CS 5430
Information-Flow Control

Elisavet Kozyri

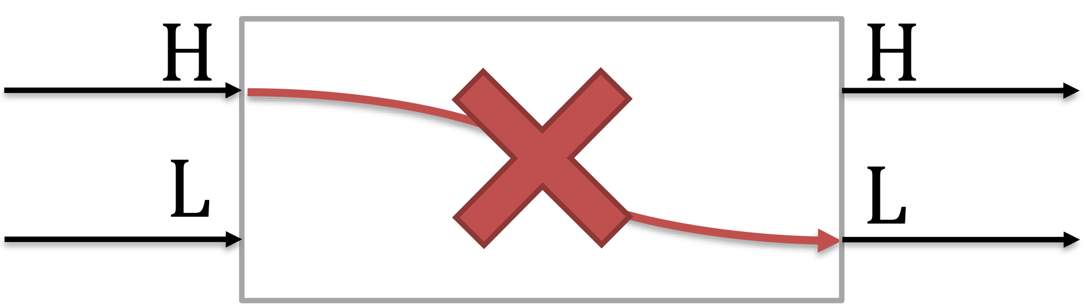
Spring 2019

Information Flow Control

- We are given:
  - a lattice $\langle L, \sqsubseteq \rangle$ of labels,
  - a program, and
  - an *environment* Γ that maps variables to labels.
- Threat model: Attacker knows L inputs, knows program code, and can observe L outputs.

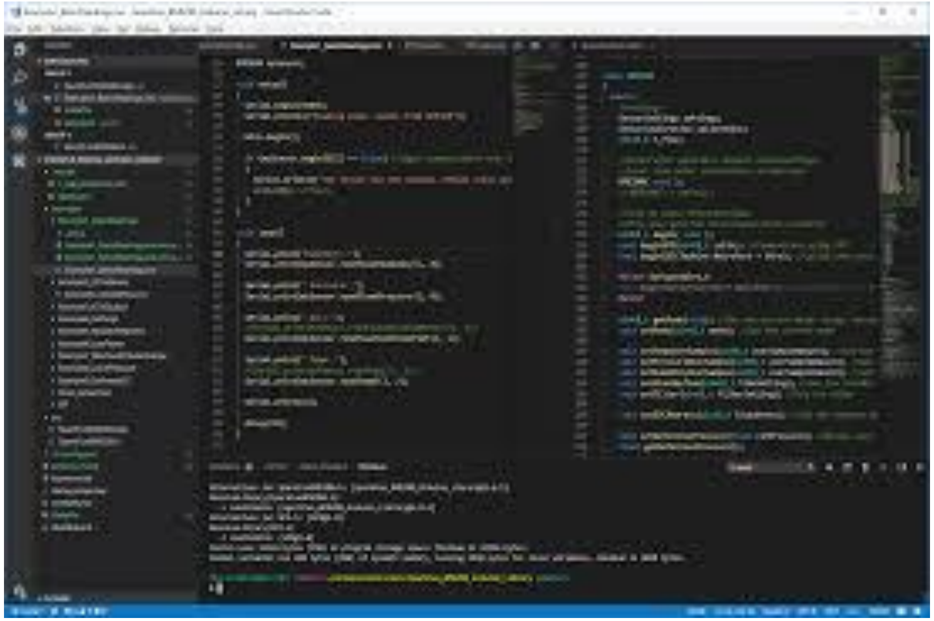
Information Flow Control

- Executions of the program should cause only allowed flows (with respect to the lattice).
- The program is expected to satisfy noninterference (NI):
  - Different H inputs, keeping L inputs fixed, should not cause different L outputs.



Information Flow Control

- How can we check that a program satisfies NI?
- Design an *enforcement mechanism:*
  - takes as an input the program, and
  - outputs whether that program satisfies NI.



Enforcing NI

- Static mechanism
  - Checking program before execution.
- Dynamic mechanism
  - Checking program during execution.
- Hybrid mechanism
  - Combination of static and dynamic.

