



IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Quarkslab

Securing every bit of your data

Traceability of the compilation process

CLAP-HiFi-LVP 2023

Bruno MATEU

21st of March



IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Quarkslab

Securing every bit of your data

1 • Context

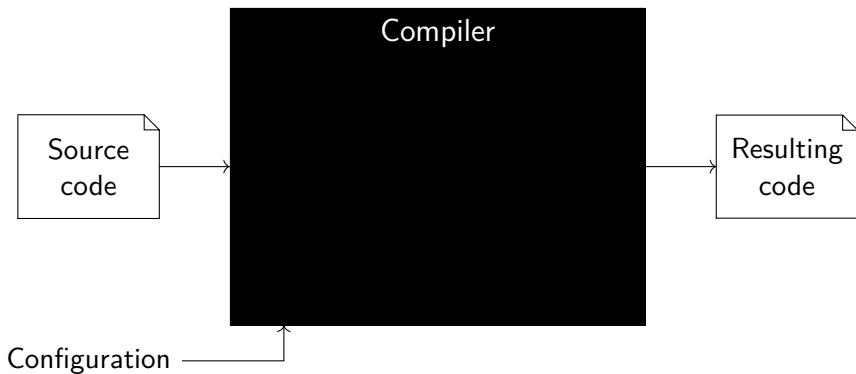
Quarkslab

Securing every bit of your data



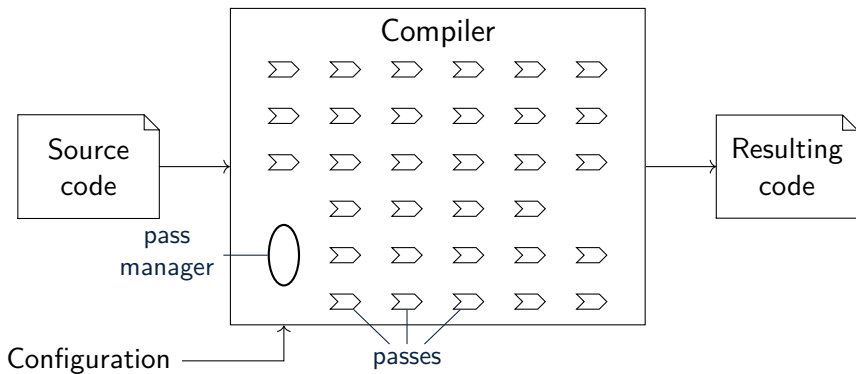
IMT Atlantique

Bretagne-Pays de la Loire
École Mines-Télécom



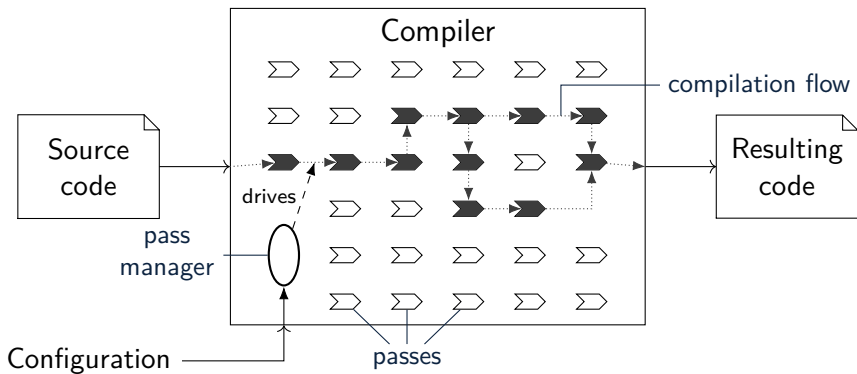
Context

LLVM Compilation and passes



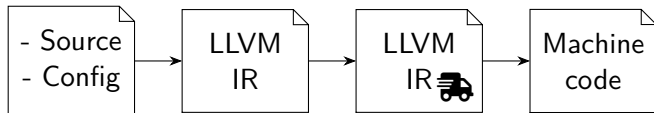
Context

LLVM Compilation and passes



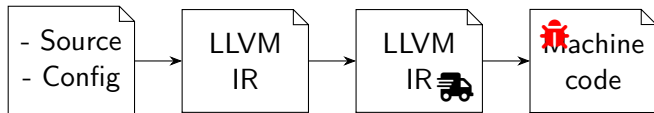
Motivations

Use-case: Debug



Motivations

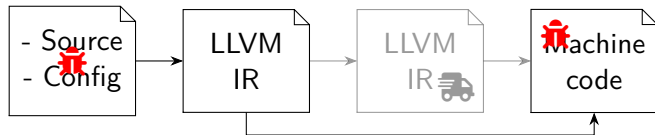
Use-case: Debug



Where is the bug ?

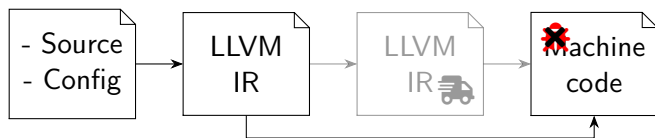
Motivations

Use-case: Debug



Motivations

Use-case: Debug



Where is the bug ?



Backward traceability : the story of instructions, from the produced binary to the source code



Backward traceability : the story of instructions, from the produced binary to the source code



Forward traceability : the story of instructions, from the source code to the produced binary

Traceability inside compilers

- Create a traceability framework
- Not dedicated to a specific usage
