

# Language-based security and Jif

Lujo Bauer

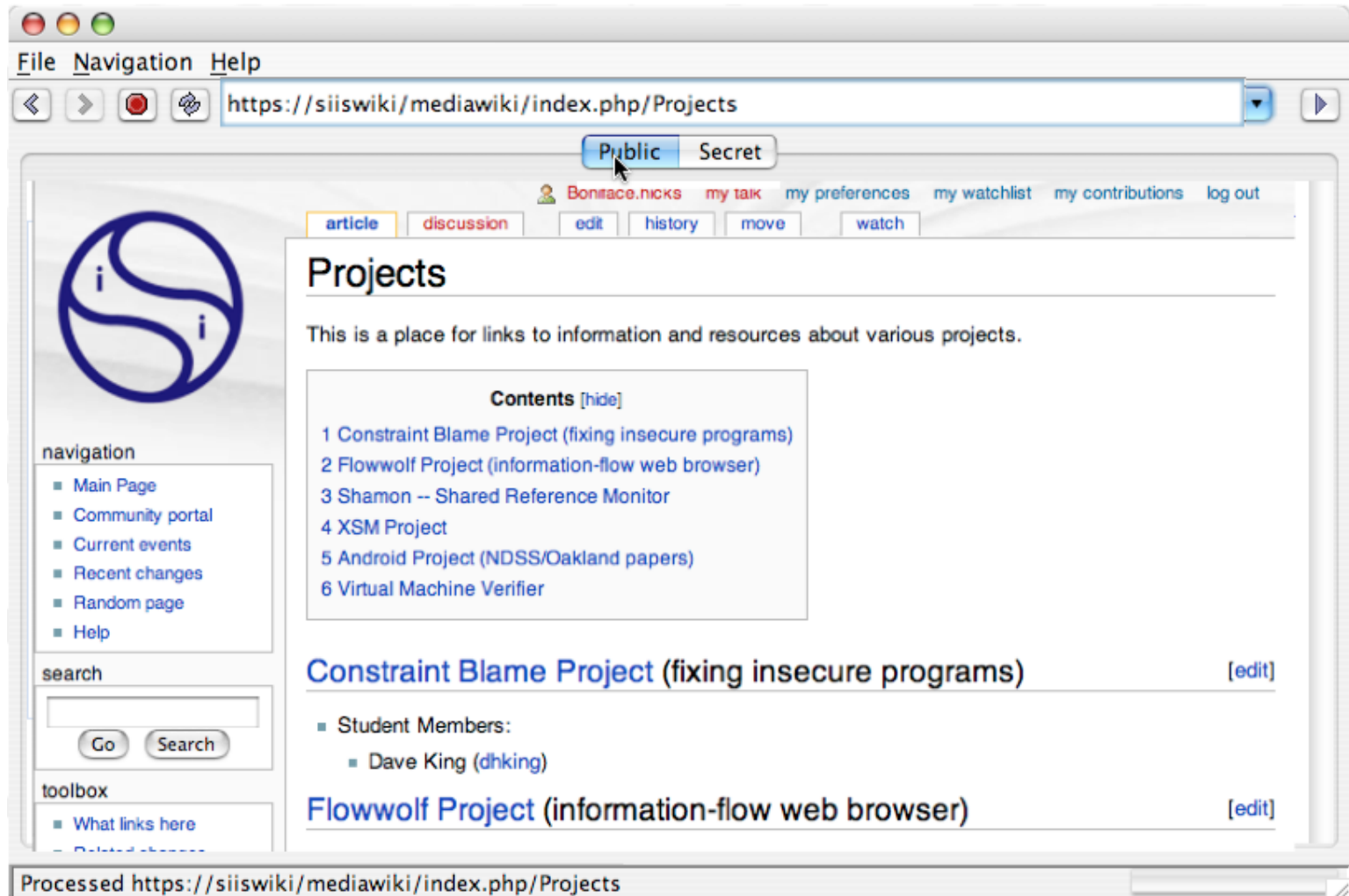
18732 Fall 2010

(with slides from Fr. Boniface Patrick Hicks  
and Michelle Mazurek)

# Protection via separation

- “Air gap” between systems/networks
  - Military networks
- Inter-VM gap
  - No leakage between VMs; VMM reference monitor
- Interprocess gap
  - No leakage between processes; OS as ref. mon.
- Can we do even better?
  - Separation within a process’ data?
  - Control leakage between variables
  - Application as reference monitor

# HTTP mixes data



The screenshot shows a web browser window with the address bar containing the URL `https://siiswiki/mediawiki/index.php/Projects`. The browser's menu bar includes 'File', 'Navigation', and 'Help'. The page content is a MediaWiki page titled 'Projects'. At the top of the page, there are two buttons: 'Public' (highlighted with a mouse cursor) and 'Secret'. Below these are user navigation links for 'Boniface.nicks', 'my talk', 'my preferences', 'my watchlist', 'my contributions', and 'log out'. A secondary navigation bar contains 'article', 'discussion', 'edit', 'history', 'move', and 'watch'. The main heading is 'Projects', followed by the text: 'This is a place for links to information and resources about various projects.' Below this is a 'Contents [hide]' box listing six items: 1 Constraint Blame Project (fixing insecure programs), 2 Flowwolf Project (information-flow web browser), 3 Shamon -- Shared Reference Monitor, 4 XSM Project, 5 Android Project (NDSS/Oakland papers), and 6 Virtual Machine Verifier. The first item, 'Constraint Blame Project (fixing insecure programs)', is expanded to show a list of 'Student Members' including 'Dave King (dhking)'. The second item, 'Flowwolf Project (information-flow web browser)', is also visible. The page includes a sidebar with a logo, a 'navigation' menu, a 'search' box, and a 'toolbox'.

# HTTP mixes data

URL 1

The screenshot shows a web browser window with the address bar containing the URL `https://siiswiki/mediawiki/index.php/Projects`. A blue arrow labeled "URL 1" points to this address bar. The browser's status bar at the bottom indicates "Processed https://siiswiki/mediawiki/index.php/Projects".

