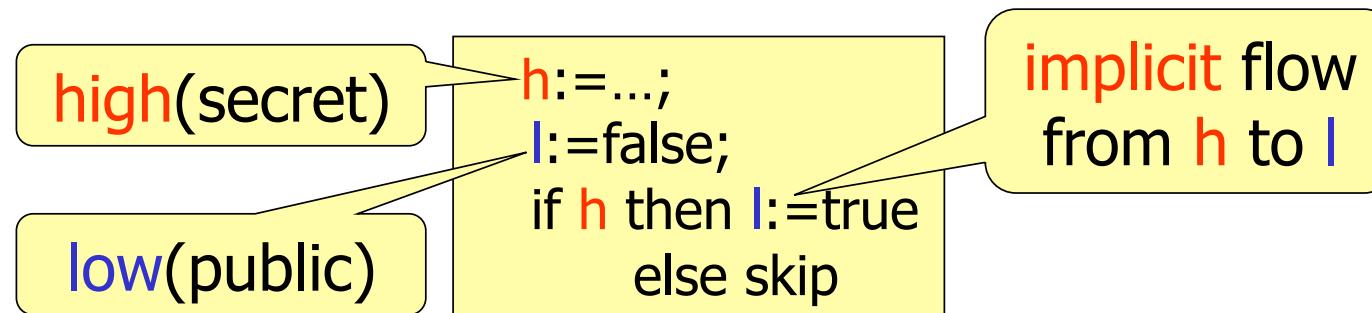


Static types for Information Flow

Continuiamo il discorso sul meccanismo di analisi del flusso dell'informazione interno di programmi e come vi ricordate la volta scorsa non abbiamo fatto vedere che cosa significa avere un modello dell'attaccante che ha controllo delle variabili di basso livello, abbiamo definito il modello del comportamento che è basato su una nozione semantica della non interferenza, abbiamo visto che ci sono diverse nozioni di non interferenza, quella che è non sensitiva alla terminazione, quella sensitiva alla terminazione e abbiamo detto che ci sono diversi modi di affrontare l'aspetto dell'enforcement, ovvero dell'andare a controllare le caratteristiche, adesso quello che noi andiamo a vedere andiamo a vedere un meccanismo statico .

Dynamic control of Information flow



Problem: have to monitor `all` execution paths!

Perché partiamo da un meccanismo statico che poi dopo, come vedrete a fine della lezione, arriveremo a una soluzione che mescola sia la parte statica che la parte dinamica perché se andiamo a considerare un meccanismo totalmente dinamico, dobbiamo controllare tutti i cammini di esecuzione, questo semplice esempio che qui trovate descritto, cioè se abbiamo questo programma che ha una variabile di alto livello quindi `high` che è `secret` e una di basso livello che è pubblica e vedete quando andiamo a fare questa semplice sequenza di istruzioni che contiene `if then else` subito ci scontriamo con dei flussi impliciti di informazioni che non sono flussi dovuti agli assegnamenti delle variabili o altre operazioni che dipendono dalle operazioni che hanno a che fare con il controllo delle guardie. Allora se noi dovessimo esaminare tutti possibili comportamenti, dovremmo in un meccanismo dinamico, controllare tutti i possibili flussi di esecuzione e tutti i possibili cammini e quindi questo lo renderebbe molto più complicato.

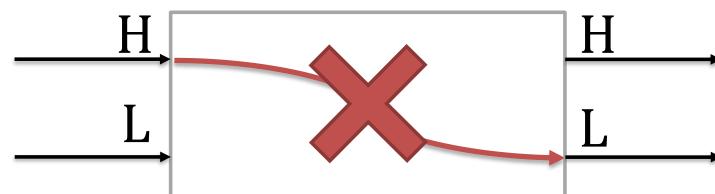
allora quello che noi vogliamo fare in generale quello che è lo studio della letteratura e delle tecniche in questo momento, nel caso del linguaggio di programmazione, ha dei meccanismi di certificazione statica che cosa sono i meccanismi? L'idea è che soltanto i programmi che verificano una delle proprietarie di sicurezza possono essere mandati in esecuzione e questa verifica della proprietà di sicurezza viene fatta staticamente dal compilatore in fase di analisi semantica. Quindi poi sono l'idea in generale, adesso in questo caso noi vediamo information flow, ma più in generale l'idea è che nel back end dei compilatori vengono affrontate tante cose, tra cui la certificazione di proprietà di sicurezza e che soltanto i programmi che superano, che hanno quindi un bollino di certificazione di alcune proprietà di sicurezza, vengono mandati in esecuzione, per cui a quel punto la certificazione statica è un meccanismo che va inserito nella toolchain del linguaggio e quindi l'evoluzione anche nel back end dei compilatori è che non solo si fanno delle trasformazioni di codice che rendono il codice maggiormente ottimizzato, ma si fanno tutte quelle analisi e noi abbiamo visto già la taint analysis, si fanno tutte quelle analisi che assicurano che certe proprietà valgono quando il programma viene mandato in esecuzione. Allora quello che noi adesso andremo a vedere, andremo a vedere questo meccanismo di certificazione statica attraverso il sistema dei tipi, quindi attraverso un sistema che codifica all'interno dei tipi le proprietà di sicurezza in modo tale da garantire questa proprietà di information flow.

