Program Analysis – Lecture 8

Information Flow Analysis (Part 1)

Prof. Dr. Michael Pradel

Software Lab, University of Stuttgart

Winter 2019/2020

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
- *Dytan: A Generic Dynamic Taint Analysis Framework*, Clause et al., ISSTA 2007

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

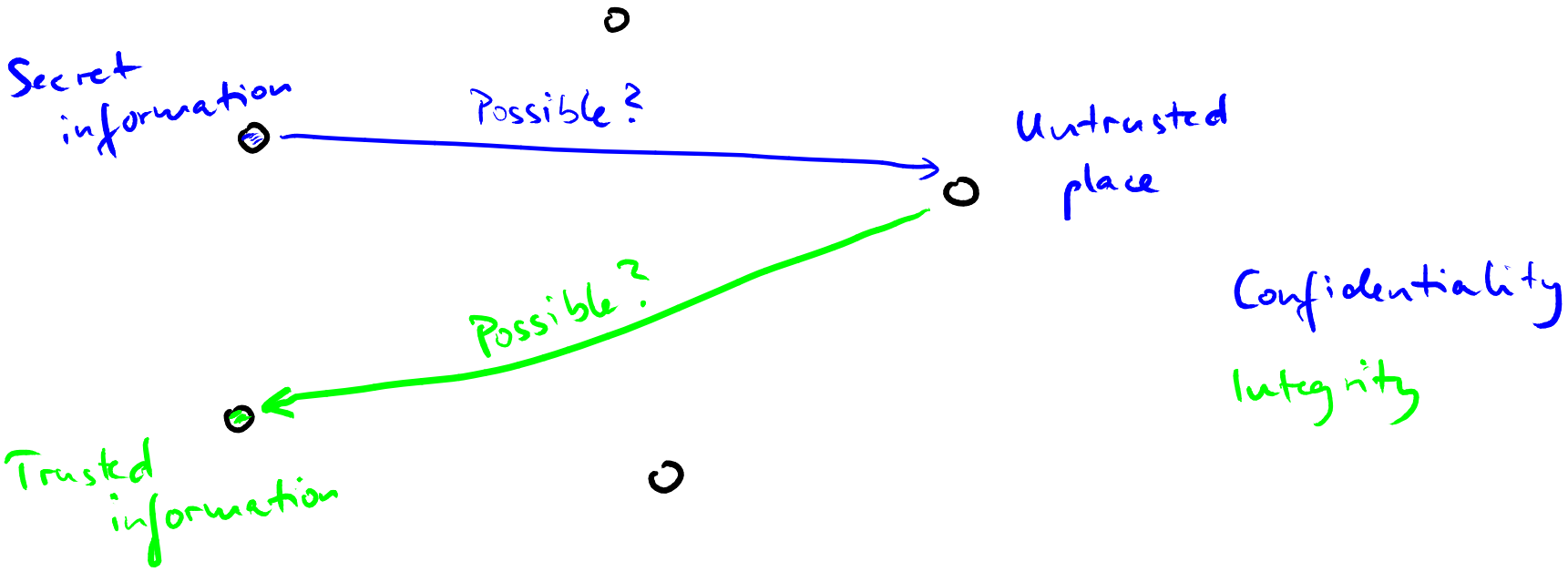
- **Goal of information flow analysis:**

  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



Example: Confidentiality

Credit card number should not leak to `visible`

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

Example: Confidentiality

Credit card number should not leak to `visible`

```
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`

Example: Integrity

`userInput` should not influence who becomes president

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

Example: Integrity

`userInput` should not influence who becomes president