- Implemented in LLVM, but designed with a global approach

Traceability inside compilers

- Create a traceability framework
- Not dedicated to a specific usage
- Implemented in LLVM, but designed with a global approach

Trace is optional

- Enable and disable it on demand
- Partial traces are still useful
- No need to implement trace features in every compiler pass to produce useful data

2 • Existing work about traceability

Quarkslab

Securing every bit of your data



IMT Atlantique

Bretagne-Pays de la Loire
École Mines-Télécom

In software engineering

The degree to which a relationship can be established between two or more products of the development process [...] ¹

¹Ravensteijn, “Visual Traceability across Dynamic Ordered Hierarchies”.

²Ibid.

Definitions of Traceability

In software engineering

The degree to which a relationship can be established between two or more products of the development process [...] ¹

Outside of software engineering

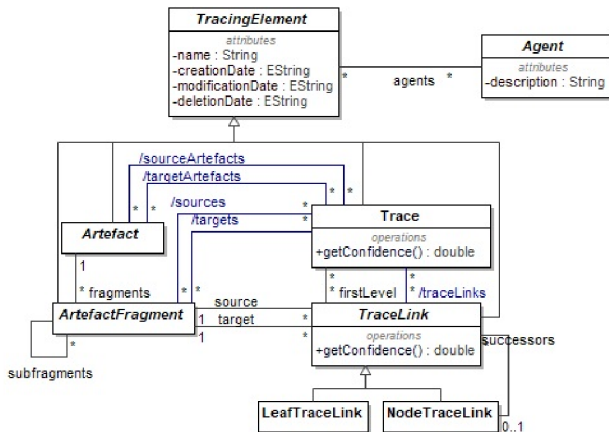
The ability to verify the history, location, or application of an item by means of documented recorded identification. [...] ²

¹Ravensteijn, “Visual Traceability across Dynamic Ordered Hierarchies”.

²Ibid.

Existing trace model

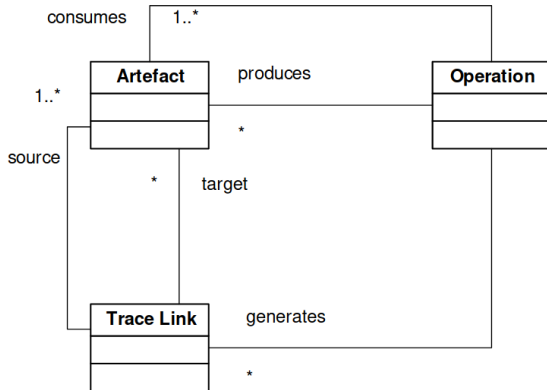
Trace α^3



³Batot, Cabot, and Gérard, "(Not) Yet Another Metamodel For Traceability".