The page content includes a navigation menu with links for "article", "discussion", "edit", "history", "move", and "watch". The main heading is "Projects", followed by the text: "This is a place for links to information and resources about various projects." Below this is a "Contents [hide]" section with a numbered list of project links:

- 1 [Constraint Blame Project \(fixing insecure programs\)](#)
- 2 [Flowwolf Project \(information-flow web browser\)](#)
- 3 [Shamon -- Shared Reference Monitor](#)
- 4 [XSM Project](#)
- 5 [Android Project \(NDSS/Oakland papers\)](#)
- 6 [Virtual Machine Verifier](#)

The first item, "Constraint Blame Project (fixing insecure programs)", is expanded to show a sub-section "Student Members:" with a list item "Dave King (dhking)".

Below the expanded section is the heading "Flowwolf Project (information-flow web browser)".

The browser's menu bar includes "File", "Navigation", and "Help". The status bar at the bottom of the browser window shows "Processed https://siiswiki/mediawiki/index.php/Projects".

# HTTP mixes data

URL 1

URL 2

Public Secret

Boniface.nicks my talk my preferences my watchlist my contributions log out

article discussion edit history move watch

## Projects

This is a place for links to information and resources about various projects.

**Contents** [hide]

- 1 [Constraint Blame Project \(fixing insecure programs\)](#)
- 2 [Flowwolf Project \(information-flow web browser\)](#)
- 3 [Shamon -- Shared Reference Monitor](#)
- 4 [XSM Project](#)
- 5 [Android Project \(NDSS/Oakland papers\)](#)
- 6 [Virtual Machine Verifier](#)

[Constraint Blame Project \(fixing insecure programs\)](#) [edit]

- Student Members:
  - [Dave King \(dhking\)](#)

[Flowwolf Project \(information-flow web browser\)](#) [edit]

Processed https://siiswiki/mediawiki/index.php/Projects

# HTTP mixes data

The image shows a screenshot of a web browser window displaying a MediaWiki page titled "Projects". The browser's address bar shows the URL `https://siiswiki/mediawiki/index.php/Projects`. The page content includes a navigation menu, a search box, and a list of project links. Three blue arrows point to specific elements on the page, labeled "URL 1", "URL 2", and "URL 3".

- URL 1** points to the "Public" button in the top navigation bar.
- URL 2** points to the "iS" logo in the left sidebar.
- URL 3** points to the "Main Page" link in the "navigation" menu.

The page content includes the following text and links:

**Public** Secret

Boniface.nicks my talk my preferences my watchlist my contributions log out

article discussion edit history move watch

## Projects

This is a place for links to information and resources about various projects.

**Contents** [hide]

- 1 [Constraint Blame Project \(fixing insecure programs\)](#)
- 2 [Flowwolf Project \(information-flow web browser\)](#)
- 3 [Shamon -- Shared Reference Monitor](#)
- 4 [XSM Project](#)
- 5 [Android Project \(NDSS/Oakland papers\)](#)
- 6 [Virtual Machine Verifier](#)

[Constraint Blame Project \(fixing insecure programs\)](#) [edit]

- Student Members:
  - Dave King (dhking)

[Flowwolf Project \(information-flow web browser\)](#) [edit]

Processed `https://siiswiki/mediawiki/index.php/Projects`

# HTTP mixes data

The screenshot shows a web browser window displaying a MediaWiki page titled "Projects". The browser's address bar shows the URL `https://siiswiki/mediawiki/index.php/Projects`. The page content includes a navigation menu, a search box, and a list of project entries. Four blue arrows labeled "URL 1", "URL 2", "URL 3", and "URL 4" point to specific elements on the page: "URL 1" points to the address bar, "URL 2" points to the site logo, "URL 3" points to the navigation menu, and "URL 4" points to the search box. The page content includes a "Public Secret" toggle, a user profile for "Boniface.nicks", and a list of projects: "Constraint Blame Project (fixing insecure programs)", "Flowwolf Project (information-flow web browser)", "Shamon -- Shared Reference Monitor", "XSM Project", "Android Project (NDSS/Oakland papers)", and "Virtual Machine Verifier".