```
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

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

Example: Integrity

`userInput` should not influence who becomes president

```
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)

Tracking Security Labels

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

secret value

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

--- = contains secret value

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:

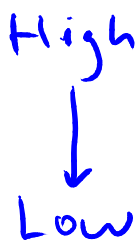
- *A Lattice Model of Secure Information Flow*, Denning, Comm ACM, 1976
- *Dytan: A Generic Dynamic Taint Analysis Framework*, Clause et al., ISSTA 2007

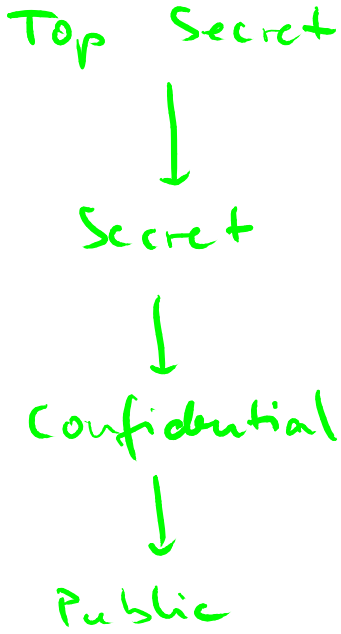
Lattice of Security Labels

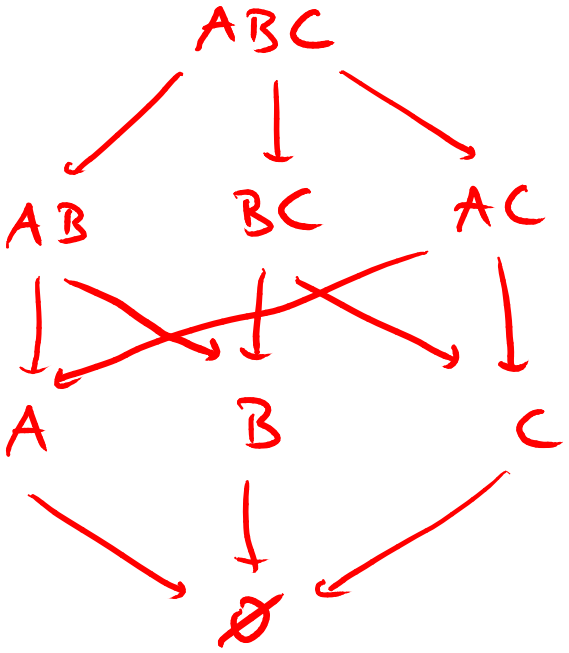
How to represent different levels of secrecy?

- **Set of security labels**
- **Form a universally bounded lattice**

Lattice : Examples







(Arrows connect more secret class with less secret class.)

Universally Bounded Lattice

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

where:

S .. set of security classes

{ABC, AB, AC, BC, A, B, C, ∅}