Static certification

- Only run programs which can be statically **verified** as secure **before** running them
- **Static certification** for inclusion in the language tool-chain
- Enforcement by **security-type systems**

Security levels as a lattice

- We are given
 - a lattice L , \sqsubseteq of security levels (labels),
 - a program
 - an environment Γ that maps variables to labels.
- Attacker knows L inputs and can observe L outputs.
- Static Program certification: non interference



allora ricordiamo il punto di partenza, il quindi di partenza è noi abbiamo un reticolo di security level e ad ogni security level viene associata una etichetta di sicurezza che caratterizza la proprietà di quel livello di sicurezza. Abbiamo il programma e poi abbiamo un informazione di ambiente che in fase di analisi statica corrisponde all'informazione che uno riesce ad arrivare ad inserire nella tabella dei simboli, quindi subito all'inizio del backend, dei compilatori cosa contiene questo ambiente? L'ambiente contiene i mapping tra i nomi delle variabile e il livello di sicurezza associato alle variabili, quindi sono le etichette che caratterizzano il livello di sicurezza di quella variabile all'interno del programma. Cosa può fare l'attaccante? Può vedere gli input pubblici, quindi quelli di livello basso, può osservare gli output pubblici, ovvero le informazioni disponibile una volta che il programma le buttafuori e sempre di livello pubblico e quello che si vuole evitare è che ci sia un flusso interferito cioè come descrivere quella grossa x in quel flusso rosso in quel disegno, nella nostra programma, che ci sia un flusso di informazioni, dove informazioni di alto livello, ovvero privata, venga trasmessa su un canale pubblico. Allora in questo caso, la certificazione statica è che questo programma soddisfa la proprietà di non interferenza.

A simple imperative language

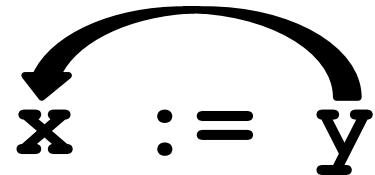
e ::= **x** | **n** | **e1+e2** | ...

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

Partiamo per introdurre i concetti e poi man mano estenderemo quindi la strategia è quella di avere un nucleo di base dove introduciamo i concetti e poi far vedere come questo nucleo di base cresce fino ad affrontare tutti gli aspetti dei linguaggi, esattamente la stessa strategia che abbiamo adottato nel caso della taint analysis. Allora, supponiamo di avere un linguaggio imperativo, un linguaggio imperativo banale da un punto di vista dell' espressività perché ho un insieme di espressioni, l'espressione possono essere variabili, costante, supponiamo in questo momento costante intere ma potrebbe metterci pure booleani insomma, un certo numero di insieme di tipi base in più un certo numero di operazioni. Poi abbiamo i comandi che sono l' assegnamento, l'ifthenelse, il while quindi è un nucleo di linguaggio funzionale, senza nulla di particolarmente diverso da quello che uno si aspetta

Certifying assignments

Assignments cause **explicit** flows of values.



The assignment satisfies Non Interference if $\Gamma(y) \sqsubseteq \Gamma(x)$.

Allora adesso andiamo a vedere come certifichiamo andiamo a vedere intuitivamente qual'è l'operazione di certificazione che dobbiamo fare per i costrutti del programma, in modo tale da garantire che valga la proprietà di non interference. Allora partiamo dalla cosa più semplice, come abbiamo fatto per la taint analysis sull'assegnamento x prende y . Noi sappiamo benissimo che questo definisce un flusso tra il valore trasportato della y nella variabile x , infatti da un punto di vista dell'assegnamento cosa vuol dire? Si valuta il valore di y , cioè si valuta la y , se la y è una variabile, si va a vedere le locazioni di memoria associata al nome y quindi concettualmente questo vuol dire si determina l'offset dell'indirizzo di base dove verrà poi memorizzata la γ , si va a vedere in memoria dove è e si trasferisce questo valore nella locazione di memoria associata alla x , quindi chiaramente c'è un flusso di informazioni di valori dal valore della y al valore della x . Quand'è che questo assegnamento soddisfa la proprietà di non interferenza? Soddisfa la proprietà di non interferenza quando il livello di sicurezza della y è compatibile con livello di sicurezza della x , se leggiamo dal basso. Leggendolo dall'alto intendo dire del lattice le vi ricordate che l'operazione di minore uguale sul lattice nel nostro laccio di lattice di sicurezza ci dice che se io ho un qualcosa che è più grande, quindi in questo caso gamma di y che è più grande di gamma di x quello che vogliamo avere è che le proprietà di sicurezza associate alla gamma di x sono quanto meno restrittive quanto quelle della y , quindi questo cosa vuol dire? Vuol dire che questo assegnamento non permette di mandare fuori dell'informazione proprio per il fatto che c'è questa compatibilità tra i due livelli di sicurezza e quindi questo è la cosa che dobbiamo tenere conto quando vogliamo fare la certificazione dei vincoli di sicurezza.

Examples

Certification Constraint:

If $\Gamma(y) \sqsubseteq \Gamma(x)$, then $x := y$ satisfies NI.

If $\Gamma(y) = L$, $\Gamma(x) = L$, then $x := y$ satisfies NI?

If $\Gamma(y) = L$, $\Gamma(x) = H$, then $x := y$ satisfies NI?

Examples

Certification Constraint:

If $\Gamma(y) \sqsubseteq \Gamma(x)$, then $x := y$ satisfies NI.

If $\Gamma(y) = L$, $\Gamma(x) = L$, then $x := y$ satisfies NI? 

If $\Gamma(y) = L$, $\Gamma(x) = H$, then $x := y$ satisfies NI? 

Allora andiamo a vedere alcuni esempi, quello che qui viene chiamato il certification constraint, il vincolo di certificazione ci dice che x prende y soddisfa la non interference quando siamo nella condizione che il livello di sicurezza della y è compatibile col livello di sicurezza della x. Se gamma di y è low e gamma di x è uguale a low allora x uguale y soddisfa la non interferenza? sicuramente sì, perché sono allo stesso livello. Andiamo a vedere invece la situazione in cui x è uguale high e y uguale a low, sicuramente anche questo soddisfa la proprietà di non interreference perchè il livello di sicurezza associata alla x, contiene tutte le caratteristiche ed è quindi restrittivo quanto quello della y e infatti mettiamo una faccina sorridente nel nostro certificato che ci dice che in questa caso qui entrambi soddisfano.