Existing trace model

TEAP⁴



⁴Paige et al., "Building model-driven engineering traceability classifications".

Existing concepts

- ▶ Artefacts: The IR at a given stage of the compilation process
- ▶ Trace links: Called events in my case

Existing concepts

- Artefacts: The IR at a given stage of the compilation process
- Trace links: Called events in my case

Instant: Timeline information

- Has a start and an end
- Describes a time window of the compilation process
- Can be nested

3 • Implementation

Quarkslab

Securing every bit of your data

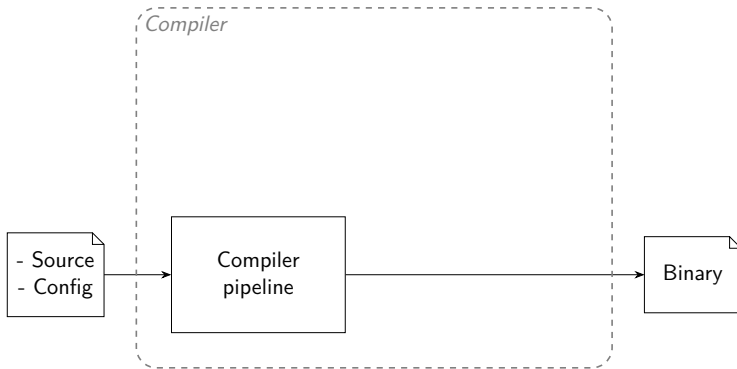
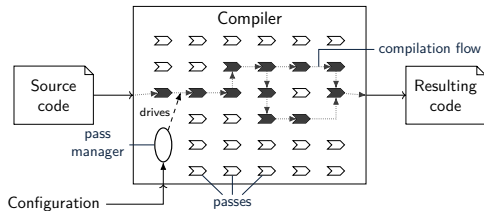


IMT Atlantique

Bretagne-Pays de la Loire
École Mines-Télécom

Trace monitor

The trace API

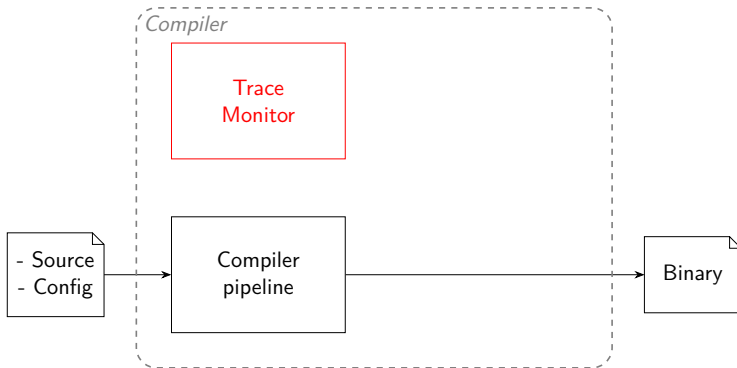


Trace monitor

The trace API

Trace Monitor

- ▶ Inside LLVMCore
- ▶ API to register Instants and Events
- ▶ Accessible from anywhere

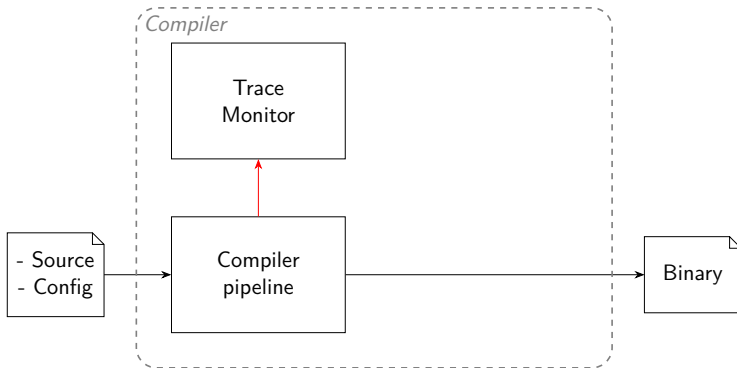


Trace monitor

The trace API

Integration with LLVM codebase

- Modifications to LLVM APIs to use trace instants and events
- New events and instants types can be created to enrich the trace

