Decompilation: Structure, Methods, Tools

Adam Sedmík, xsedmi04
Lukáš Wagner, xwagne10

Abstract

Decompilation is process of translating machine code back to original high level programming language. Decompilers can be viewed as compilers, where source and target languages are switched. Although decompilers can be of a different structure compared to traditional compiler. Sometimes, the term decompiler is used for dissasemblers which merely convert binary programs to low level assembly language. This dissasembly is only a single step of the decompilers that attempt to reconstruct original higher level source code.

Decompilation is very difficult process, which is often undecidable; therefore, methods used aim at least for partial decompilation. These methods are divided into stages, giving us the structure of decompiler. The first stage is dissasembly, where machine code instructions can be converted into more easily analyzable intermediate representation. Before conversion to code, there are the syntax and semantic analysis stages, which look for idioms. Idiom is sequences of machine code that can be converted to single complex expression. Next stage is control flow analysis, searching for high level control structures. Following is data flow analysis, including analysis of variables and more complex expressions. Finally there is code generation, where code of target high level language will be generated based on control flow and intermediate instructions.

This presentation focuses on the basic structure of decompilers and explore some of the methods used in analysis. In the end, we list some popular decompilers, for a quick look on how they handle decompilation.