What about

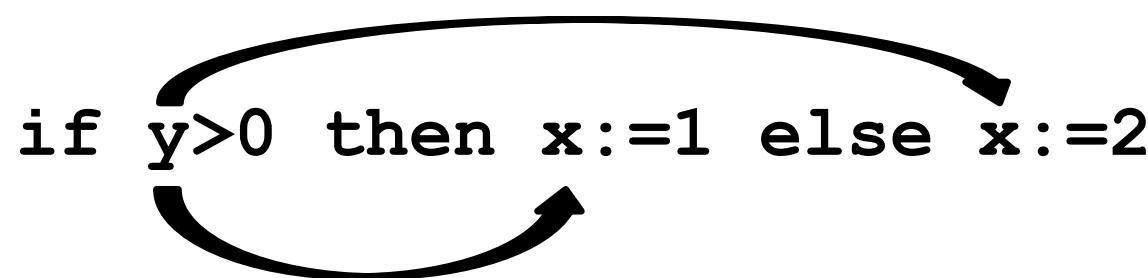
$$X := Y + Z$$

The assignment satisfies NI, if $\Gamma(y+z) \sqsubseteq \Gamma(x)$.

The assignment satisfies NI, if $\Gamma(y) \sqcup \Gamma(z) \sqsubseteq \Gamma(x)$.



Conditionals



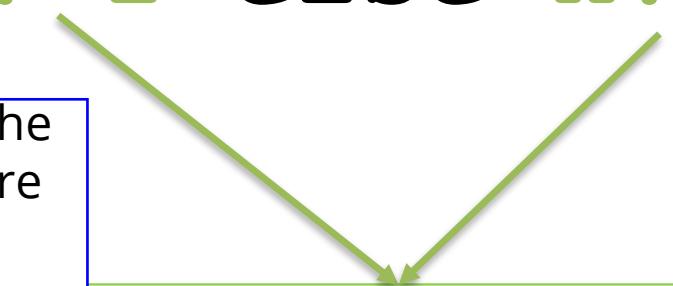
Conditional commands (if-statements and while-statements) determines **implicit flows** of values.

C'è un flusso tra il ramo `then` e il ramo `else` a seconda della condizione della guardia questo però è il flusso diretto, poi abbiamo anche un flusso implicito di informazione, ed è che noi riusciamo ad andare a vedere il valore del segno della `y` semplicemente andando a vedere il risultato alla fine, quindi vuol dire che c'è un flusso implicito di informazione che dobbiamo tenere conto. Andiamo ad esaminarla nel dettaglio.



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

In un'espressione di ifthenelse come quella che abbiamo visto, abbiamo che andando a vedere poi senza conoscere il valore della y immaginiamo che il valore della y sia di alto livello, per cui l'attaccante non può vedere il valore della y, però andando a vedere il valore della x se assumiamo che la y è di basso livello, l'attaccante può andare a comprendere non il valore del variabile di alto livello y sicuramente no, ma sicuramente può andare a vedere se è maggiore o minore uguale di zero, andando a discriminare i valori di x uguale a 1 o uguale a 2.



They reveal information about y>0.



Typing conditionals

Cosa vuol dire in termini di certificazione di sicurezza? Che io quando vado a certificare un costrutto condizionale, non mi basta il livello di sicurezza della x , cioè non posso agire localmente e devo tener conto del contesto dove questa operazione è eseguita. Il contesto lo dovete immaginare come l'ambiente operazionale che corrisponde a quello che l'attaccante può avere d'informazione, quindi il contesto rappresenta l'ambiente operazionale in cui l'attaccante opera. Vuol dire che quando noi andiamo a certificare un `ifthenelse` dobbiamo metterci un contesto, il contesto dell'attaccante e in modo particolare il contesto dell'attaccante in questo caso deve tener conto del fatto che l'esecuzione del ramo `then` o del ramo `else` è determinata sostanzialmente dal contesto della y , quindi l'attaccante è in grado di interferire utilizzando questa operazione di contesto, dunque, in modo particolare di dedurre delle informazioni. In modo particolare quindi se noi dobbiamo nel nostro esempio, certificare che questa condizionale sciocco soddisfa la proprietà di non interferenza, dobbiamo definire una nozione di contesto in modo particolare il contesto è il valore di sicurezza associata a gamma di y e dobbiamo controllare che il contesto sia compatibile con le informazioni che sono presenti a livello della x , quindi vuol dire che il livello di sicurezza della x per poter prendere il flusso di informazione dovuto a quell'assegnamento deve essere tale da avere le stesse proprietà di sicurezza del contesto in cui opera e quindi vuol dire che non devo guardare la x in isolamento per il condizionale, ma devo tener conto del contesto.

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

#1 The security level of x is not sufficient to type IF-branches

The security context (cxt) of both branches is that of $\Gamma(y)$

#2 In or example: Check if $cxt \sqsubseteq \Gamma(x)$

where $cxt = \Gamma(y)$.

Typing conditionals (cont)

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



**Check if $\text{ctx} \sqcup \Gamma(e) \sqsubseteq \Gamma(x)$
where $\text{ctx} = \Gamma(y)$.**

Se noi andiamo a vedere quali sono le informazioni di contesto che ci servono per definire la certificazione statica del condizionale, nel caso particolare di questa operazione, in modo particolare è una generalizzazione di quello che abbiamo visto prima, x prende e , quello che noi dobbiamo andare a verificare come proprietà di certificazione che il contesto in modo particolare in questo esempio particolare, il valore di sicurezza, le etichette di sicurezza associate alla y join il livello di sicurezza dell'espressione che è esattamente quelle informazioni che dicevamo prima, deve essere compatibile con livello di sicurezza della x . Allora se noi riusciamo a verificare che questa proprietà contestuale vale vuol dire che questo condizionale soddisfa la non interferenza

Typing conditionals (cont)

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



Check if $ctx \sqcup \Gamma(e) \sqsubseteq \Gamma(x)$

where $ctx = \Gamma(y)$.



IMPLICIT FLOW

EXPLICIT FLOW

Questo check vuol dire che il livello di sicurezza della x deve essere informativo allo stesso modo e quindi deve avere tutti i vincoli chiave che ha sia il livello implicito del flusso quello che è rappresentato dal contesto, ma anche dal livello esplicito del flusso caratterizzato dal tipo di sicurezza dell'espressione. Questo è esattamente l'informazione o il vincolo di certificazione che dobbiamo tener conto, cioè dobbiamo tener conto del flusso implicito, il contesto e del flusso esplicito dovuto alla valutazione dell'espressione

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

CHECK: $\Gamma(y) \sqcup \Gamma(z) \sqsubseteq \Gamma(x)$

$\Gamma(x) = L, \Gamma(y) = L, \Gamma(z) = L$

The IF stat. satisfies NI?