Static enforcement mechanism

```
x := 2;
y := x+z;
if y>0 then
  z:= 10;
  x:= x-1;
else
  z:= w
```

Static enforcement mechanism

```
x := 2;
y := x+z;
if y>0 then
  z:= 10;
  x:= x-1;
else
  z:= w
```

Static enforcement mechanism

```
x := 2;
y := x+z;
if y>0 then
  z:= 10;
  x:= x-1;
else
  z:= w
```

Static enforcement mechanism

```
x := 2;
y := x+z;
```

```
if y>0 then
  z:= 10;
  x:= x-1;
else
  z:= w
```

Static enforcement mechanism

```
x := 2;
y := x+z;
if y>0 then
  z:= 10;
  x:= x-1;
else
  z:= w
```

Programs are written using this syntax:

```
e ::= x | n | e1+e2 | ...

c ::= x := e
    | if e then c1 else c2
    | while e do c
    | c1; c2
```

Checking an assignment

Assignments cause **explicit** flows of values.

```
x := y
```

It satisfies NI, if $\Gamma(\mathtt{y}) \sqsubseteq \Gamma(\mathtt{x})$.

Checking an assignment

If $\Gamma(\mathtt{y}) \sqsubseteq \Gamma(\mathtt{x})$, then it satisfies NI.

```
x := y
```

Examples for confidentiality

$\Gamma(\mathtt{x})$ is L.
$\Gamma(\mathtt{y})$ is L.
Does this assignment satisfy NI? ☺

$\Gamma(\mathtt{x})$ is H.
$\Gamma(\mathtt{y})$ is L.
Does this assignment satisfy NI? ☺

Checking an assignment

```
x := y + z
```

It satisfies NI, if $\Gamma(\mathtt{y+z}) \sqsubseteq \Gamma(\mathtt{x})$.

It satisfies NI, if $\Gamma(\mathtt{y}) \sqcup \Gamma(\mathtt{z}) \sqsubseteq \Gamma(\mathtt{x})$.

Checking an if-statement

```
if y>0 then x:=1 else x:=2
```

Conditional commands (e.g., if-statements and while-statements) cause **implicit** flows of values.

Context

```
if y>0 then x:=1 else x:=2
```

They reveal information about `y>0`.

Context label ctx

```
if y>0 then x:=1 else x:=2
```

Their ctx is $\Gamma(\mathtt{y})$.

Context label ctx

```
if y>0 then x:=1 else x:=2
```

Check if $ctx \sqsubseteq \Gamma(\mathbf{x})$,
where $ctx = \Gamma(\mathbf{y})$.

Context label ctx

```
if y>0 then x:=e else x:=2
```

Check if
$ctx \sqcup \Gamma(\mathbf{e}) \sqsubseteq \Gamma(\mathbf{x})$.

Implicit flow

Explicit flow

Checking an if-statement

If $\Gamma(\mathbf{y}) \sqcup \Gamma(\mathbf{z}) \sqsubseteq \Gamma(\mathbf{x})$, then it satisfies NI.

```
if y>0 then x:=z+1 else x:=z+2
```

Examples for confidentiality

$\Gamma(\mathbf{x})$ is L.
$\Gamma(\mathbf{y})$, $\Gamma(\mathbf{z})$ are L. ☺
Does this if-statement satisfy NI?

$\Gamma(\mathbf{x})$, $\Gamma(\mathbf{z})$ are H.
$\Gamma(\mathbf{y})$ is L. ☺
Does this if-statement satisfy NI?

Checking an if-statement

```
if z>0 then
    if y>0 then x:=1 else x:=2
else
    x:=3
```

Their ctx is $\Gamma(\mathbf{z}) \sqcup \Gamma(\mathbf{y})$.

Its ctx is $\Gamma(\mathbf{z})$.

