mcsema

Statically translate X86 binary to LLVM IR

Supports Windows PE and Linux ELF files
X86 FPU instructions supported
Built with LLVM 3.2, Protocol Buffers, and Boost

Use LLVM transformations and passes on binary code
Analyze X86 binary code with LLVM tools such as:
KLEE, LLBMC, and PAGAI

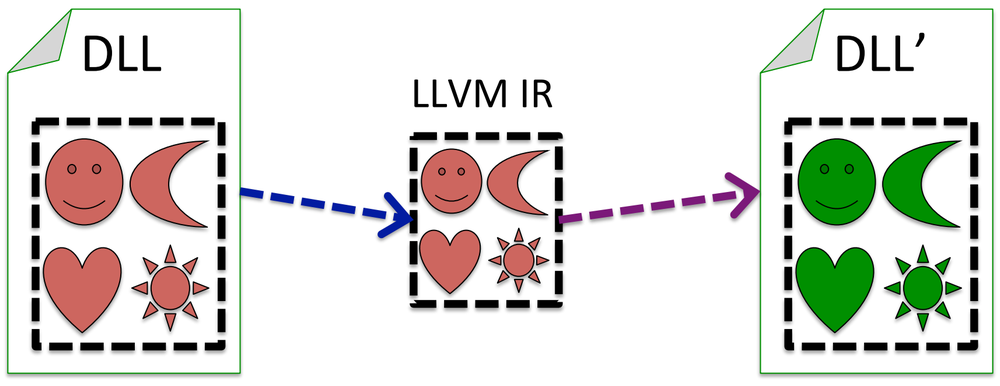
TRAIL OF BITS

Source and documentation at
github.com/trailofbits/mcsema

Features

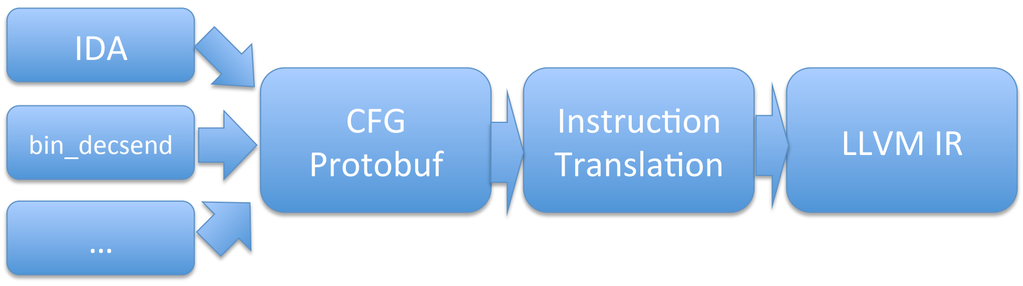
Most X86 instructions
Windows PE and Linux ELF files
Integer instructions
FPU registers
SSE registers
Explicit Flags registers
Callbacks
External Calls
Jump Tables
Data References
SSE instructions (very few)
FPU instructions (some)

Re-Emit Translated X86



Use LLVM optimizations, obfuscation, and security passes
Many "source-only" LLVM tools now work on binary code
Tested Windows Apps running recompiled kernel32.dll

Architecture



Modular Architecture

Designed to translate code from arbitrary sources
CFG recovery separate from translation
Integrate with tools such as INSIGHT or jakstab

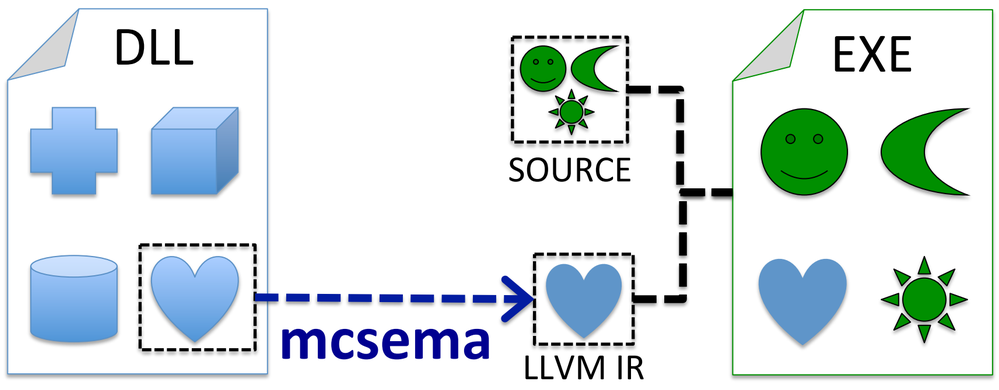
Control Flow Graph Recovery

Control flow graphs specified as Google protocol buffers
Use our CFG recovery tool bin_descend
Use existing tools such as IDA Pro to generate CFG

Translate Instructions

Meticulously implement each X86 instruction as sequence of LLVM IR with the same input and output behavior

Per-Function Translation



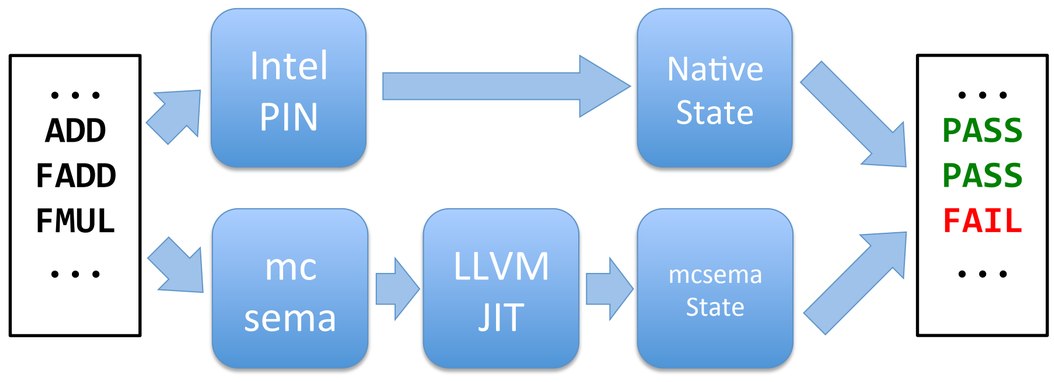
Translate just the functions you want and their dependencies
Reuse specific functions from a library

Calling convention agnostic
Saved register state between function boundaries

Future Improvements

More Linux/ELF support
More instruction translations
Stack variable recovery
Exceptions support
Privileged instructions
More optimizations
More tests
Update LLVM

Unit Tests



Instruction level comparison of translated instructions vs native execution