**URL 1** → `https://siiswiki/mediawiki/index.php/Projects`

**URL 2** → `/index.php/Projects`

**URL 3** → `/index.php/Projects`

**URL 4** → `/index.php/Projects`

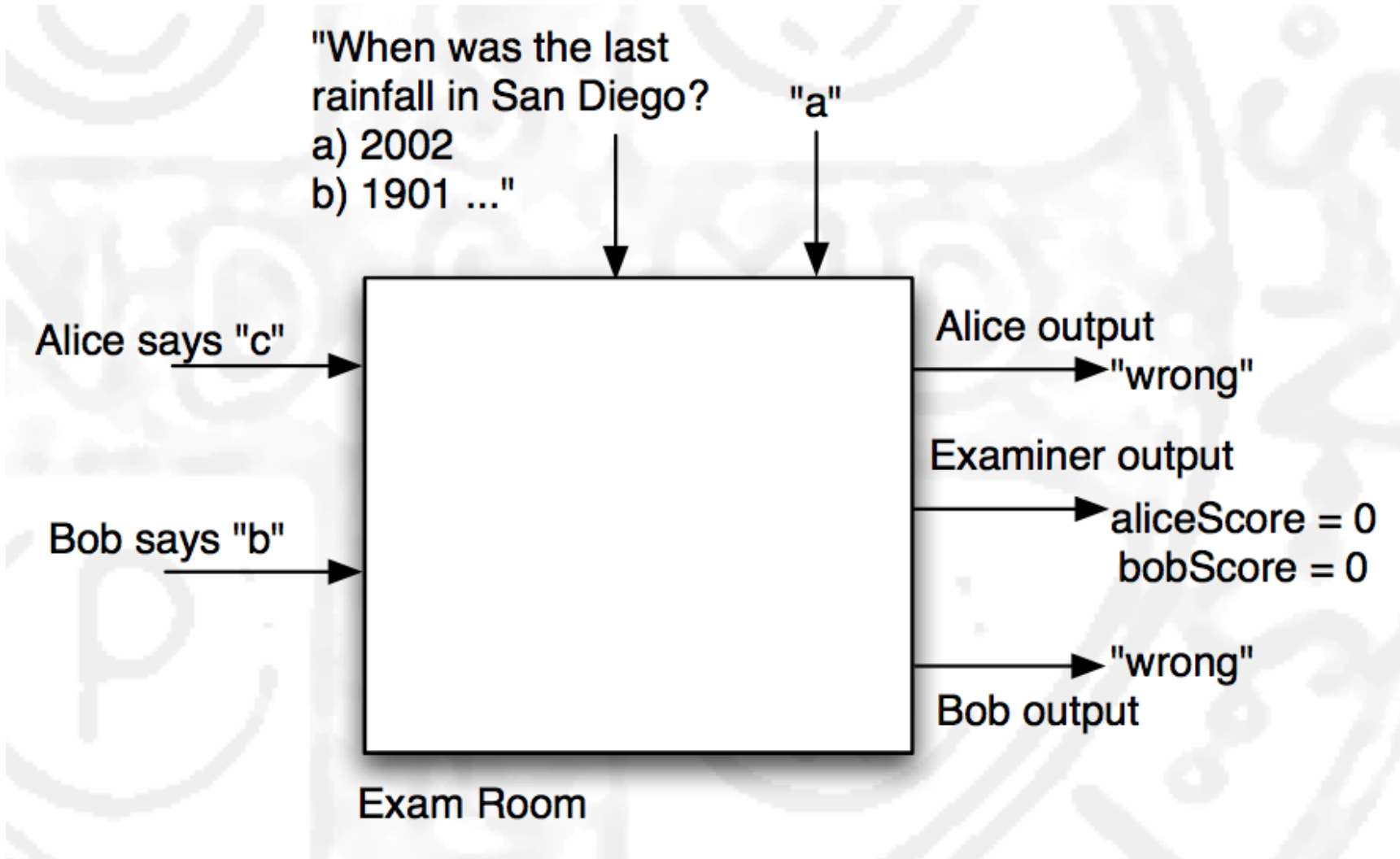
# Application-level RefMons

- Requirements
  - Tamperproof
  - Verifiable
  - Complete mediation
- Inline reference monitor
  - Need to prove complete mediation
  - Dynamic checking required at mediation points
- Type-based approach
  - Complete mediation via strong typing (**compositional**)
  - Much static checking is possible

# Security-typed languages

- Add information flow labels to types
- Add label comparison to type checking
- Examples
  - Jif
  - Fable
  - FlowCaml
  - Aura

# Exam room example



# Exam room example

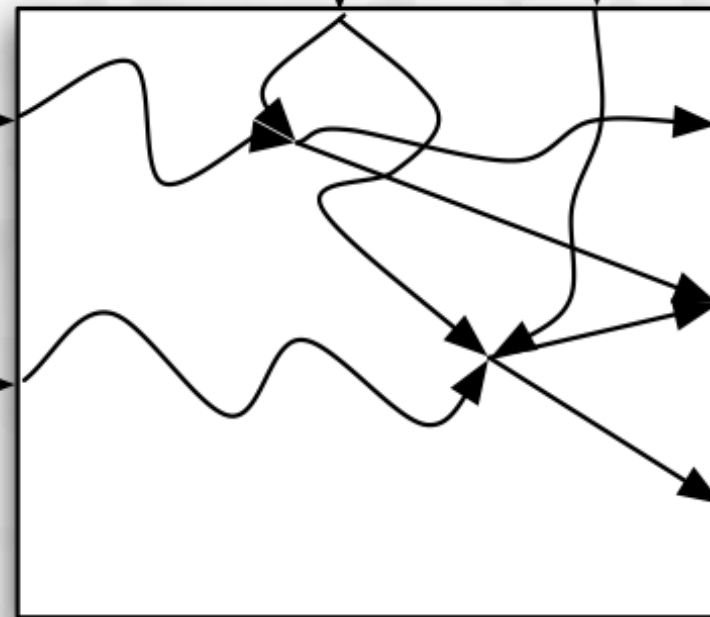
"When was the last  
rainfall in San Diego?"

- a) 2002
- b) 1901 ..."

"a"

Alice says "c"

Bob says "b"



Alice output

"wrong"

Examiner output

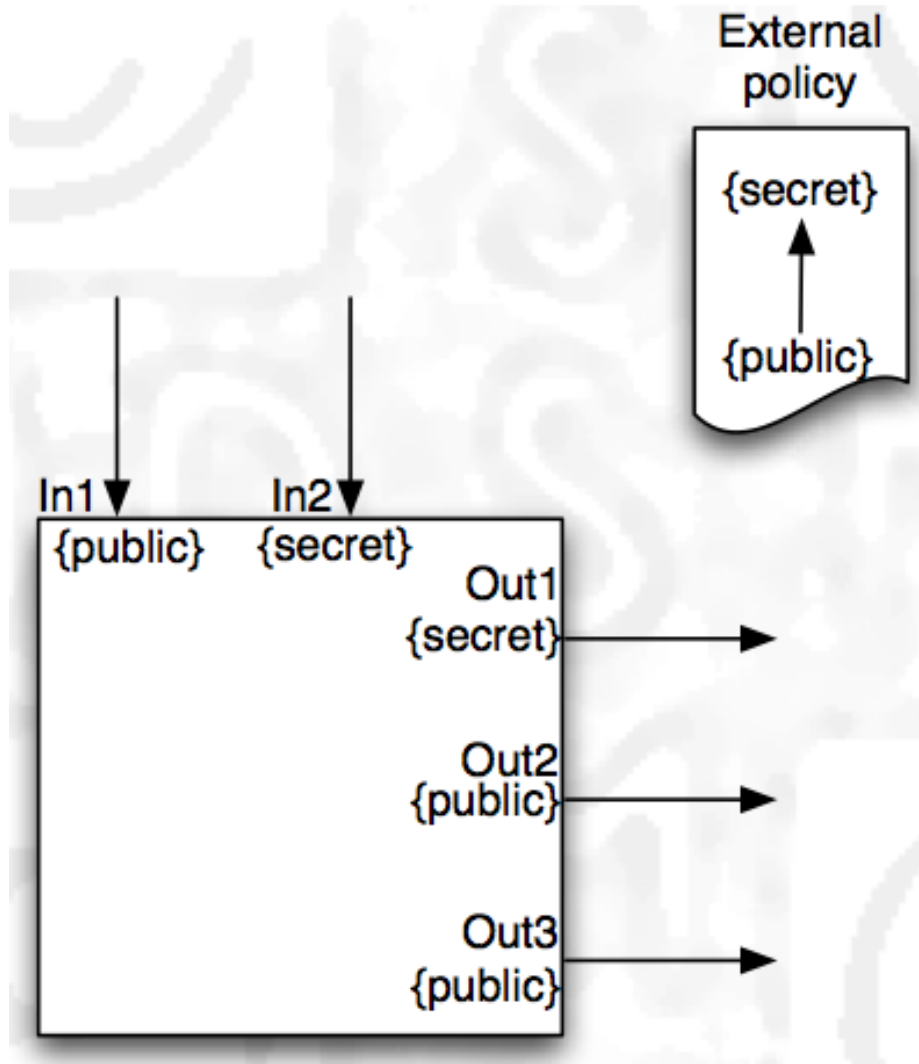
aliceScore = 0  
bobScore = 0

"wrong"