$\Gamma(x) = H, \Gamma(y) = L, \Gamma(z) = H$



The IF stat. satisfies NI?

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

CHECK: $\Gamma(y) \sqcup \Gamma(z) \sqsubseteq \Gamma(x)$

$\Gamma(x) = L, \Gamma(y) = L, \Gamma(z) = L$

The IF stat. satisfies NI? 

$\Gamma(x) = H, \Gamma(y) = L, \Gamma(z) = H$

The IF stat. satisfies NI? 

```

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

```

x:=3



Its *ctx* is $\Gamma(z)$.



Their *ctx*
is $\Gamma(z) \cup \Gamma(y)$.

Andiamo a vedere la cosa come scala a costrutti più complicati. Siamo partiti dal assegnamento, siamo passati al condizionale, poi siamo passati a condizionale uno dentro l'altro e qui stiamo facendo esattamente la stessa cosa allora andiamo a vedere questo queste condizionale. Qual è il contesto? Il contesto di riferimento è dato da quello che avevo prima, cioè l'informazione relativa al livello di sicurezza della zeta join le informazioni di sicurezza della y che è la seconda condizione. Quindi quando vado a valutare l'assegnamento x più uno mi devo tener conto dell'informazione che ho passato e che quindi implicitamente ho due valori da considerare, due valori da considerare che non considero singolarmente, ma dato che uso un reticolo nel modello semantico dei livelli di sicurezza, vuol dire che il contesto deve avere il vincolo più grande e il vincolo più grande mi è dato dal join del livello di sicurezza della zeta e di quello della y e quindi a questo punto quando vado a fare il controllo usando le stessa idea del controllo del contesto che abbiamo visto prima per l'assegnamento devo considerare il join tra z e y, anzi tra il livello di sicurezza della zeta e della y. La cosa invece è diversa nel ramo else dell'if esterno perché quello è un semplice assegnamento, quindi il livello di sicurezza di contesto è dato soltanto dal livello di sicurezza della variabile della guardia zeta, infatti il contesto è gamma di zeta.

```

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

```

The diagram shows a flow from the assignment $x := 3$ down to the `else` keyword, which then branches to the `if y>0` condition of the inner if-statement.

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

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

Allora a questo punto facendo la stessa analisi che abbiamo fatto nel caso precedente, quand'è che l'assegnamento del ramo `then` certifica la proprietà di non interferenza, soddisfa la proprietà di sicurezza? Quando il contesto è minore uguale del livello di sicurezza della x , ovvero quando il contesto del ramo `else` che è il sup tra il livello di sicurezza di x e y , è compatibile con il livello di sicurezza della x , nel secondo caso invece devo semplicemente andare a vedere il livello di sicurezza della $zeta$ se è compatibile con livello di sicurezza della x . Quindi vedete che l'idea scala a costrutti più articolati, quindi si riesce a fare questa cosa in modo consistente.

Type system for Information flow

- Static
- Fixed environment Γ
- Security levels (labels) as types
 - Security level $\Gamma(x)$ is the type of x .
- **Goal: type-correctness \Rightarrow noninterference**

Dopo aver capito qual'è l'idea intuitiva, andiamo a vedere nel dettaglio come progettare il sistema di tipo. Allora deve essere statico la prima ipotesi che noi facciamo è che facciamo un assegnamento statico delle variabili a i livelli di sicurezza, quindi questo cosa vuol dire? Vuol dire che staticamente noi sappiamo quali sono le variabili di alto livello, le variabili di basso livello e questo assegnamento non può essere modificato. Vuol dire che, ad esempio, non possiamo dedurre delle nuove informazioni sulle variabili in cui precedentemente avevamo associato a un livello di sicurezza. Come vedrete questo, poi è un vincolo molto stringente che rilasceremo più avanti, però inizialmente questo è il vincolo che abbiamo, cioè abbiamo fissato l'ambiente che definisce il livello di sicurezza e l'abbiamo fissato a priori. A questo punto avendolo fissato a priori vale il solito discorso che con gamma di x intendiamo il tipo di x e il nostro obiettivo è se il sistema associa un tipo a un costrutto del linguaggio, allora, viene garantita la non inferenza ed ecco la certificazione. Quindi questo cosa vuol dire? vuol dire che i programmi ben tipati sono programmi che rispettano la proprietà di non interferenza.

Typing expressions

- Judgement $\Gamma \vdash e : I$
- Intuitive meaning: According to environment Γ , expression e has type (i.e., label) I .

Allora partiamo da l'espressione, quindi abbiamo quelle che si chiamano giudizi di tipo, cioè dei meccanismi che vanno a definire il type checker del linguaggio, in modo particolare per le espressioni con quello che in inglese chiamano il giudizio di tipo, la regola di tipo $\Gamma \vdash e : I$ intuitivamente vuol dire: nell'ambiente di tipo in cui ho associato alle variabili presenti in e il tipo descritto da I , se sono in grado di dire che in questo ambiente l'espressione e ha tipo I allora sto controllando la correttezza e sto dicendo anche qual'è il tipo dell'espressione. Dove il tipo dell'espressione è l'etichetta associata all'espressione.



$$\frac{}{\Gamma \vdash n : \perp}$$

\perp Is the least restrictive type
The minimum of the lattice

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

$$\frac{\Gamma \vdash e1 : \ell1, \Gamma \vdash e2 : \ell2}{\Gamma \vdash e1 + e2 : \ell1 \sqcup \ell2}$$



Let $\Gamma(x) = L$ and $\Gamma(y) = H$.
 What is the type of $x+y+1$?

