

SCCI Master 2

A SOUND TYPE SYSTEM FOR SECURE FLOW ANALYSIS

Paper By: Denise Volpano, Geoffrey Smith and
Cynthia Irvine

Presented By: Endri VANGJEL, Marvin TCHOULA NJIA,
Mohamed AL ALI , Mohammed AL MANSOORI



Motivation

- Ensure Secure Information Flow
- Build Basis of provably-secure programming languages

Outline

- Introduction
 - Secure Flow
 - Noninterference
- Lattice Model
- Type System
 - Definition and Goal
 - Core Language
 - Inference Rules
- Type Soundness and proof
- Conclusions



Introduction

- Secure Flow
- Non-interference

Lattice Model

- Bell and Lapadula Extension
- Lattice is a couple (SC, \leq)
 - Security Classes
 - Secrecy
 - Integrity
- Certification Conditions
 - Explicit Flows
 - Implicit Flows

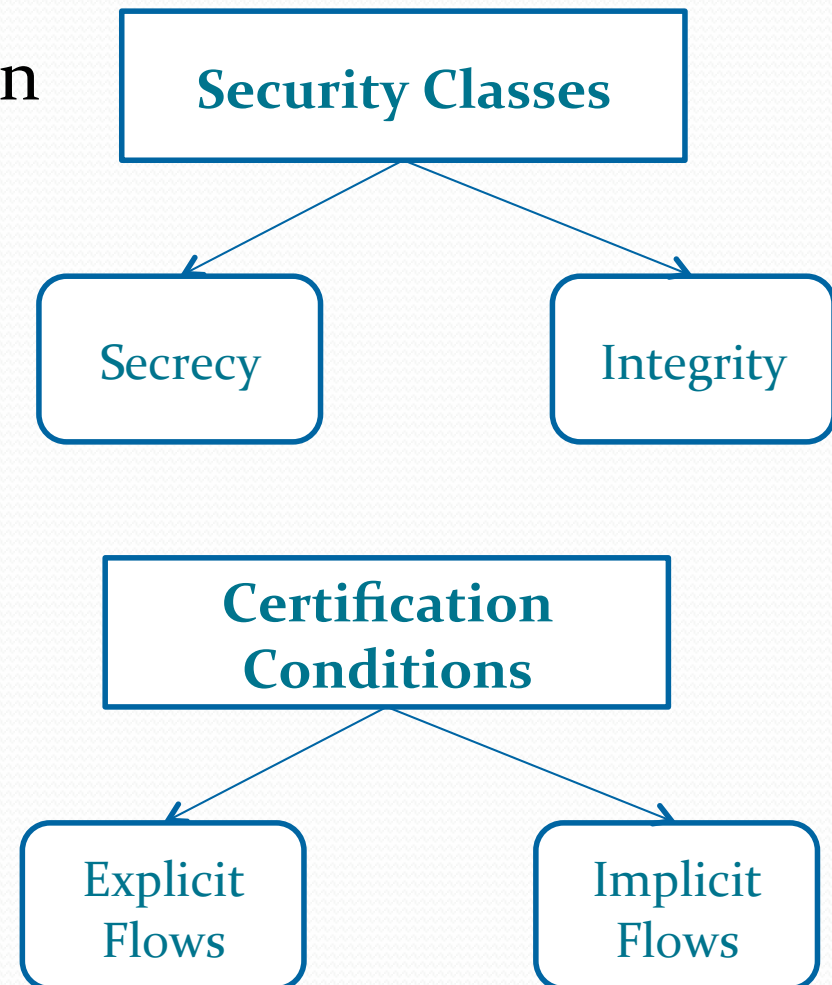


Figure 1: Lattice Model



Lattice Model

Type System

Definition

Goal

Core Language

Secure Flow Types

Secure Flow Typing Rules

Inference Rules

Type Soundness

Type System

- Definition
 - Formal system
 - Contains Type Inference Rules
 - Focused on Secure Info. Flow
- Goal
 - Check Program's Correctness
 - Separate Security Policies from Algorithms