Checking an if-statement

```
if z>0 then
    if y>0 then x:=1 else x:=2
else
    x:=3
```

Check if $ctx \sqsubseteq \Gamma(\mathbf{x})$,
where $ctx = \Gamma(\mathbf{z}) \sqcup \Gamma(\mathbf{y})$.

Check if $ctx \sqsubseteq \Gamma(\mathbf{x})$,
where $ctx = \Gamma(\mathbf{z})$.

A type system for information flow control

- Static
- Fixed environment Γ
- Labels as types
  - Label $\Gamma(\mathbf{x})$ is the type of $\mathbf{x}$.
- Typing rules for all possible commands.
- **Goal**: type-correctness $\Rightarrow$ noninterference

We are already familiar with type systems!

Example of typing rule from Java:

`x + y : int`

assuming **`x : int`** and **`y : int`**

Typing rules for expressions

Judgement $\Gamma \vdash \texttt{e} : \ell$

- According to environment Γ, expression $\texttt{e}$ has type (i.e., label) ℓ.

Typing rules for expressions

Expression: $\Gamma \vdash \texttt{e+e}' : \ell \sqcup \ell'$

assuming $\Gamma \vdash \texttt{e} : \ell$ and $\Gamma \vdash \texttt{e}' : \ell'$

Inference rule:

Premises ⟶ $\Gamma \vdash \texttt{e} : \ell \qquad \Gamma \vdash \texttt{e}' : \ell'$

Conclusion ⟶ $\Gamma \vdash \texttt{e+e}' : \ell \sqcup \ell'$

$$\frac{\Gamma \vdash \texttt{e} : \ell \qquad \Gamma \vdash \texttt{e}' : \ell'}{\Gamma \vdash \texttt{e+e}' : \ell \sqcup \ell'}$$

Typing rules for expressions

Constant:

$$\frac{}{\Gamma \vdash \mathbf{n} : \bot}$$

Bottom: the least restrictive label.

Variable:

$$\frac{\Gamma(\mathbf{x}) = \ell}{\Gamma \vdash \mathbf{x} : \ell}$$

Example

- Let $\Gamma(\mathbf{x}) = \mathrm{L}$ and $\Gamma(\mathbf{y}) = \mathrm{H}$.
- What is the type of **`x+y+1`**?
- *Proof tree:*

$\Gamma(\mathbf{x}) = \mathrm{L}$ $\qquad\qquad$ $\Gamma(\mathbf{y}) = \mathrm{H}$

Example

- Let $\Gamma(\mathbf{x}) = \mathrm{L}$ and $\Gamma(\mathbf{y}) = \mathrm{H}$.
- What is the type of **`x+y+1`**?
- *Proof tree:*

$$\frac{\Gamma(\mathbf{x}) = \ell}{\Gamma \vdash \mathbf{x} : \ell}$$

$$\frac{\Gamma(\mathbf{x}) = \mathrm{L}}{\Gamma \vdash \mathbf{x} : \mathrm{L}} \qquad \frac{\Gamma(\mathbf{y}) = \mathrm{H}}{\Gamma \vdash \mathbf{y} : \mathrm{H}}$$

Example

- Let $\Gamma(\mathbf{x}) = \mathrm{L}$ and $\Gamma(\mathbf{y}) = \mathrm{H}$.
- What is the type of **`x+y+1`**?
- *Proof tree:*

$$\frac{}{\Gamma \vdash \mathbf{n} : \bot}$$

$$\frac{\Gamma(\mathbf{x}) = \mathrm{L}}{\Gamma \vdash \mathbf{x} : \mathrm{L}} \qquad \frac{\Gamma(\mathbf{y}) = \mathrm{H}}{\Gamma \vdash \mathbf{y} : \mathrm{H}} \qquad \Gamma \vdash \mathbf{1} : \mathrm{L}$$