$$\begin{array}{c}
 \frac{\Gamma(x) = L \quad \Gamma(y) = H}{\Gamma \vdash x : L \quad \Gamma \vdash y : H \quad \Gamma \vdash 1 : L} \\
 \\[10pt]
 \hline
 \Gamma \vdash x + y + 1 : H
 \end{array}$$

Noi dobbiamo definire in un ambiente di tipo dove a x viene associato un livello basso e a y viene associata un livello alto, quindi gamma di x è L e gamma di y è H dobbiamo definire qual è il livello di sicurezza, quindi il tipo di $x + y + 1$ quindi il nostro obiettivo è definire in gamma qual è il tipo di x più y più uno allora si si parte dal basso, si va verso l'alto, quindi la prima cosa che dobbiamo fare andiamo a vedere il tipo degli operandi, dobbiamo andare a vedere il tipo di x , il tipo di y e il tipo di uno. Il tipo di uno è facile, diamo subito la risposta è un assioma, non dobbiamo calcolare niente, il tipo di uno è il bottone del reticolo, quindi low perché stiamo considerando il reticolo minore di H . A questo punto per calcolare il tipo della x e il tipo della y dobbiamo andare a determinare con un'operazione di lookup qual è il valore di sicurezza associato a x e qual'è il valore di sicurezza associato a y quindi a questo punto il tipo della x è L questo è quello che ci dice il nostro ambiente dei tipi, il tipo della y è H , a questo punto chiamiamo L join H join L in questo reticolo queste operazioni di join mi da H e quindi il tipo dell'espressione x più y più uno è H . Questa è la il modo formale di scriverlo andiamo a vederlonel dettaglio quando io vado a calcolare x più y più uno devo sommare sostanzialmente il livello di sicurezza della x che è low, il livello di sicurezza della y che è high e il livello di sicurezza della costante uno che è low. Quale sarà mai il livello di sicurezza compatibile con tutti questi 3 livelli che sono low high e low? Sarà high perché sto facendo un'operazione che manipola delle informazioni di livello alto e quindi il tipo di risultato non potranno non essere che alto, non ho un meccanismo di declassificazione.

Typing statements

Judgements of the form

$$\Gamma, \text{ctx} \vdash c$$

Γ is the **environment** mapping variables to security labels
ctx is the **context** label (implicit flow)

Intuitive meaning: According to environment Γ , and context label ctx , statement c is type correct.

adesso dobbiamo fare lo stesso gioco, dobbiamo farlo però per i comandi, quindi dobbiamo definire i giudizi di tipo per i comandi, ma i giudizi di tipo per i comandi devono tener conto anche che ho un flusso隐式的 mentre nell'espressione avevo solamente il flusso esplicito, nei comandi devo tener conto del contesto. Allora cosa vuol dire? Vuol dire che i giudizi di tipo quindi le regole di type checker tengono come parametri un certo numero di informazioni, in modo particolare tengono come informazione l'ambiente di tipo che ha le stesse caratteristiche di come abbiamo visto per le espressioni, il contesto quindi l'ambiente di tipo che caratterizza il flusso esplicito, poi il contesto che mi caratterizza il flusso implicito delle espressioni, il comando di cui dobbiamo definire il livello di sicurezza e poi il risultato dell'operazione di typechecker che sarà il tipo associato. Quindi il contesto è semplicemente il livello di sicurezza implicito delle informazioni contestuali che mi determinano dei flussi di informazione. Intuitivamente quando scrivo gamma contest vdash c mi dice senza metterci niente che il comando c, lo statement c con quelle informazioni di contesto con quell'informazione di ambiente di tipo per i livelli di sicurezza è tipato correttamente, quindi è come se mettessi due punti ok.

Assignment rule

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

INTUITION

$\Gamma(x)$ should be at least as restrictive as $\ell = \Gamma(e)$ (**to prevent explicit flows**) and at least as restrictive as ctx (**to prevent implicit flows**)

va bene, allora andiamo a vedere le regole anche questo caso le regole si leggono dal basso verso l'alto, da sinistra verso destra. Allora siamo in un ambiente gamma, il contesto l'abbiamo definito e dobbiamo vedere se x prende e è corretto. Allora qual è la premessa? Per definire che x è corretta dobbiamo verificare che l'espressione e è ben tipata quindi dobbiamo usare soltanto l'ambiente di tipo perché nelle espressioni abbiamo soltanto flusso esplicito, quindi dobbiamo non considerare nel calcolo del tipo dell'espressione il contesto e andando a vedere che il flusso esplicito associato ad e mi dice che ha un livello di sicurezza I. Allora però I come deve essere utilizzato? I deve essere utilizzato insieme all'informazione contestuale, quindi la definizione del vincolo di sicurezza è che l'espressione x prende e è corretta se il livello di sicurezza della x a cui sto assegnando il valore della valutazione della e deve essere tanto restrittivo quanto il valore di sicurezza del flusso esplicito che I join le informazioni di contesto che mi caratterizza invece il flusso隐式, quindi vuol dire che l'espressione è tipata correttamente, a patto che il livello di sicurezza della x sia informativo quanto il flusso esplicito e il flusso esplicito associato a questo assegnamento. Vuol dire che deve essere tale da prevenire sia i flussi impliciti che i flussi espliciti.