Core Language

- Block-structured language

(phrases) $p ::= e \mid c$
(expressions) $e ::= x \mid l \mid n \mid e + e' \mid e - e' \mid e = e' \mid e < e'$
(commands) $c ::= e := e' \mid c; c' \mid \mathbf{if} \ e \ \mathbf{then} \ c \ \mathbf{else} \ c' \mid$
 $\mathbf{while} \ e \ \mathbf{do} \ c \mid \mathbf{letvar} \ x := e \ \mathbf{in} \ c$

- Core Language Types

(data types) $\tau ::= s$
(phrase types) $\rho ::= \tau \mid \tau \ \mathit{var} \mid \tau \ \mathit{cmd}$

Type System

- Secure Flow Typing Rules
 - Example

$$\frac{\begin{array}{l} \gamma \vdash e : \tau \text{ var,} \\ \gamma \vdash e' : \tau \end{array}}{\gamma \vdash e := e' : \tau \text{ cmd}}$$

Inference Rules

- Rules define Type System
- Typing Judgment
 - Example:

$$\lambda; \gamma \vdash p : \rho$$

λ : location typing

γ : identifier

typing

p : phrase

ρ : phrase type

Inference Rules

(INT)	$\lambda; \gamma \vdash n : \tau$	
(VAR)	$\lambda; \gamma \vdash x : \tau \text{ var}$	if $\gamma(x) = \tau \text{ var}$
(VARLOC)	$\lambda; \gamma \vdash l : \tau \text{ var}$	if $\lambda(l) = \tau$
(ARITH)	$\frac{\lambda; \gamma \vdash e : \tau, \lambda; \gamma \vdash e' : \tau}{\lambda; \gamma \vdash e + e' : \tau}$	
(R-VAL)	$\frac{\lambda; \gamma \vdash e : \tau \text{ var}}{\lambda; \gamma \vdash e : \tau}$	

Figure 2: Inference Rules

Type Soundness

- Important Lemmas
 - Simple Security
 - Applies to expressions
 - Confinement
 - Applies to commands

Type Soundness and proof

- Soundness means that variable in a well typed program do not interfere with variables at low security levels.

- - (a) $\lambda \vdash c : \rho$
 - (b) $\mu \vdash c \Rightarrow \mu'$
 - (c) $\nu \vdash c \Rightarrow \nu'$
 - (d) $\text{dom}(\mu) = \text{dom}(\nu) = \text{dom}(\lambda)$
 - (e) $\nu(l) = \mu(l) \quad \forall l \text{ such that } \lambda(l) \leq \tau$

Then $\nu'(l) = \mu'(l) \quad \forall l \text{ such that } \lambda(l) \leq \tau$

Idea of the proof

- The soundness of the system is established with respect to natural semantic which consist of a set of evaluation rules.

$$\text{(BASE)} \quad \mu \vdash n \Rightarrow n$$

$$\text{(CONTENTS)} \quad \mu \vdash l \Rightarrow \mu(l) \quad \text{if } l \in \text{dom}(\mu)$$

$$\text{(ADD)} \quad \frac{\mu \vdash e \Rightarrow n, \quad \mu \vdash e' \Rightarrow n'}{\mu \vdash e + e' \Rightarrow n + n'}$$

$$\text{(UPDATE)} \quad \frac{\mu \vdash e \Rightarrow n, \quad l \in \text{dom}(\mu)}{\mu \vdash l := e \Rightarrow \mu[l := n]}$$

$$\text{(SEQUENCE)} \quad \frac{\mu \vdash c \Rightarrow \mu', \quad \mu' \vdash c' \Rightarrow \mu''}{\mu \vdash c; c' \Rightarrow \mu''}$$

Conclusions

- The paper
 - Formulates Denning's Analysis
 - Proves System Type Soundness
 - Builds Basis of provably-secure programming languages

Questions

- Do you have any Questions