Bob output

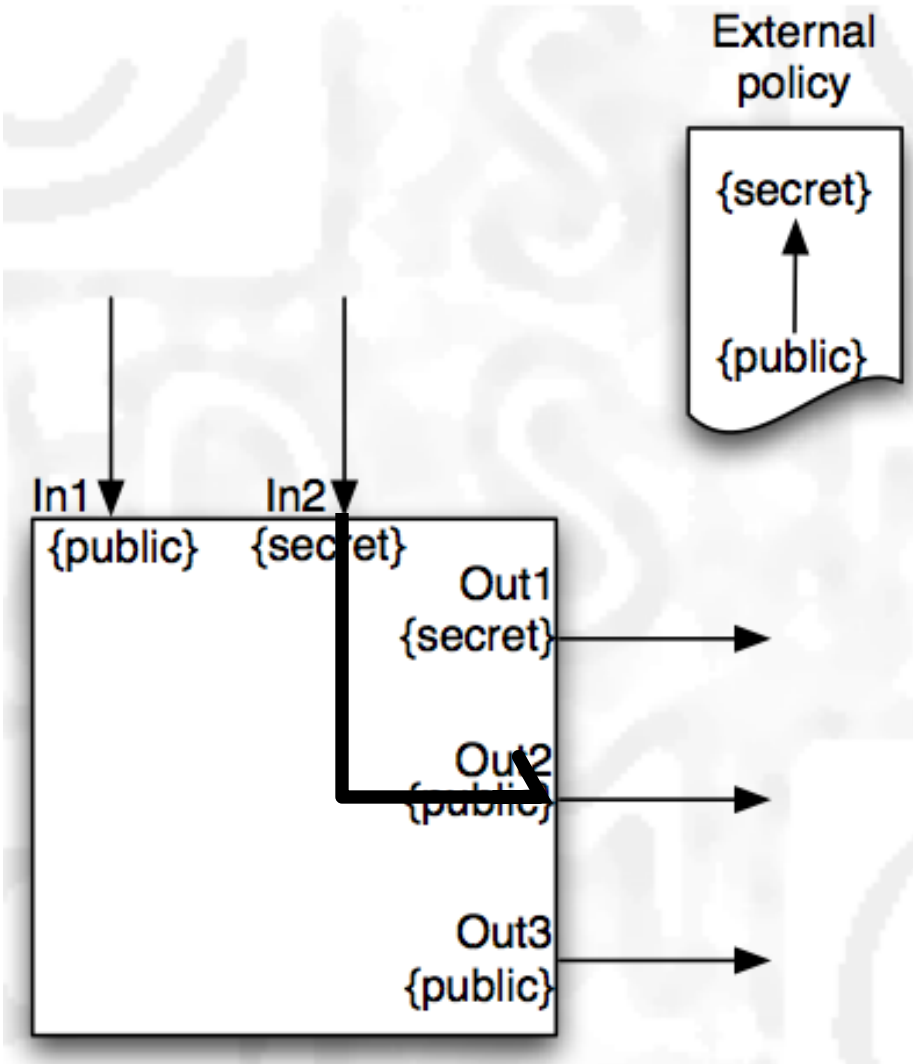
Exam Room

# Add labels to control InfoFlow



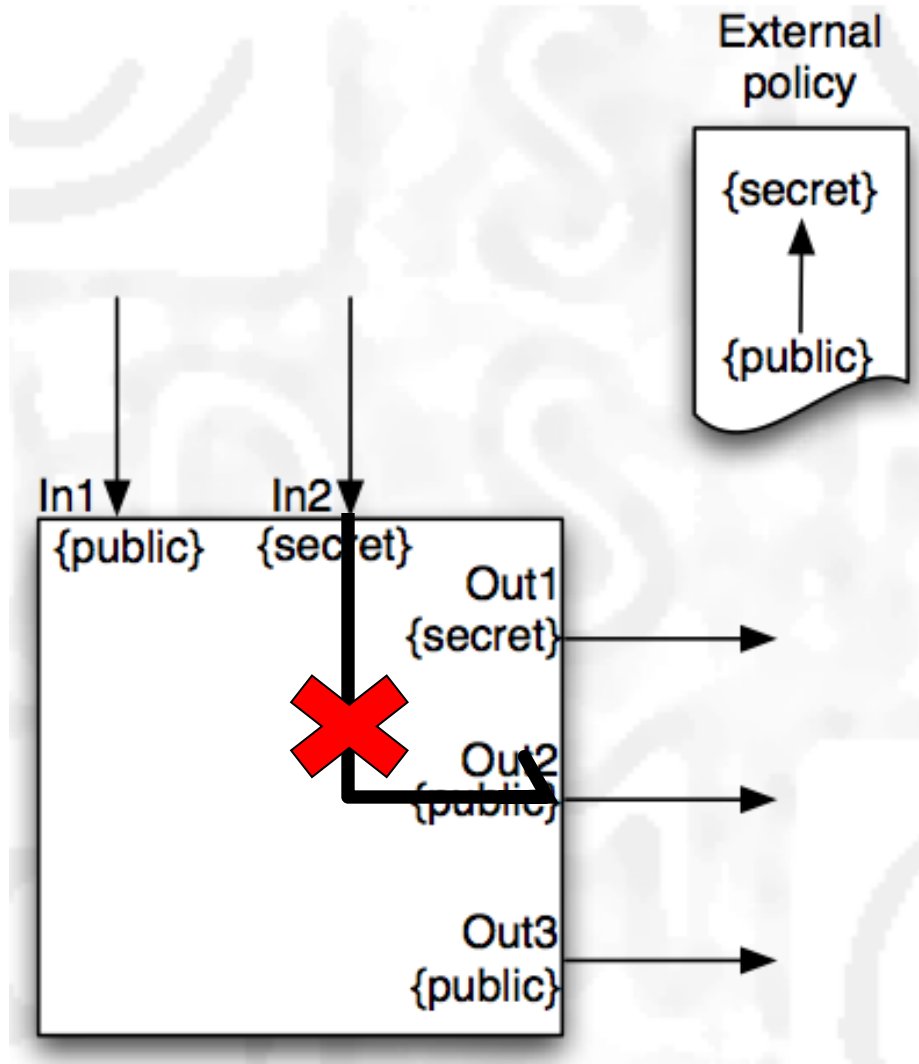
- Add labels to each value
- Label semantics define policy