IF Statement

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

$$\frac{\begin{array}{c} \Gamma(y) \sqsubseteq \Gamma(x) \\ \hline \Gamma \vdash y > 0 : \Gamma(y) \quad \Gamma, \Gamma(y) \sqcup L \vdash x := 1 \end{array}}{\Gamma, L \vdash \text{if } y > 0 \text{ then } x := 1 \text{ else } x := 2}$$



$$\frac{\Gamma \vdash y > 0 : \Gamma(y) \quad \frac{\Gamma(y) \sqsubseteq \Gamma(x)}{\Gamma, \Gamma(y) \sqcup L \vdash x := 1} \quad \frac{\Gamma(y) \sqsubseteq \Gamma(x)}{\Gamma, \Gamma(y) \sqcup L \vdash x := 2}}{\Gamma, L \vdash \text{if } y > 0 \text{ then } x := 1 \text{ else } x := 2}$$

QUESTION:

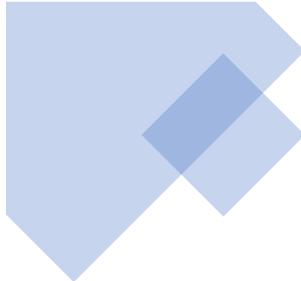
What is the relation between $\Gamma(x)$ and $\Gamma(y)$, such that the above judgement can be proved?

$$\frac{\frac{\Gamma(y) \sqsubseteq \Gamma(x)}{\Gamma \vdash y > 0 : \Gamma(y)} \quad \frac{\Gamma(y) \sqsubseteq \Gamma(x)}{\Gamma, \Gamma(y) \sqcup L \vdash x := 1} \quad \frac{\Gamma(y) \sqsubseteq \Gamma(x)}{\Gamma, \Gamma(y) \sqcup L \vdash x := 2}}{\Gamma, L \vdash \text{if } y > 0 \text{ then } x := 1 \text{ else } x := 2}$$

QUESTION:

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

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


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

$$\Gamma, L \vdash \text{if } z > 0 \text{ then } \{\text{if } y > 0 \text{ then } x := 1 \text{ else } x := 2\} \\ \text{else } \{x := 3\}$$

Adesso andiamo a vedere l'altro esempio che facevamo prima, cioè quello vi ricordate era in cascata con 2 espressioni ifthenelse quella che conteneva la zeta, che poi aveva il ramo then che aveva l'if e quella che conteneva semplicemente il ramo else con x prende tre allora se noi andiamo a vedere il contesto andiamo a vedere il contesto $z = 0$ vale il solito discorso di prima, il tipo associato alla guardia è gamma di x. A questo punto, noi andiamo a tipare l'espressione che corrisponde al ramo then dove nel contesto l'esteso con il tipo di gamma di zeta e poi vedete andiamo a verificare l'assegnamento x uguale a tre nel contesto l'esteso con il flusso implicito del tipo gamma di z, x uguale a 3. Poi dobbiamo fare la stessa operazione che avevamo fatto prima, cioè andare a vedere se il costrutto if y uguale a zero eccetera, eccetera, eccetera è tipato in quell'ambiente e poi quindi sappiamo qual è la condizione che dobbiamo metterci.

While Statement

$$\frac{\Gamma \vdash e : \ell \quad \Gamma, \ell \sqcup ctx \vdash c}{\Gamma, ctx \vdash \text{while } e \text{ do } c}$$

Adesso ci rimane e da verificare il while, noi dobbiamo andare a verificare se un costrutto, while e do c è corretto rispetto ai tipi bene la prima cosa che dobbiamo fare dobbiamo andare a vedere qual è il flusso implicito dell'espressione? E' il solito typechecker che va a determinare il livello I dell'espressione a questo punto dobbiamo andare a valutare il corpo dell'espressione con la stessa strategia ed è che dobbiamo utilizzare le informazioni del flusso implicito dovuto alla valutazione dell'espressione, estendendo il contesto con livello di sicurezza calcolato dal typechecker della guardia del while e se questo ci permette di tipare completamente lo statement che caratterizza il corpo del while allora il while nella sua completa interezza è tipato correttamente.

Sequencing

$$\frac{\Gamma, \textcolor{blue}{ctx} \vdash c1 \quad \Gamma, \textcolor{blue}{ctx} \vdash c2}{\Gamma, \textcolor{blue}{ctx} \vdash c1 ; c2}$$

Infine ci rimane la sequenza la sequenza ci dice la cosa ovvia, se devo vedere la sequenza di comandi $c1$ e $c2$ nel contesto bene quello che devo è che singolarmente i due comandi sono tipati correttamente, se questo è il caso, allora anche la composizione è tipata correttamente

$$\Gamma, \ell \sqcup \Gamma(e) \vdash x := 1 \quad \Gamma, \ell \sqcup \Gamma(e) \vdash x := 2$$

$$\begin{array}{c} \Gamma, \ell \vdash \text{if } e \text{ then } \{x := 1\} \\ \quad \text{else } \{x := 2\} \end{array}$$

$$\Gamma, \ell \vdash x := 3$$

$$\Gamma, \ell \vdash \text{if } e \text{ then } \{x := 1\} \text{ else } \{x := 2\}; \quad x := 3$$

questo è l'esempio dell'ifthenelse di prima scritto nel dettaglio ea questo punto possiamo mettere tutte le regole del typechecker tutte assieme che sono esattamente quello che abbiamo esaminato pocanzi andando a vedere queste regole qual è un ipotesi molto forte che viene fatta sulle espressioni? Quando io definisco il tipo, non considero il flusso implicito perché non metto informazione contestuale che vuol dire che le espressioni non possono avere effetti collaterali

Static type system

Assignment-Rule:

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

If-Rule:

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

While-Rule:

$$\frac{\Gamma \vdash e : \ell \quad \Gamma, \ell \sqcup ctx \vdash c}{\Gamma, ctx \vdash \text{while } e \text{ do } c}$$

Sequence-Rule:

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

Exercize

- Implement the static type system in OCAML
- #1 Define syntax of expressions
- #2 Define syntax of statements
- #3 Define security laticce
- #4 From Typing rules to OCAML code

supponiamo che all'inizio dei tempi, supponendo che questo sia il programma, l'unica cosa che sappiamo è che noi andiamo a verificare all'inizio dei tempi questo programma, avendo come informazioni di contesto il bottom dell'altro, cioè quindi all'inizio dei tempi non abbiamo informazioni contestuale, se non il livello minimo di sicurezza, che questa è un'ipotesi ragionevole.

$$\Gamma, \perp \vdash \mathbf{if } \ x > 0 \ \mathbf{then} \ z := 1 \ \mathbf{else} \ z := 2 \ \mathbf{end}; \ y := z.$$

Note:

the context label ctx can be set to the bottom of the lattice



$$\Gamma, \perp \vdash \text{if } x > 0 \text{ then } z := 1 \text{ else } z := 2 \text{ end;} \quad y := z.$$