Example

- Let $\Gamma(\mathbf{x}) = \mathrm{L}$ and $\Gamma(\mathbf{y}) = \mathrm{H}$.
- What is the type of $\mathbf{x+y+1}$?
- *Proof tree:*

$$\frac{\Gamma \vdash \mathbf{e} : \ell \qquad \Gamma \vdash \mathbf{e}' : \ell'}{\Gamma \vdash \mathbf{e+e}' : \ell \sqcup \ell'}$$

$$\dfrac{\dfrac{\Gamma(\mathbf{x}) = \mathrm{L}}{\Gamma \vdash \mathbf{x} : \mathrm{L}} \qquad \dfrac{\Gamma(\mathbf{y}) = \mathrm{H}}{\Gamma \vdash \mathbf{y} : \mathrm{H}} \qquad \Gamma \vdash \mathbf{1} : \mathrm{L}}{\Gamma \vdash \mathbf{x + y + 1} : \mathrm{H}}$$

Typing rules for commands

Judgement $\Gamma, ctx \vdash$ **`c`**

- According to environment Γ, and context label ctx, command **`c`** is type correct.

Assignment rule

$$\frac{\Gamma \vdash \mathtt{e} : \ell \qquad\qquad \ell \sqcup ctx \sqsubseteq \Gamma\mathtt{(x)}}{\Gamma, ctx \vdash \mathtt{x:=e}}$$

If-rule

$$\frac{\Gamma \vdash \mathtt{e} : \ell \qquad \Gamma, \ell \sqcup ctx \vdash \mathtt{c1} \qquad \Gamma, \ell \sqcup ctx \vdash \mathtt{c2}}{\Gamma, ctx \vdash \mathtt{if\ e\ then\ c1\ else\ c2}}$$

If-rule (example)

$$\dfrac{\Gamma \vdash \mathbf{y>0} : \Gamma(\mathbf{y}) \qquad \dfrac{\Gamma(\mathbf{y}) \sqsubseteq \Gamma(\mathbf{x})}{\Gamma, \Gamma(\mathbf{y}) \sqcup L \vdash \mathbf{x:=1}} \qquad \dfrac{\Gamma(\mathbf{y}) \sqsubseteq \Gamma(\mathbf{x})}{\Gamma, \Gamma(\mathbf{y}) \sqcup L \vdash \mathbf{x:=2}}}{\Gamma, L \vdash \mathbf{if\ y>0\ then\ x:=1\ else\ x:=2}}$$

What is the relation between $\Gamma(\mathbf{x})$ and $\Gamma(\mathbf{y})$, such that the above judgement can be proved?

If-rule (example)

$$\dfrac{\Gamma \vdash \mathbf{y>0} : \Gamma(\mathbf{y}) \qquad \dfrac{\Gamma(\mathbf{y}) \sqsubseteq \Gamma(\mathbf{x})}{\Gamma, \Gamma(\mathbf{y}) \sqcup \mathrm{L} \vdash \mathbf{x:=1}} \qquad \dfrac{\Gamma(\mathbf{y}) \sqsubseteq \Gamma(\mathbf{x})}{\Gamma, \Gamma(\mathbf{y}) \sqcup \mathrm{L} \vdash \mathbf{x:=2}}}{\Gamma, \mathrm{L} \vdash \mathbf{if\ y>0\ then\ x:=1\ else\ x:=2}}$$

What is the relation between $\Gamma(\mathbf{x})$ and $\Gamma(\mathbf{y})$, such that the above judgement can be proved?

If-rule (example)

$$\frac{\Gamma(\mathbf{y}) \sqsubseteq \Gamma(\mathbf{x})}{\Gamma, \mathrm{L} \vdash \texttt{if y>0 then x:=1 else x:=2}}$$

What is the relation between $\Gamma(\mathbf{x})$ and $\Gamma(\mathbf{y})$, such that the above judgement can be proved?

