Digital Watermarks for LLVM Intermediate Representation

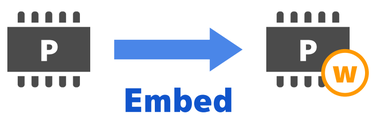
R. Nagayama, L. Chen, H. Inaba Kyoto Institute of Technology

Background

There is little research on software watermarking.
2 basic approach of software watermarking:

1. Modifying the Program Binary

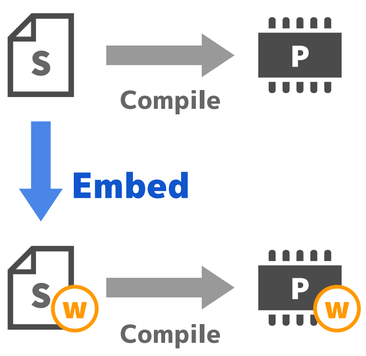
Embedding **by modifying the executable or the bytecode** directly.



- No resistance to **overwrite attacks**
- Depends on **target platform**

2. Modifying the Source Code

Embedding **in the source code** and compiling it.



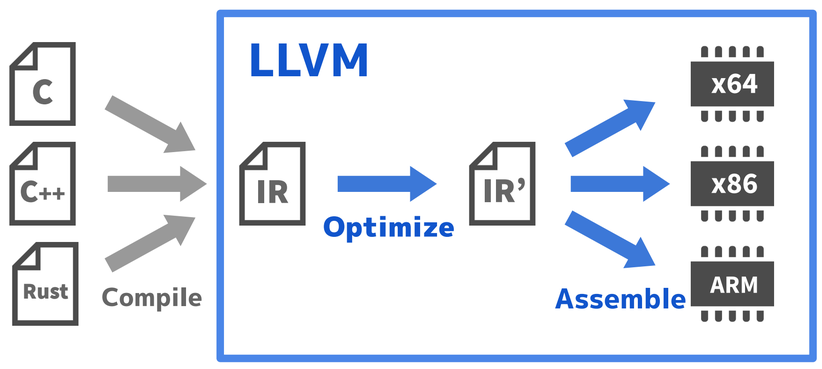
- No resistance to **optimization**
- Depends on **development language**

LLVM

LLVM is a compiler infrastructure that supports native code output to various target platforms.

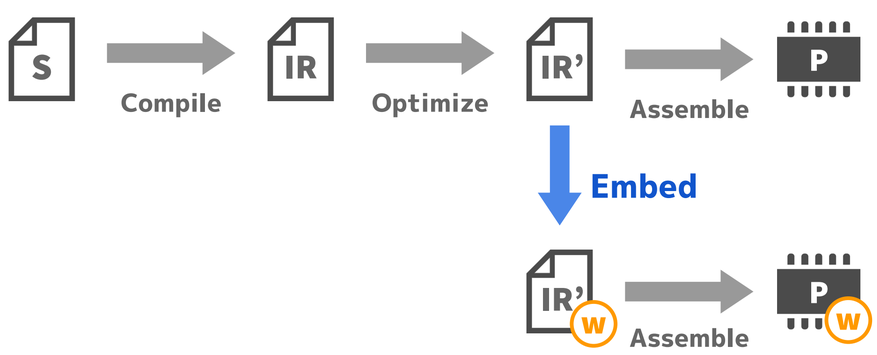
LLVM IR is an intermediate representation (IR) provided by LLVM.

LLVM provides several optimization paths for IRs.



Idea

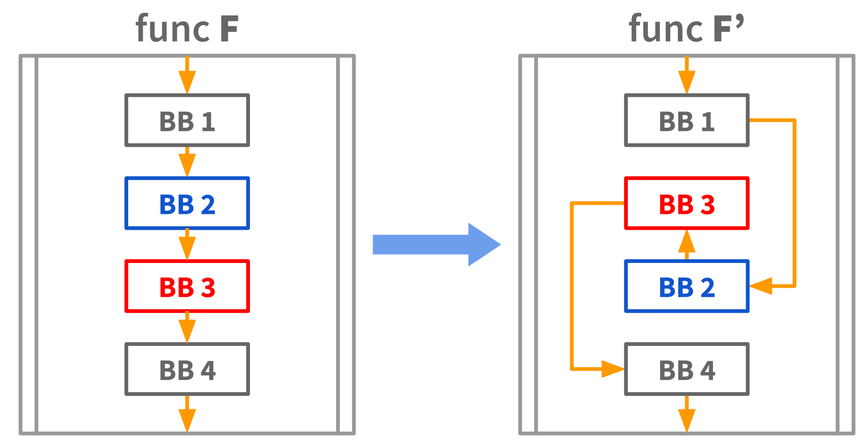
Embedding watermarks **in LLVM IRs**.



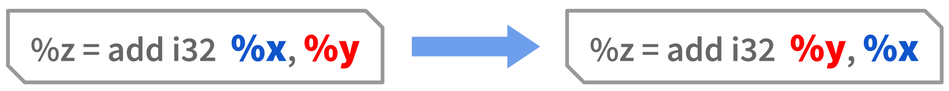
Proposal

We proposed 3 embedding methods.

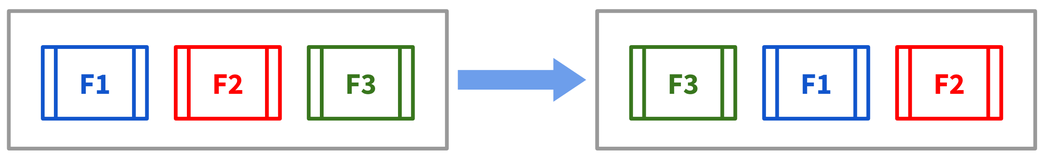
Method-1: Changing the Order of Basic Blocks



Method-2: Swapping instruction operands



Method-3: Changing the order of functions



Evaluation

1. Resistance to Optimization

LLVM has 3 optimization levels.

- IR level optimization (**IRO**)
- Machine code optimization (**MCO**)
- Link time optimization (**LTO**)

	IRO	MCO	LTO
Method-1	✓		
Method-2	✓		
Method-3	✓	✓	

2. Embedded Information Capacity

	Source code size [KB]	by Method-1 [bit]	by Method-2 [bit]	by Method-3 [bit]
tree	98.6	2208	1285	79
cJSON	110.8	1596	1212	213
8cc	200.6	4452	1819	368
zlib	307.9	3840	3511	200
jemalloc	625.2	19524	14600	1398
lua	665.8	11556	7926	1167

Conclusion

1. Resistance

- Resistant to **overwrite attacks**
- Resistant to **IR level optimization**
- (Method-3) Resistant to **machine code optimization**

2. Originality

- Independent of **target platforms**
- Independent of **development languages**