# Add labels to control InfoFlow



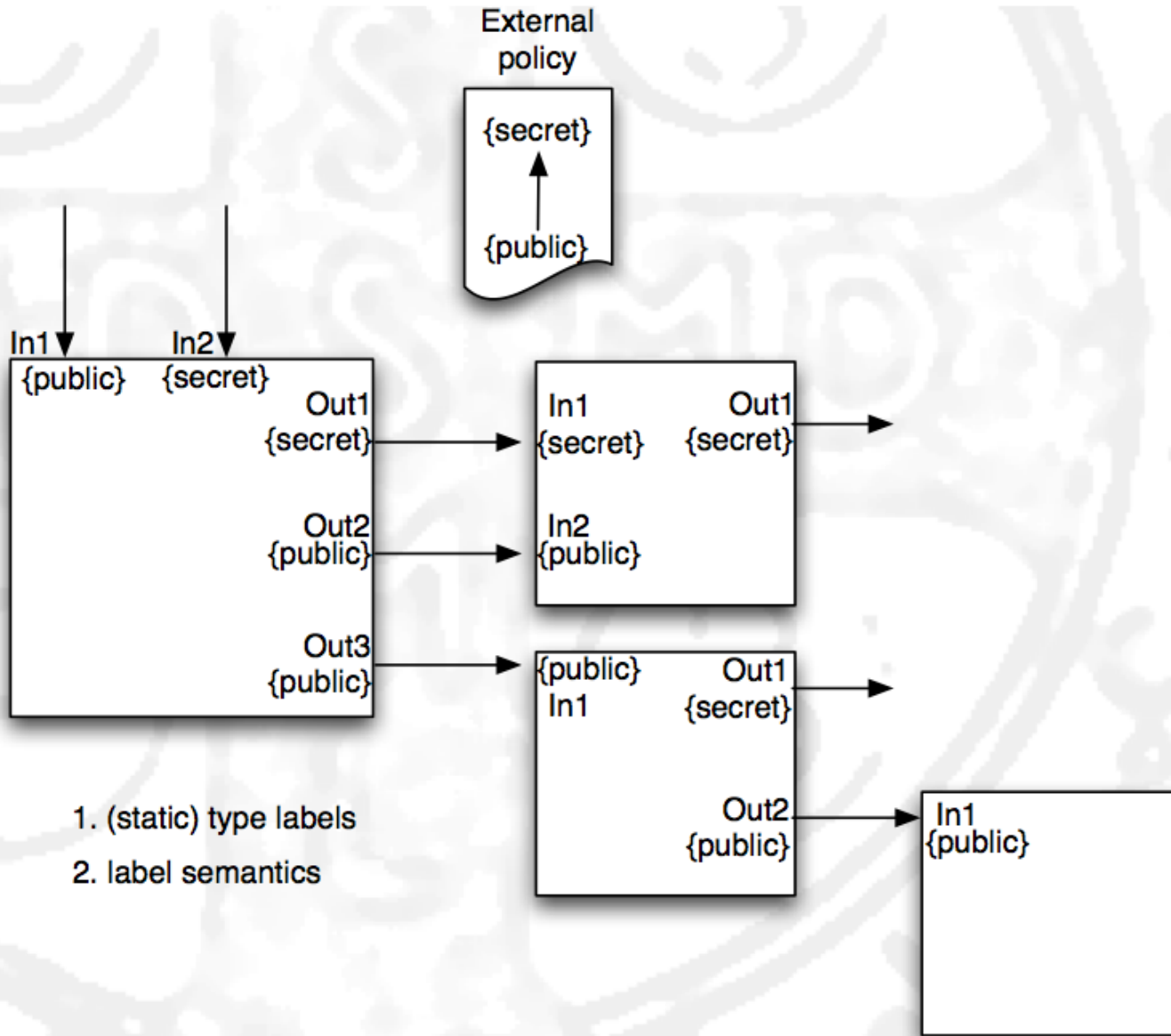
- Add labels to each value
- Label semantics define policy

