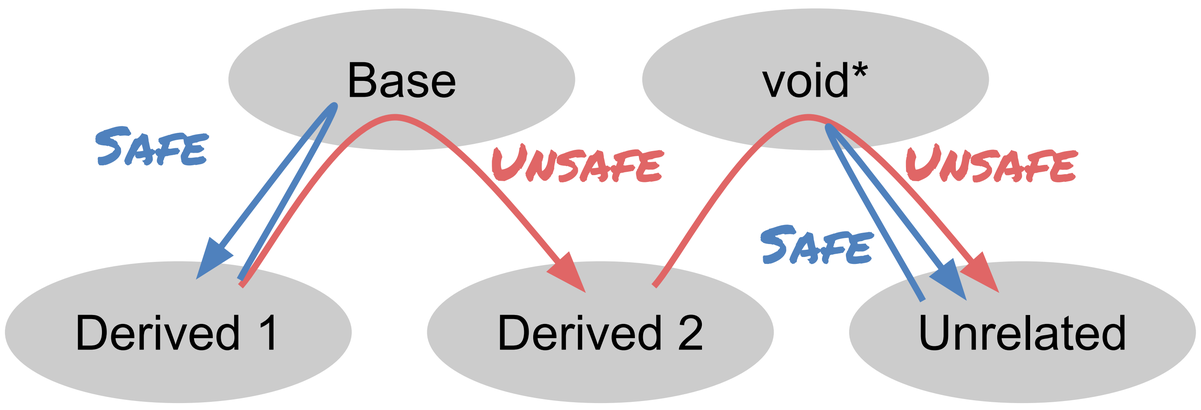
type++: Prohibiting Type Confusion With Inline Type Information

Nicolas Badoux, Flavio Toffalini, Yuseok Jeon, Mathias Payer
EPFL EPFL, RUB UNIST EPFL

Artifact Evaluated — NDSS Symposium — Available — Functional — Reproduced

"type++ is a dialect of C++ that enforces type safety."

Type confusion in C++



Only **`dynamic_cast`** are safe

CVEs in and many more…

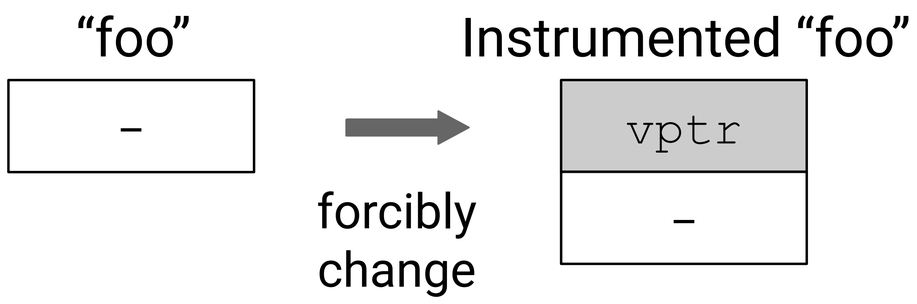
type++ dialect: guarantees

- Every object has a type associated inline
- Enabling dynamic checks at run time
- Optimization: Only classes cast are typed
- Leverage standard *Run Time Type Information* (RTTI) to instrument classes

type++ remove all risk of derived type confusion in C++

Design & C++ ABI incompatibilities

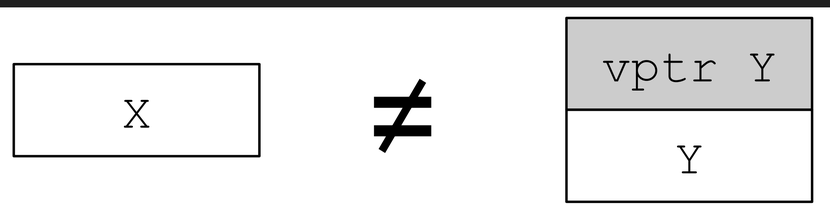
- Call a default constructor to initialize the vptr for non-polymorphic object
- Listing of custom allocator
- Wrapper to communicate with external code



⚠ Induces limited changes to the C++ ABI

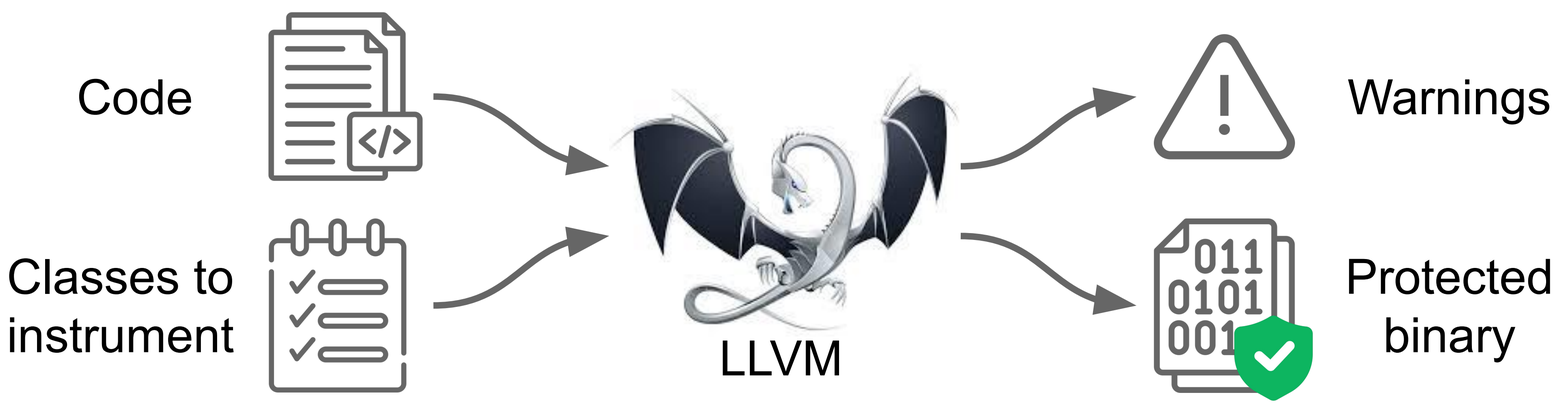
- Interactions with `sizeof()`, *e.g.*, SFINAE expressions

```
static_assert(sizeof(X) == sizeof(Y));
```



- Interactions with non-type++ code through a wrapper or by not instrumenting theses classes

Implementation & Usage



GitHub
HexHive/typepp

Evaluation

Benchmarks: SPEC CPU 2006 & 2017

- **0.98% performance overhead**
- 28x more casts protected than LLVM-CFI
- **122 type confusions found, 14 new**

Use case: partial Chromium

- 1'928 out of 2'099 classes protected
- **229 LoC modified**
- Instrumentation incurs 1.42% overhead
- **94.6% of casts protected**

N. Badoux, F. Toffalini, J. Yuseok & M. Payer, "type++: Prohibiting Type Confusion with Inline Type Information" in Network and Distributed System Security Symposium, NDSS 2025, San Diego, USA, Feb. 24 - 28. DOI: 10.5281/zenodo.13687049.