Program Analysis
Information Flow Analysis (Part 2)
Outline

1. Introduction
2. Information Flow Policy
3. Analyzing Information Flows

Mostly based on these papers:

Lattice of Security Labels

How to represent different levels of secrecy?

- Set of security labels
- Form a universally bounded lattice
Lattice: Example

High
\downarrow
Low

Top secret
\downarrow
Secret
\downarrow
Confidential
\downarrow
Public

ABC
\downarrow
AB \quad BC \quad AC
\downarrow
A \quad B \quad C
\downarrow
\emptyset

(Arrow connects more secret class to less secret class.)
Universally Bounded Lattice

Tuple \((S, \rightarrow, \perp, T, \oplus, \otimes)\)

where \(S\) is set of security classes

\[\{ABC, AB, AC, BC, A, B, C, \emptyset\}\]

\(\rightarrow\) is partial order (see figure)

\(\perp\) is lower bound \(\emptyset\)

\(T\) is upper bound \(ABC\)

\(\oplus\) is least upper bound operator, \(S \times S \rightarrow S\)

("combine two pieces of information")

union, e.g., \(AB \oplus A = AB\)

\(\emptyset \oplus AC = AC\)

\(\otimes\) is greatest lower bound operator, \(S \times S \rightarrow S\)

intersection, e.g., \(ABC \otimes C = C\)
Quiz: Which of the following are universally bounded lattices?

1. A
2. Foo
3. A
4. 1

D ⊕ E = ?

- three common upper bounds: \{A, B, C\}
- but none is the least upper bound

no upper bound (infinite)
Information Flow Policy

Policy specifies secrecy of values and which flows are allowed:

- Lattice of security classes
- Sources of secret information
- Untrusted sinks

Goal:
No flow from source to sink
Information Flow Policy

Policy specifies *secrecy of values* and which *flows* are allowed:

- Lattice of security classes
- Sources of secret information
- Untrusted sinks

Goal:
No flow from source to sink

```javascript
var creditCardNb = 1234;
var x = creditCardNb;
var visible = false;
if (x > 1000) {
  visible = true;
} 
```
Information Flow Policy

Policy specifies **secrecy of values** and which **flows** are allowed:

- Lattice of security classes
- Sources of secret information
- Untrusted sinks

**Goal:**
No flow from source to sink

```javascript
var creditCardNb = 1234;
var x = creditCardNb;
var visible = false;
if (x > 1000) {
    visible = true;
}
```
"No flow from high to low" is impractical

E.g., code that checks password against a hash value propagates information to subsequence statements

But: This is intended

```javascript
var password = .. // secret
if (hash(password) === 23) {
    // continue normal program execution
} else {
    // display message: incorrect password
}
```
Declassification

- "No flow from high to low" is impractical
- E.g., code that checks password against a hash value propagates information to subsequence statements
  But: This is intended

```javascript
var password = .. // secret
if (hash(password) === 23) {
  // continue normal program execution
} else {
  // display message: incorrect password
}
```

Declassification: Mechanism to remove or lower security class of a value