# Add labels to control InfoFlow

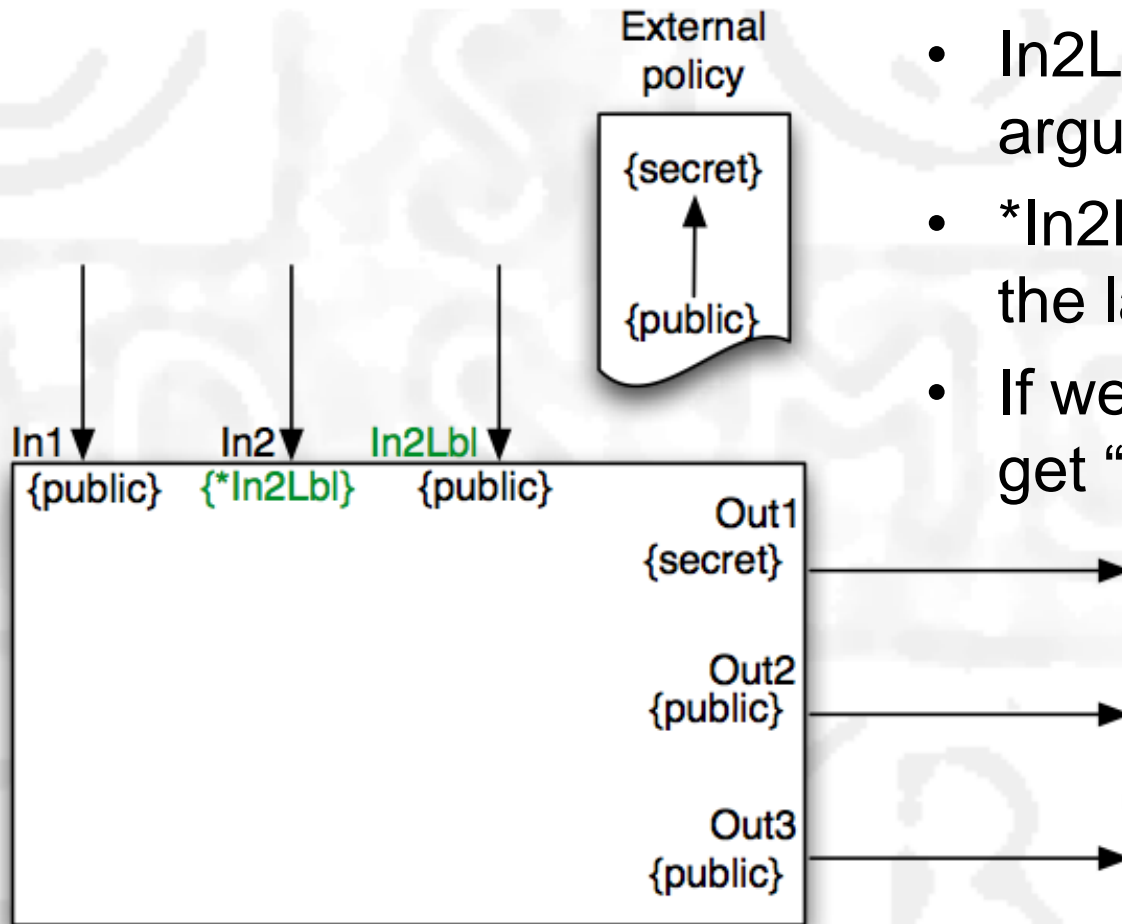


- Add labels to each value
- Label semantics define policy
- Compiler type checking **guarantees** each module contains **no bad information flows**

# Compositionality is easy



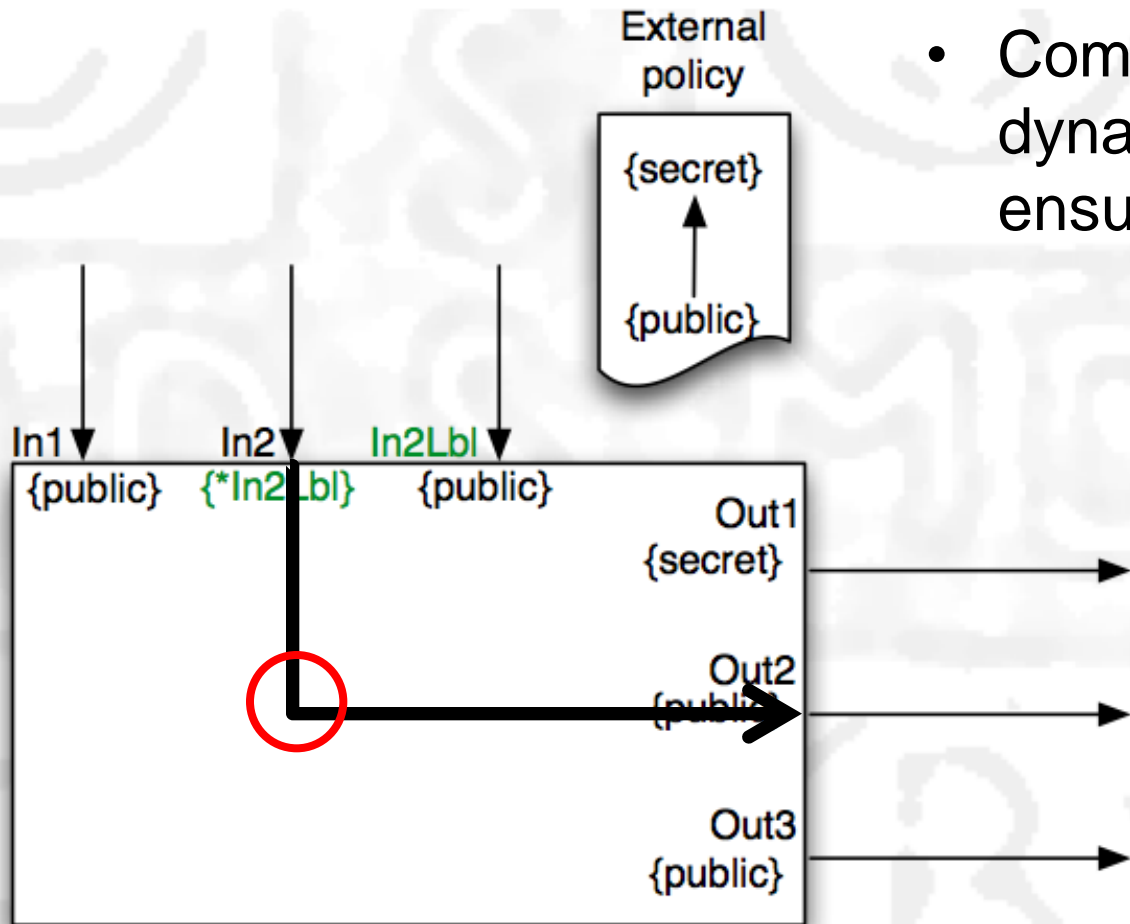
# Dynamic labels



- In2Lbl is a label argument
- \*In2Lbl is the content of the label
- If we used “In2Lbl” we’d get “public”

