Program Analysis – Lecture 8
Information Flow Analysis (Part 1)
Outline

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

Mostly based on these papers:

- A Lattice Model of Secure Information Flow, Denning, Comm ACM, 1976
Secure Computing Systems

- Overall goal: Secure the data manipulated by a computing system

- Enforce a security policy
  - Confidentiality: Secret data does not leak to non-secret places
  - Integrity: High-integrity data is not influenced by low-integrity data
Information Flow

- **Goal of information flow analysis:**
  Check whether information from one "place" propagates to another "place"
  - For program analysis, "place" means, e.g., code location or variable

- **Complements techniques that impose limits on releasing information**
  - Access control lists
  - Cryptography
"Places" in program that hold data

Secret information \rightarrow Possible? \rightarrow Untrusted place

Trusted information \rightarrow Possible? \rightarrow Confidentiality

Integrity
Example: Confidentiality

Credit card number should not leak to visible

```javascript
var creditCardNb = 1234;
var x = creditCardNb;
var visible = false;
if (x > 1000) {
    visible = true;
}
```
Example: Confidentiality

Credit card number should not leak to visible

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

- Secret information propagates to `x`.
- Secret information (partly) propagates to `visible`.

```javascript
// Secret information
```
**Example: Integrity**

```
var designatedPresident = "Michael";
var x = userInput();
var designatedPresident = x;
```
Example: Integrity

`userInput` should not influence who becomes president

```javascript
var designatedPresident = "Michael";
var x = userInput();
var designatedPresident = x;
```

Low-integrity information propagates to high-integrity variable
Example: Integrity

`userInput should not influence who becomes president`

```javascript
var designatedPresident = "Michael";
var x = userInput();
if (x.length === 5) {
    var designatedPresident = "Paul";
}
```
Example: Integrity

userInput should not influence who becomes president

```javascript
var designatedPresident = "Michael";
var x = userInput();
if (x.length === 5) {
    var designatedPresident = "Paul";
}
```

Low-integrity information propagates to high-integrity variable
Confidentiality vs. Integrity

Confidentiality and integrity are dual problems for information flow analysis

(Focus of this lecture: Confidentiality)
How to analyze the flow of information?

- Assign to each value some *meta information* that tracks the *secrecy* of the value
- Propagate *meta information* on program operations
Example

```javascript
var creditCardNb = 1234;
var x = creditCardNb;
var visible = false;
if (x > 1000) {
    visible = true;
}
```
Non-Interference

Property that information flow analysis aims to ensure:

Confidential data does not interfere with public data

- Variation of confidential input does not cause a variation of public output
- Attacker cannot observe any difference between two executions that differ only in their confidential input
Outline

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

Mostly based on these papers:

- A Lattice Model of Secure Information Flow, Denning, Comm ACM, 1976
Lattice of Security Labels

How to represent different levels of secrecy?

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

High
↓
Low

Top Secret
↓
Secret
↓
Confidential
↓
Public

(Arrows connect more secret clan with less secret clan.)
Universally Bounded Lattice

Tuple $(S, \rightarrow, \perp, \top, \oplus, \otimes)$

where: $S$ set of security classes

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

$\rightarrow$ partial order of $S$ (see figure)

$\perp$ lower bound $\emptyset$

$\top$ upper bound $ABC$

$\oplus$ least upper bound operator, $S \times S \rightarrow S$

"combine two pieces of information"

union, e.g., $AB \oplus A = AB$, $\emptyset \oplus AC = AC$

$\otimes$ 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. \( \text{Foo} \)
3. \( \text{Bar} \)
4. \( \text{C} \)

\( D \oplus E = ? \)

Three common upper bounds \((A, B, C)\), but none is the least upper bound.

\( 3 \)
\( 2 \)
\( 1 \)
\( 0 \)

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;
}
```
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

- "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
Outline

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

Mostly based on these papers: