

SUMMARY: A SOUND TYPE SYSTEM FOR SECURE FLOW ANALYSIS [1]

The subject of this paper revolves around secure information flow within programs which contain multiple sensitivity levels. This subject has been studied extensively in the past, and several papers have been published on this matter. Most notably is the work of Denning in secure flow analysis and the lattice model which is an extension of Bell and LaPadula model. Denning has formulated efficient static analysis that uses a compiler to verify secure information flow in programs. However, the soundness of Denning's analysis was not proven. It is required to be sure that if the analysis for a given program on some input succeeds, then the program executes securely. The paper formulates Denning's approach into a type-based approach of the analysis. A type system is a formal system of type inference rules for making judgments about programs. They are used to ensure type correctness and other wide variety of program properties. In the case of this paper, the property of interest is secure information flow. The type system proposed is proven to be sound with respect to a standard programming language semantics for a core deterministic language.

The paper starts by giving an overview of Denning's lattice model. Then the informal and formal treatments of the type system are reviewed. Examples are also given to show how to use the typing rules. The soundness theorem is then proven with respect to a standard semantics for the language. At the end, other works on this subject are discussed with some future directions.

In the lattice model, information flow policies are defined by a lattice (SC, \leq) where SC is a finite set of security classes partially ordered by \leq and may include either secrecy or integrity classes. Two types of flows are defined: explicit and implicit flows. The secure flow type system is composed of the core language and type inference rules. In this case the rules ensure secure flow and not data type compatibility. The core language consists of types which are stratified into two levels, *Data types* (τ) and *phrase types* (ρ). Phrase types include variables (τ *var*) and commands (τ *cmd*). Then some basic secure flow typing rules are defined in order to guarantee secure explicit and implicit flows, similar to the certification rules in the Denning's lattice model.

The main purpose of this paper is to prove the soundness of the type system. Soundness is simplified in that variables in a well-typed program do not interfere with variables at lower security levels. In order to reach the proof the paper proves a series of lemmas. Most importantly are the simple security which applies to expressions and the confinement property which applies to commands. The confinement property mainly assures that a command of type τ *cmd* operates on variables of the same security class τ or higher. This ensures secure implicit flow. Moreover, simple security states in terms of secrecy that when an expression e is evaluated only variables at level τ or lower will have their contents read. Both lemmas are used to prove the soundness of the system which is formulated as a kind of noninterference property.