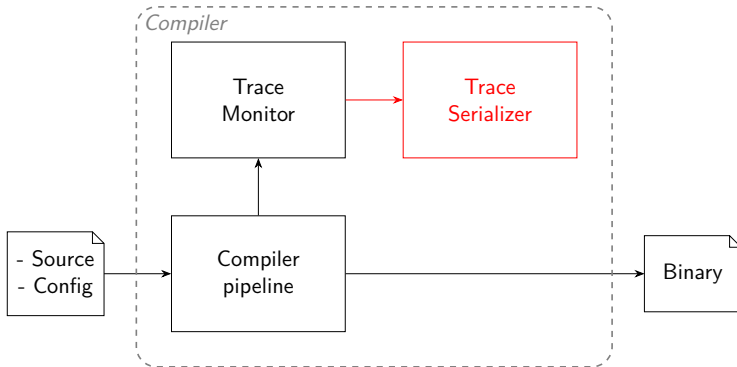


Trace monitor

The trace API

Serializer

- No pre-analysis is done by the serializer
- Easily parseable by external tools

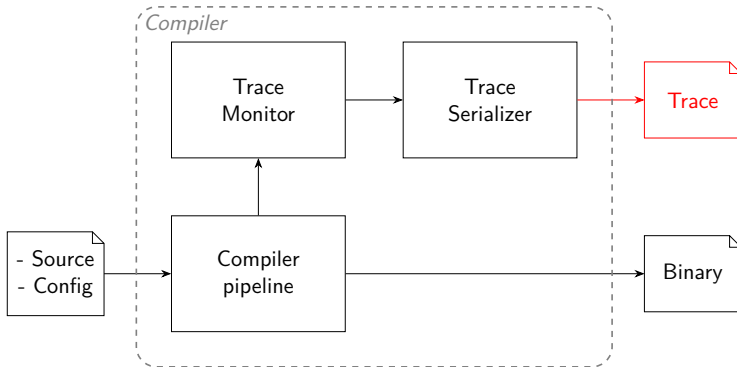


Trace monitor

The trace API

Serializer

- No pre-analysis is done by the serializer
- Easily parseable by external tools

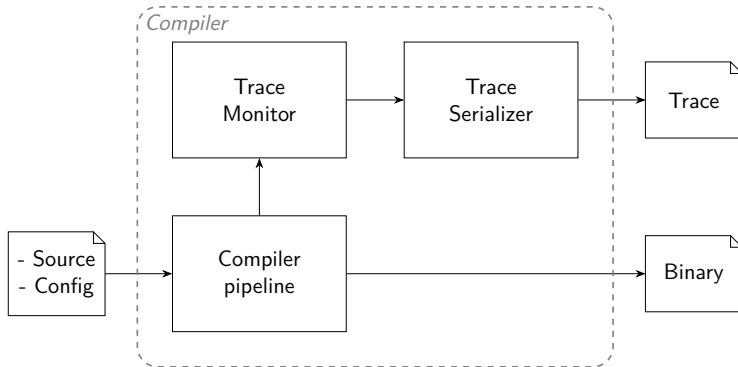


Trace monitor

The trace API

Link with the binary

- Binary and Trace are separate artifacts
- Each *Value* is uniquely identified in the trace



