



ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

allvm - Binary Decompile

Sandeep Dasgupta
University of Illinois Urbana Champaign
March 28, 2016



Goal & Motivation

Possible Approaches

Our Approach

Decompile Machine Code \rightarrow LLVM IR

mcsema

Demo



- Research Goal
 - Obtain “richer” LLVM IR than native machine code.
 - Enable advanced compiler techniques (e.g. pointer analysis, information flow tracking, automatic vectorization)
- Motivation
 - Absence of source-code
 - What-you-see-is-not-what-you-execute
 - End-user security enforcement
 - Platform aware optimizations



Goal & Motivation

Possible Approaches

Our Approach

Decompile Machine Code \rightarrow LLVM IR

mcsema

Demo



The 3 Possible Approaches

- Decompile Machine Code \rightarrow LLVM IR
 - Easy to adopt
 - No compiler support needed
- “Annotated” Machine Code \rightarrow LLVM IR
 - Effective reconstruction of higher level IR
 - Minimal compiler support needed
- Ship LLVM IR
 - Benefit: *No loss* of information via conversion to and from binary code.



Decompile Machine Code \rightarrow LLVM IR

- Challenge: Quality
 - Reconstructing code and control flow - much researched.
 - Variable recovery
 - Function & ABI rules recovery



“Annotated” Machine Code \rightarrow LLVM IR

- Challenge:
 - Annotations must be “minimal” & sufficient
 - Annotations must be compiler and IR-independent
 - Adoption



- Challenge:
 - Adoption in Non LLVM based compilers
 - Stable distribution format for shipping
 - Risks to intellectual property
 - Code size bloat



Goal & Motivation

Possible Approaches

Our Approach

Decompile Machine Code \rightarrow LLVM IR

mcsema

Demo



Our Approach

- Long term goal
 - Minimal compiler-independent annotations to reconstruct high-quality IR
- Short term goals
 - ① Experiment with Machine Code \rightarrow LLVM IR, to **understand** the challenges better
 - To select an existing decompilation framework.
 - Experiment with different variable and type recovery strategies
 - ② Design suitable annotations for what cannot be inferred without them



Goal & Motivation

Possible Approaches

Our Approach

Decompile Machine Code \rightarrow LLVM IR

mcsema

Demo



Variable & Function Parameter Recovery

- Benefit
 - Enables many fundamental analysis (Dependence, Pointer analysis)
 - Functional IR
- State of the art
 - Divine
 - State of the art variable recovery
 - Second Write
 - Heuristics for function parameter detection
 - Scalable variable and type recovery
 - TIE
 - Type recovery



Goal & Motivation

Possible Approaches

Our Approach

Decompile Machine Code \rightarrow LLVM IR

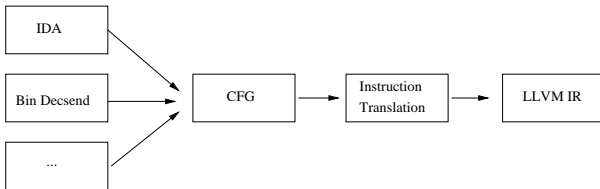
mcsema

Demo



Choosing mcsema

- Functional LLVM IR
- Separation of modules: CFG recovery and CFG \rightarrow LLVM IR
- Actively supported and open sourced





Support & Limitations

- What Works
 - Integer Instructions
 - FPU and SSE registers
 - Callbacks, External Call, Jump tables
- In Progress
 - FPU and SSE Instructions: Not fully supported
 - Exceptions
 - Better Optimizations



Goal & Motivation

Possible Approaches

Our Approach

Decompile Machine Code \rightarrow LLVM IR

mcsema

Demo



mcsema: Demo