- RTTIInit: negligible
- ABI change → cache effectiveness: dominant source (~80%)

Sourcerer Performance: Caste Study Astar

Worst performance overhead of Sourcerer on SPEC CPU benchmarks

- 16.99% compared to the baseline
- Up to 100% memory overhead

Caused by the doubling in size of `pointt`

- Used in tight loop of `flexarray::add`
- Breaks cache optimization thought by the developer

Remediation through either:

- Replacing the unrelated cast causing the instrumentation
- Removing verification for this object type

Evaluation: Porting Effort & Security

5x more object types verified on the SPEC CPU benchmarks

Only 20% more porting effort for a total of 450 LoC changed (out of 2MLoC)

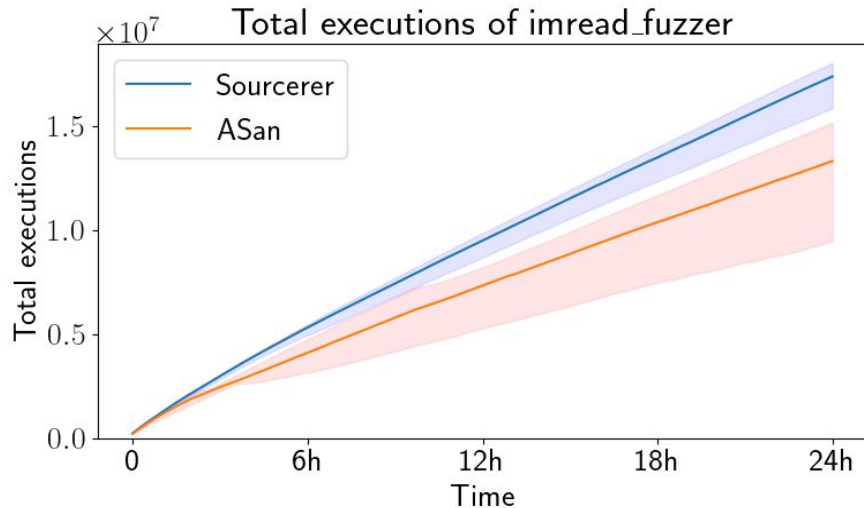
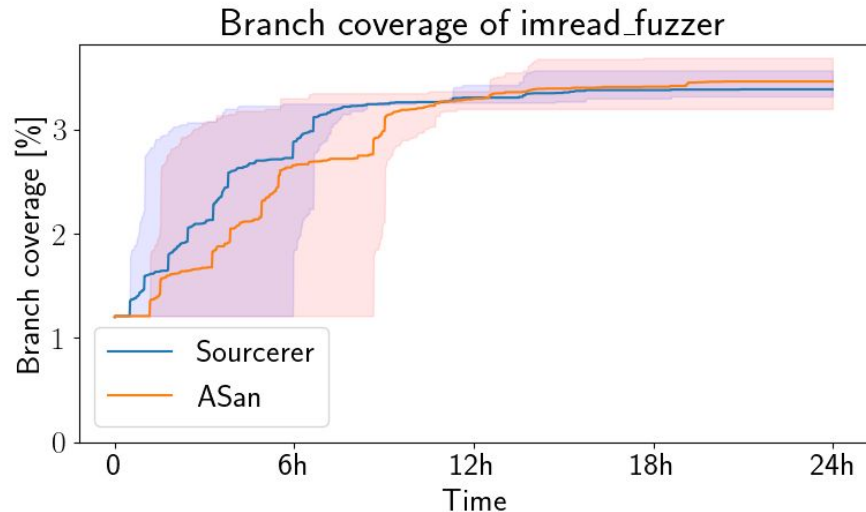
Sourcerer identified 30 new type confusion bugs (and all known ones)

- Developer relying on undefined behavior for objects with a similar layout

Evaluation: Fuzzing for Type Confusions

Similar coverage to sanitizers for other vulnerabilities

Three unrelated type confusion bugs in OpenCV (1.3 MLoC)



Fuzzing Limitations

Our efforts were impeded by:

- Lots of unrelated type confusion accepted by the developer
 - “It works” 🙄
 - Still based on undefined behavior
- Sourcerer is designed to be a sanitizer, not a mitigation, as it cannot always recover after a type confusion (details in the paper)

Sourcerer: channeling the void



Legacy code still relies heavily on void* casts



Complete sanitizer, including unions & unrelated casts



RTTIInit: efficient initializer for inline type information



Low-overhead sanitizer: 5.14% on average



Possibility to reduce overhead with localized changes



First fuzzing campaign targeting type confusion bugs



Artifact: github.com/HexHive/Sourcerer



hexhive