Using the trace

```
0x5555555519a <fib+90>    qid:22 mov    -0x8(%rbp),%edi
0x5555555519d <fib+93>    qid:51 xor    %eax,%eax
0x5555555519f <fib+95>    qid:51 sub    $0x1,%eax
0x555555551a2 <fib+98>    qid:52 add    %eax,%edi
0x555555551a4 <fib+100>   qid:0  call   0x55555555140 <fib>
0x555555551a9 <fib+105>   qid:0  mov    %eax,-0xc(%rbp)
0x555555551ac <fib+108>   qid:24 mov    -0x8(%rbp),%edi
0x555555551af <fib+111>   qid:54 add    $0x3fd1606c,%edi
0x555555551b5 <fib+117>   qid:55 sub    $0x2,%edi
0x555555551b8 <fib+120>   qid:56 sub    $0x3fd1606c,%edi
0x555555551be <fib+126>   qid:0  call   0x55555555140 <fib>
0x555555551c3 <fib+131>   qid:0  mov    %eax,%ecx
0x555555551c5 <fib+133>   qid:0  mov    -0xc(%rbp),%eax
B> 0x555555551c8 <fib+136>   qid:58 sub    $0xcfc58a84,%eax
0x555555551c9 <fib+141>   qid:59 add    %ecx,%eax
0x555555551cf <fib+143>   qid:60 add    $0xcfc58a84,%eax
0x555555551d4 <fib+148>   qid:28 mov    %eax,-0x4(%rbp)
0x555555551d7 <fib+151>   qid:150 lea   0x2e3e(%rip),%rax
0x555555551de <fib+158>   qid:150 mov    (%rax),%eax
0x555555551e0 <fib+160>   qid:151 lea   0x2e39(%rip),%rcx
0x555555551e7 <fib+167>   qid:151 mov    (%rcx),%ecx
0x555555551e9 <fib+169>   qid:152 mov    %eax,%edx
0x555555551eb <fib+171>   qid:152 sub    $0x1,%edx
0x555555551ee <fib+174>   qid:153 imul  %edx,%eax
0x555555551f1 <fib+177>   qid:154 and    $0x1,%eax
0x555555551f4 <fib+180>   qid:155 cmp    $0x0,%eax

Trace
Event: creation of 58
name: %20 = sub 132 %13, -809137532
operands: ['0', '57']
opcode: sub
in instants ['ModuleToFunctionPassAdaptor', 'InstructionsSubstitution']

Event: creation of 59
name: %21 = add 132 %20, %19
operands: ['58', '0']
opcode: add
in instants ['ModuleToFunctionPassAdaptor', 'InstructionsSubstitution']

Event: creation of 74
name: <badref> = add 132 %18, %17
operands: ['50', '0']
opcode: add
in instants ['BogusControlFlow']

Event: delete of 58
name: delete
in instants []

record-ful Thread 0x7ffff7f946 In: fib
(gdb) c
Continuing.

Breakpoint 2, fib (arg=<optimized out>) at debug_ir_-a24d11.ll:46
(gdb) |
```

Trace Model

- ▶ Flexible and on-going process

Trace Model

- ▶ Flexible and on-going process

Implementation

- ▶ Limited to the middle-end for now

Trace Model

- ▶ Flexible and on-going process




Implementation

- ▶ Limited to the middle-end for now

Using traces

- ▶ GDB integration
- ▶ Future automated tools

Bibliography

-  Ravensteijn, W. J. P. van. “Visual Traceability across Dynamic Ordered Hierarchies”. MA thesis. Eindhoven University of Technology, 2011. URL: <https://research.tue.nl/en/studentTheses/visual-traceability-across-dynamic-ordered-hierachies>.
-  Batot, Edouard R., Jordi Cabot, and Sébastien Gérard. “(Not) Yet Another Metamodel For Traceability”. In: *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. Oct. 2021, pp. 787–796. DOI: 10.1109/MODELS-C53483.2021.00125.
-  Paige, R. et al. “Building model-driven engineering traceability classifications”. In: *ECMDA Traceability Workshop (ECMDA-TW)*. Sintef. 2008, p. 49. URL: <https://www.semanticscholar.org/paper/4d83fdf48055ee609ea7bfff0e467e6eae45e0ff>.

Costs

Lua		886kB source, 300kB compiled			
Options		-O0	-O1	-O2	-O3
time (s)	Clang-14	1.16	3.81	4.24	4.42
	Clang-15 patched	17.00	22.36	23.44	23.70
Trace size (MB)		5.80	43.51	48.53	50.04

keepassxc		9.3MB source, 6.9MB compiled			
Options		-O0	-O1	-O2	-O3
time (s)	Clang-14	327.55	371.96	377.75	384.157
	Clang-15 patched	6302.95		7790.97	7721.80
Trace size (MB)		38		250	

LLVM Metamodel