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

$\bot$ .. lower bound ∅

$\top$ .. upper bound ABC

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

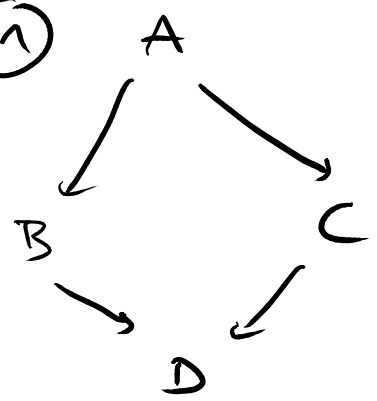
("combine two pieces of information")

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

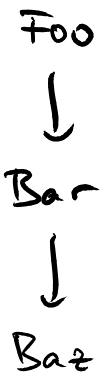
$\otimes$ .. greatest lower bound operator, $S \times S \rightarrow S$

intersection, e.g., $ABC \otimes C = C$

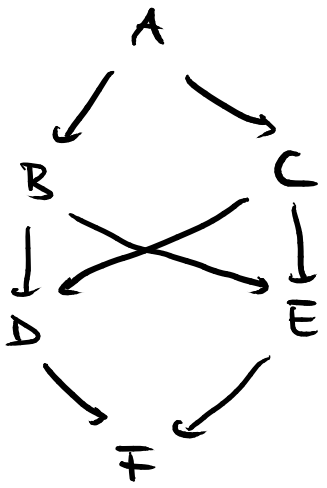
Quiz: Which of the following are universally bounded lattices?

① 

✓

② 

✓

③ 

$D \oplus E = ?$

three common upper bounds (A, B, C), but none is the least upper bound

④ 

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:

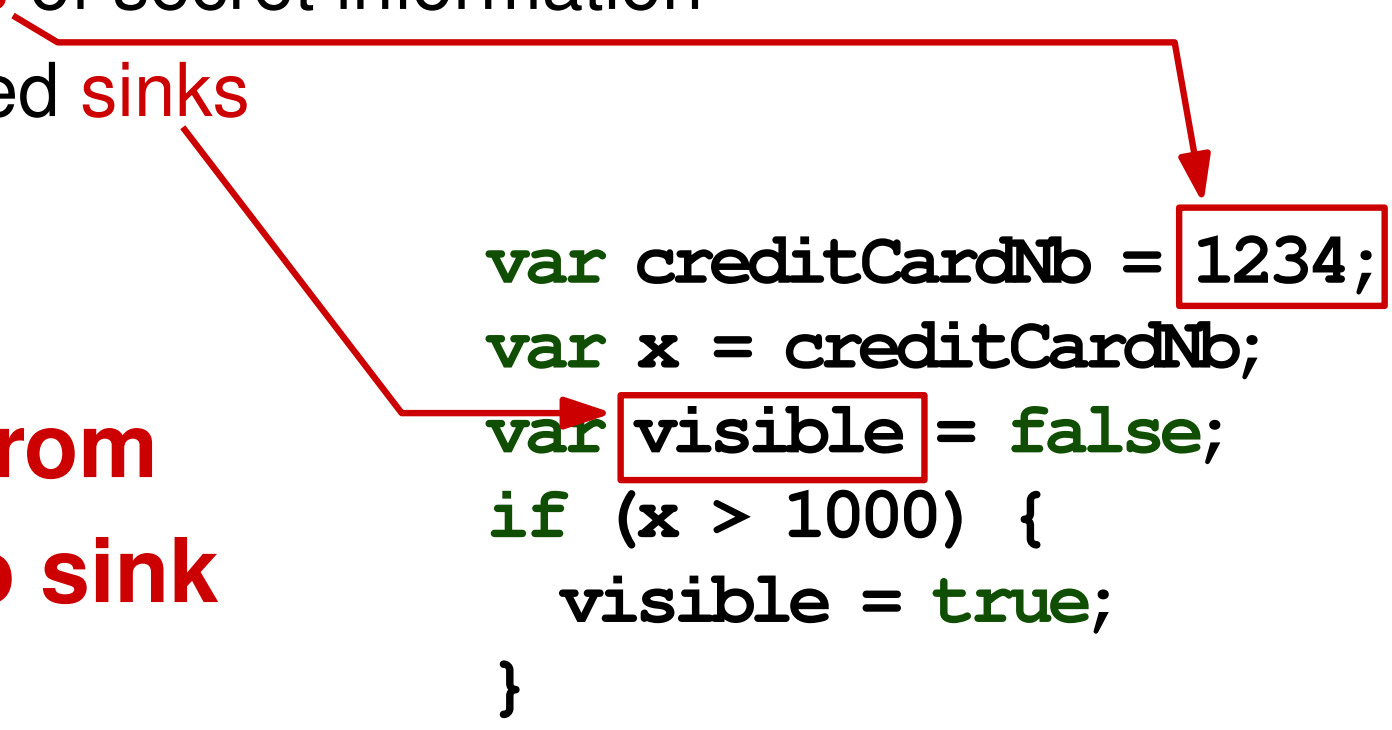
No flow from source to sink

```
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

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

```
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

```
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:

- *A Lattice Model of Secure Information Flow*, Denning, Comm ACM, 1976
- *Dytan: A Generic Dynamic Taint Analysis Framework*, Clause et al., ISSTA 2007