SEQUENCING
RULE

$$\begin{aligned} &\Gamma, \perp \vdash \text{if } x > 0 \text{ then } z := 1 \text{ else } z := 2 \text{ end} \\ &\Gamma, \perp \vdash y := z. \end{aligned}$$

allora andiamo a leggere al contrario la cosa e affinché questa espressione che è una sequenza sia verificata, vedete la regola della sequenza dei comandi mi dice che i due comandi, quindi il primo l'ifthenelse e l'assegnamento y prende z devono essere tipati correttamente nello stesso contesto e nello stesso ambiente dei tipi. Quindi vuol dire che con gamma bottom y prende zeta deve essere tipato correttamente e con gamma bottom, if etc... deve essere tipato correttamente e questo è dalla regola della sequenza.

$$\Gamma, \perp \vdash y := z$$

Holds IF

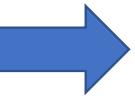
$$\Gamma(z) \sqcup \perp \sqsubseteq \Gamma(y)$$

allora questo punto andiamo a esaminare il secondo assegnamento che è semplice quindi nell'ambiente gamma con informazioni di contesto il minimo del reticolo dobbiamo dire se l'assegnamento y prende z è tipato correttamente allora andando a vedere la regola dell'assegnamento quand'è che gamma nel contesto bottom permette di dire che y è tipato correttamente? Questo vale quando gamma di zeta join bottom è compatibile con il livello di sicurezza della y. Vuol dire che qui il contesto essendo bottom mi dà poca informazione l'unica cosa che mi dice è che vado ad avere un flusso tra z e y, in questo caso è un flusso diretto, il flusso indiretto è bottom quindi è pubblico, quindi non mi comporta nulla, questo vuol dire semplicemente che gamma di zeta deve essere compatibile, semplificando questa condizione di certificazione vuol dire che gamma di zeta deve essere compatibile con livelli sicurezza di y.

$\Gamma, \perp \vdash \text{if } x > 0 \text{ then } z := 1 \text{ else } z := 2 \text{ end}$

HOLDS IF

$\Gamma \vdash x > 0 : \Gamma(x)$

$\Gamma, \perp \sqcup \Gamma(x) \vdash z := 1$  $\Gamma, \Gamma(x) \vdash z := 1$
 $\Gamma, \perp \sqcup \Gamma(x) \vdash z := 2$  $\Gamma(x) \sqsubseteq \Gamma(z)$



$$\frac{\Gamma \vdash x > 0 : \Gamma(x) \quad \frac{\Gamma(x) \sqsubseteq \Gamma(z)}{\Gamma, \Gamma(x) \vdash z := 1} \quad \frac{\Gamma(x) \sqsubseteq \Gamma(z)}{\Gamma, \Gamma(x) \vdash z := 2} \quad \frac{\Gamma(z) \sqsubseteq \Gamma(y)}{\Gamma, \perp \vdash y := z}}{\Gamma, \perp \vdash \text{if } x > 0 \text{ then } z := 1 \text{ else } z := 2 \text{ end}} \quad \frac{\Gamma, \perp \vdash \text{if } x > 0 \text{ then } z := 1 \text{ else } z := 2 \text{ end}; \quad y := z}{\Gamma, \perp \vdash y := z}$$

**Program is well typed if
 $\Gamma(x) = L$, $\Gamma(y) = H$, and $\Gamma(z) = L$**

A questo punto mettendo tutte le informazioni che abbiamo calcolato in questo modo e rimettendole a fatica per chi come me ha dovuto scrivere questo albero di regola, vedete, questo è quello che viene fuori, questo è il modo di scriverlo, mettendo tutta assieme i vari passi che abbiamo detto in questo contesto, quindi adesso possiamo fare un po di giochi, quindi immaginiamo di essere in un contesto di tipi dove x è l , y è h , e z è nell'ambiente di tipo uno si deve domandare se questo programma è ben tipato in questa condizione, andiamo a vedere la condizione che ci si mette, cioè gamma di x è minore di gamma di z , x è l , z è l funziona perché a questo punto vedete le condizioni che abbiamo per quanto riguarda il primo comando sono verificate e quindi il primo comando è tipato correttamente, per quanto riguarda il secondo comando che è l'assegnamento y prende z , dobbiamo semplicemente controllare che gamma di $zeta$ è minore di gamma di y . Gamma di $zeta$ è low, gamma di y è high e a questo punto anche questo è verificato, quindi questo programma è tipato correttamente

$$\frac{\Gamma \vdash x > 0 : \Gamma(x) \quad \frac{\Gamma(x) \sqsubseteq \Gamma(z)}{\Gamma, \Gamma(x) \vdash z := 1} \quad \frac{\Gamma(x) \sqsubseteq \Gamma(z)}{\Gamma, \Gamma(x) \vdash z := 2} \quad \frac{\Gamma(z) \sqsubseteq \Gamma(y)}{\Gamma, \perp \vdash y := z}}{\Gamma, \perp \vdash \text{if } x > 0 \text{ then } z := 1 \text{ else } z := 2 \text{ end}} \quad \frac{\Gamma, \perp \vdash \text{if } x > 0 \text{ then } z := 1 \text{ else } z := 2 \text{ end}; \quad y := z}{\Gamma, \perp \vdash y := z}$$

Program is not well typed if

$\Gamma(x) = L, \Gamma(y) = L, \text{ and } \Gamma(z) = H$

questo stesso programma invece, non è tipato correttamente quando gamma di x è l, gamma di y è l ma gamma di z è uguale ad high e vedete che questo ci porta esattamente in una condizione in modo particolare nella verifica delle condizioni di contesto relative al secondo comando del secondo statement della sequenza, noi dobbiamo controllare se gamma di z è minore a gamma di y, ma gamma di z è h, gamma di y è l e non è verificato perché è il contrario è low che è minore di high e non il viceversa e quindi questo programma con questo assegnamento di valori alle variabili di tipo non è tipato correttamente e quindi questo per far vedere come le condizioni vengono utilizzate.