If-rule (example)

$$\frac{\Gamma \vdash \mathbf{z>0} : \Gamma(\mathbf{z}) \qquad \Gamma, \Gamma(\mathbf{z}) \sqcup \mathrm{L} \vdash \mathbf{if\ y>0\ then\ x:=1\ else\ x:=2} \qquad \Gamma, \Gamma(\mathbf{z}) \sqcup \mathrm{L} \vdash \mathbf{x:=3}}{\Gamma, \mathrm{L} \vdash \mathbf{if\ z>0\ then\ \{if\ y>0\ then\ x:=1\ else\ x:=2\}\ else\ \{x:=3\}}}$$

while-rule

$$\frac{\Gamma \vdash \mathbf{\texttt{e}} : \ell \qquad \Gamma, \ell \sqcup ctx \vdash \mathbf{\texttt{c}}}{\Gamma, ctx \vdash \mathbf{\texttt{while e do c}}}$$

Sequence rule

$$\frac{\Gamma, ctx \vdash \texttt{c1} \qquad \Gamma, ctx \vdash \texttt{c2}}{\Gamma, ctx \vdash \texttt{c1;c2}}$$

Sequence rule (example)

$$\dfrac{\dfrac{\Gamma, \ell \sqcup \Gamma(\mathtt{e}) \vdash \mathtt{x:=1} \quad \Gamma, \ell \sqcup \Gamma(\mathtt{e}) \vdash \mathtt{x:=2}}{\Gamma, \ell \vdash \mathtt{if\ e\ then\ \{x:=1\}\ else\ \{x:=2\}}} \qquad \Gamma, \ell \vdash \mathtt{x:=3}}{\Gamma, \ell \vdash \mathtt{if\ e\ then\ \{x:=1\}\ else\ \{x:=2\};\ x:=3}}$$

Theorem

Type correctness $\Rightarrow$ Noninterference

- If a program does not satisfy NI, then type checking fails.
  - Example, where Γ `(x)=`L and Γ `(y)=`H:

$$\frac{\Gamma \vdash \mathbf{y} : \mathrm{H} \qquad \mathrm{H} \sqcup \mathrm{L} \sqsubseteq \Gamma(\mathbf{x})}{\Gamma, \mathrm{L} \vdash \mathbf{x{:=}y}}$$

Fails

Does not satisfy NI

- But, if type checking fails, then the program might or might nor satisfy NI.

$$\frac{\Gamma \vdash \mathbf{y*0} : \mathrm{H} \qquad \mathrm{H} \sqcup \mathrm{L} \sqsubseteq \Gamma(\mathbf{x})}{\Gamma, \mathrm{L} \vdash \mathbf{x{:=}y*0}}$$

Fails

Satisfies NI

Soundness and completeness

The type system is *sound*:

$$\text{Type correctness} \Rightarrow \text{Noninterference}$$

but not *complete*:

$$\text{Noninterference} \not\Rightarrow \text{Type correctness}$$

This type system has false positives.

Can we build a complete mechanism?

- Is there an enforcement mechanism for information flow control that has no false positives?
  - A mechanism that rejects only programs that do not satisfy noninterference?
- No! [Sabelfeld and Myers, 2003]
  - "The general problem of confidentiality for programs is undecidable."
  - The halting problem can be reduced to the information flow control problem.
  - Example:

```
if h>0 then c; l:=2 else skip
```

    - If we could precisely decide whether this program is secure, we could decide whether **c** terminates!

But we can build a mechanism with fewer false positives.

NEXT LECTURES

This type system does not prevent leaks through termination channels.

Example of termination channel:

```
while h != 0 do skip;
l:=2
```

where **h** is a high variable and **l** is a low variable.

- The program leaks over termination channel.
  - It does not satisfy *termination sensitive* noninterference.
- But, the program is type correct.
  - It satisfies (vanilla) noninterference.

A solution

- To prevent covert channels due to infinite loops,
- strengthen the typing rule for while-statement, to allow only low guard expression:

$$\frac{\Gamma \vdash \mathtt{e} : \bot \qquad \Gamma, ctx \vdash \mathtt{c}}{\Gamma, ctx \vdash \mathtt{while\ e\ do\ c}}$$

- Now, type correctness implies termination sensitive NI.
- But, the enforcement mechanism becomes overly conservative.
- Another solution?