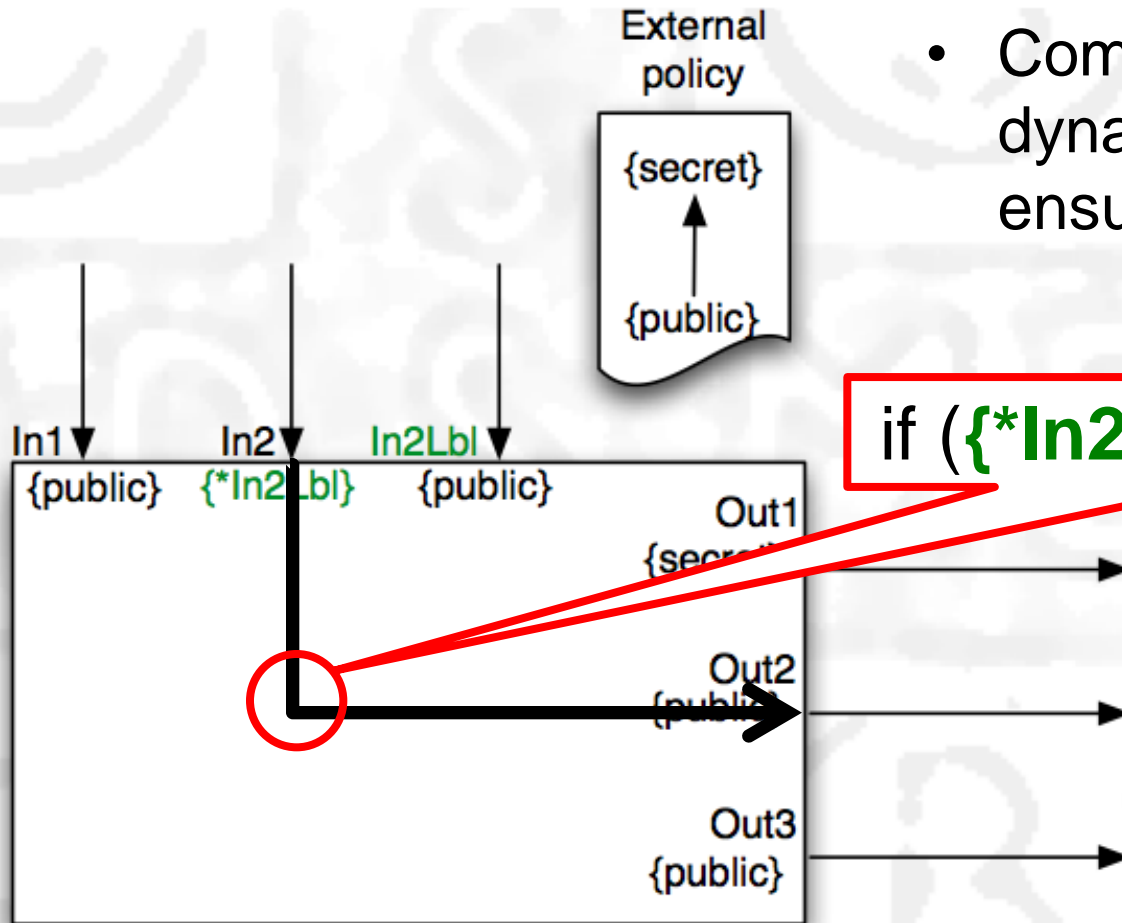
# Dynamic labels

- Compiler requires dynamic check to ensure mediation



# Dynamic labels

- Compiler requires dynamic check to ensure mediation



# Why JIF?

- Most mature security-typed language
- Full-featured variant of Java
  - Jifclipse IDE
- Used to build some real applications
  - Email client (JPmail)
  - Firewall (FlowWall)
  - Servlet framework and servlets (SIF)
  - E-voting (Civitas)
  - Battleship?
- Also, homework 3

# JIF: Decentralized Label Model

- Environment of mutual distrust
  - No central authority
  - Policies from potentially many sources
- Principals can declassify their own data
  - No trusted declassifier
  - No power to weaken others' policies
  - Assume that people you don't trust will not observe your policies voluntarily
  - Requires trusted execution platform (JRE)

# Principals, policies, labels

- Principals are people
  - Policies are defined in terms of principals
- Two kinds of policies: confidentiality, integrity
  - Confidentiality: Who **can read** my data?
  - Integrity: Who I believe **could have influenced** the data?
- Labels: Syntax for expressing policies

# A little formalism: Principals

- Acts for:  $q$  acts for  $p$  ( $q \geq p$ )
  - $p$  delegates authority to  $q$
  - Any action taken by  $q$  is implicitly authorized by  $p$
- Groups: all members **act for** the group
- $\top$  = top (\*) ... acts for everyone
- $\perp$  = bottom ( $\_$ ) ... everyone acts for it
- $p \& q$  ...  $p \& q \geq p, p \& q \geq q$
- $p, q$  ...  $p \geq p, q; q \geq p, q$

# A little formalism: Confidentiality

- $o \rightarrow r$  =  $o$ 's policy is to allow  $q$  to read if  $q \geq r$

```
int{Bob→Alice} x;
```

```
int{Bob→Alice} z;
```

```
int y = x;
```

```
z = y;
```

```
int{Bob:Alice} x;  
int{Bob:Alice,Bob} x;
```

```
Type of y inferred as {Bob→Alice}
```

- $o$ 's policy is only relevant if principal executing the code is  $q$ , where  $q \geq o$

# A little formalism: Confidentiality

- $o \rightarrow r$  =  $o$ 's policy is to allow  $q$  to read if  $q \geq r$

```
int{Bob→Bob} x;
```

```
int{Bob→Alice} z;
```

```
if (x == 1)
```

```
    z = 1;
```

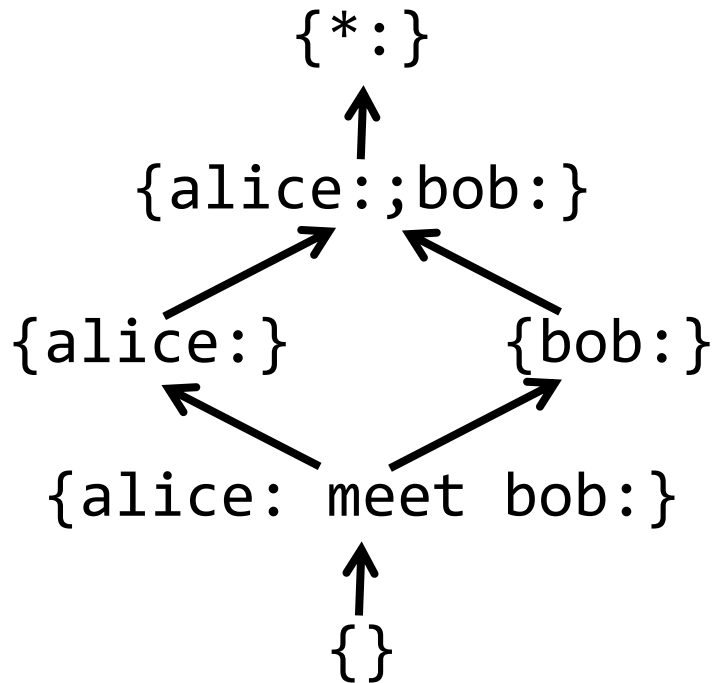


```
pc : {Bob→Bob}
{Bob→Alice} <≠ {Bob→Bob}
```

- Jif maintains a program counter  $pc$  and infers labels for it
- Operation isn't allowed if  $pc$  doesn't have sufficient privilege

# A little formalism: Labels

- Labels form a lattice




# A little formalism: Integrity

- $o \leftarrow w$  =  $o$ 's policy is to allow  $q$  to write if  $q \geq w$

```
int{Bob←Bob} x;
```

```
int{Bob←Alice} z;
```

```
...
```

```
 x = z;
```

# Secret messages example

- Consider whether any functions leak information from `aliceInstructions` to `bobInstructions` or vice versa
- Note: In Jif, a method's header should fully characterize all flows in the method to facilitate compositionality

# Secret messages example

```
public class SecretMessages[label alice, label bob] {
    String{*alice} aliceInstructions;
    String{*bob} bobInstructions;

    public SecretMessages(String{*alice} ai, String{*bob} bi) {
        aliceInstructions = ai;
        bobInstructions = bi;
    }

    public String{*alice} getAliceMsg(){
        return aliceInstructions;
    }

    public String{*bob} getBobMsg(){
        return bobInstructions;
    }
}
```

# Explicit flow prevention

```
public class SecretMessages[principal alice, principal bob] {  
    String{alice:} aliceInstructions;  
    String{bob:} bobInstructions;
```

```
    public SecretMessages(String{alice:} ai, String{bob:} bi) {  
        aliceInstructions = ai;  
        bobInstructions = bi;  
    }
```

...

```
    public String{bob:} leak(){  
        bobInstructions = aliceInstructions;  
        return bobInstructions;  
    }  
}
```



**Bob cannot read Alice's data!**

# Implicit flow prevention

```
public class SecretMessages[principal alice, principal bob] {  
    String{alice:} aliceInstructions;  
    String{bob:} bobInstructions;
```

```
    public SecretMessages(String{alice:} ai, String{bob:} bi) {  
        aliceInstructions = ai;  
        bobInstructions = bi;  
    }
```


...


```
    public String{bob:} implicitLeak() {  
        try {  
            if (aliceInstructions.equals("Attack at dawn"))  
                bobInstructions = "Attack at dawn";  
        } catch (NullPointerException e) {}  
        return bobInstructions;  
    }  
}
```



**Bob cannot read Alice's data!**

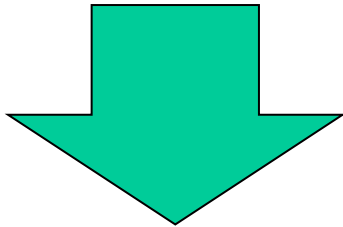
# Implicit flow prevention (2)

```
public String{bob:} implicitLeak2() {
    try {
        aliceInstructions.trim();
    } catch (NullPointerException e) {
 bobInstructions = null;
    }
    return bobInstructions;
}

public String{bob:} implicitLeak3() {
    try {
        aliceInstructions.trim();
 bobInstructions = null;
    } catch (NullPointerException e) {}
    return bobInstructions;
}
```


# Implicit flow prevention (3)

```
public String{bob:} implicitLeak3() {  
    try {  
        aliceInstructions.trim();  
        bobInstructions = null;  
    } catch (NullPointerException e) {}  
    return bobInstructions;  
}
```



```
public String{bob:} implicitLeak3() {  
    try {  
        bobInstructions = null;  
        aliceInstructions.trim();  
    } catch (NullPointerException e) {}  
    return bobInstructions;  
}
```

# Side-effect leakage

```
public class SecretMessages[label alice, label bob] {  
    String{*alice} aliceInstructions;  
    String{*bob} bobInstructions;  
  
    public String{*alice} getAliceMsg() {  
        return aliceInstructions;  
    }  
  
     public void setAliceMsg() {  
        aliceInstructions = "";  
    }  
}
```

- Some information is stored at level {\*alice}
- Must be careful where this is called


# Side-effect leakage

```
public void sideEffectLeak() {  
    try {  
        if (bobInstructions.equals("Attack at Dawn"))  
            setAliceMsg();  
    } catch (NullPointerException e) {}  
}
```

- To preserve compositionality, Jif requires method headers to cover all information flows

# Side-effect leakage

```
public void sideEffectLeak() {  
    try {  
        if (bobInstructions.equals("Attack at Dawn"))  
            setAliceMsg();  
    } catch (NullPointerException e) {}  
}
```



```
public class SecretMessages[label alice, label bob] {  
    String{*alice} aliceInstructions;  
    String{*bob} bobInstructions;  
  
    public String{*alice} getAliceMsg() {  
        return aliceInstructions;  
    }  
  
    public void setAliceMsg{*alice}() {  
        aliceInstructions = "";  
    }  
}
```

# Dynamic labels

```
import java.io.PrintStream;

public class SpyMessage {
    final public label{} lbl;
    String{*lbl} secret;

    public SpyMessage(principal{} user) {
        lbl = new label{user:};
        this.secret = "Attack at dawn";
    }

    public String{*lbl} getSecretOutput() {
        return secret;
    }
}
```

- Labels have labels!
- lbl refers to the label on the label
- \*lbl refers to the label itself

# Dynamic label checking

```
public class Main {  
    public static void main{}(principal{} user, String[] args) {  
        final SpyMessage{user:} myMessage = new SpyMessage(user);  
  
        PrintStream[{user:}] outS = null;  
        jif.runtime.Runtime[user] rt = null;  
        try {  
            rt = jif.runtime.Runtime[user].getRuntime();  
            outS = rt != null ? rt.stdout(new label{user:}) : null;  
        }  
        catch (java.lang.SecurityException e) {}  
        if (myMessage.lbl <= new label{user:}) ← Label required  
            by outS  
            if (outS == null)  
                outS.println(myMessage.getSecretOutput());  
        DEBUG.println("That's all.");  
    }  
}
```

**Label on myMessage**

# Authority declarations

- How does code acquire authority?

```
class Game authority(referee) {  
    void start() where authority(referee) {  
        ...    // authority of referee  
    }  
    void halftimeShow() {  
        ...    // no authority  
    }  
}
```

# Jif features

- Principals
  - User-defined
  - Principal hierarchy
- Label polymorphism and parameterized classes
- Automatic label inference
- Run-time label checking
- Downgrading
  - *Declassification* for confidentiality
  - *Endorsement* for integrity

# Take-away

- Practical non-interference is hard!
- Many features necessary to implement non-interference usefully

# Resources

- Jif homepage: <http://www.cs.cornell.edu/jif/>
- Jifclipse: <http://siis.cse.psu.edu/jifclipse/>