Type system and Non Interference

If a program does not satisfy NI, then type checking fails.

– Example, where $\Gamma(x) = L$ and $\Gamma(y) = H$:

$$\frac{\Gamma \vdash y : H \quad H \sqcup L \subseteq \Gamma(x)}{\Gamma, L \vdash x := y}$$

Fails

Does not
satisfy NI



Type system and Non Interference

$$\Gamma(x) = L \text{ and } \Gamma(y) = H$$

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

$$\frac{\Gamma \vdash y^* 0 : H \quad H \sqcup L \subseteq \Gamma(x)}{\Gamma, L \vdash x := y^* 0}$$

Fails

Satisfies NI



This type system is not complete.

- c satisfies noninterference $\not\Rightarrow \Gamma, ctx \vdash c$
 - There is a command c , such that noninterference is satisfied, but c is not type correct.
- Example 1:
 - $\Gamma(x) = H, \Gamma(y) = L$
 - c is `if x>0 then y:=1 else y:=1`
- Example 2:
 - $\Gamma(x) = H, \Gamma(y) = L$
 - c is `if 1=1 then y:=1 else y:=x`
- The type system is *conservative*. It has *false negatives*:
 - There are programs that are not type correct, but that satisfy noninterference.

Can we build a complete mechanism?

- Is there an enforcement mechanism for information flow control that has no false negatives?
 - 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>1 then c; l:=2 else skip
 - If we could precisely decide whether this program is secure, we could decide whether **c** terminates!



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 noninterference.

Assume that $\Gamma, \text{ctxt} \vdash \text{skip}$



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

$$\frac{\Gamma \vdash e : \perp \quad \Gamma, ctx \vdash c}{\Gamma, ctx \vdash \text{while } e \text{ do } c}$$

type correctness implies termination sensitive NI

Uno potrebbe dire: come lo facciamo a risolvere questa cosa? Facciamo a risolvere questa cosa andando a modificare leggermente il sistema dei tipi, se noi assumiamo in modo particolare che la guardia del while è di basso livello allora a questo punto il sistema dei tipi garantisce anche la proprietà di termination sensitive e di non interferenza proprio per il fatto che andiamo a fare questa semplice modifica della guardia.

... However

while $h > 0$ do $h = h + 1$; is not well typed!!!

Però abbiamo anche delle limitazioni, ad esempio in modo particolare while h maggiore di 0 do h prende h più 1 dato che la guardia di alto livello non è tipato correttamente, ma questo ovviamente non ha successo vista la proprietà di non interference se sia essa sensitive o no, quindi vedete ci sono un po di ingredienti che modificando alcuni ingredienti la certificazione ha degli effetti positivi o negativi a seconda dal punto di vista in cui uno lo guarda, cioè un modo per far vedere che quando uno progetta deve fare poi tutti i casi possibili

Dynamic mechanisms: decrease false positives over static mechanisms through the use of run-time information.

RECAP DA SLIDE 45 Avevamo affrontato il problema di andare a vedere il problema di avere una nozione di sistema di tipo che permetteva di affrontare anche gli aspetti di coerenza rispetto alla terminazione, ha detto uno delle possibili scelte relativamente a questo aspetto, cioè per rendere il sistema tra virgolette sensitivo, quindi dipendente dalle caratteristiche di terminazione, bisognava modificare la regola del while dove si ammetteva soltanto la possibilità di assegnare un tipo a espressione che era nella guardia del while con tipo più basso del reticolo delle etichette di sicurezza. Infatti, vedete questo corrisponde a definire come regola di typing del while una regola dove un costrutto while è ben tipato se la guardia ha il tipo minimo nel reticolo di sicurezza e a questo punto il corpo del while è tipato nel contesto sia del flusso隐式的 che di quell'esplicito dato dal gamma ovviamente abbiamo un prezzo da pagare, una regola di questo tipo quindi permette di essere sensitivi alla non terminazione ma un prezzo da pagare, ad esempio questa semplice espressione (SLIDE 46) che sicuramente soddisfa la proprietà di non interferenza, non è tipata perché la guardia anche se maggiore di zero ha tipo più alto del bottom e quindi non permette di tipare un'espressione che invece sicuramente non rompe la proprietà di non interferenza. Allora quello che adesso noi andremo a vedere è quello che succede se noi vogliamo adottare dei meccanismi più dinamici che permettono di diminuire il numero di falsi positivi, quindi ovvero di programmi che non sono tipati correttamente benché tuttavia soddisfano proprietà di non interference, utilizzando delle informazioni che sono disponibili a runtime e quindi rilassare gli aspetti di controllo statico dei tipi, permetterci alcune caratteristiche di dinamicità o detto in altri termini, cercare di prendere gli aspetti positivi dell'uso di strumenti di natura dinamica e a questo punto andiamo a vedere come questo si fa e quali sono le caratteristiche del sistema che viene fuori.

Dynamic mechanisms use run time information to decrease false negatives.

A run-time monitor checks/deduces labels along the execution:

- When an assignment $x := e$ is executed
 - either check whether $\Gamma(e) \sqcup ctx \sqsubseteq \Gamma(x)$ holds (fixed Γ)
 - *The execution of a program is halted when a check fails.*
 - or deduce $\Gamma(x)$ such that $\Gamma(e) \sqcup ctx \sqsubseteq \Gamma(x)$ holds (flow-sensitive Γ).
- Monitor maintains a context label ctx .



Dynamic mechanisms use run time information to decrease false negatives.

A run-time monitor checks/deduces labels along the execution:

- When an assignment $x := e$ is executed
 - either check whether $\Gamma(e) \sqcup ctx \sqsubseteq \Gamma(x)$ holds (fixed Γ)
 - *The execution of a program is halted when a check fails.*
 - or deduce $\Gamma(x)$ such that $\Gamma(e) \sqcup ctx \sqsubseteq \Gamma(x)$ holds (flow-sensitive Γ).
- Monitor maintains a context label ctx .
 - When execution enters a conditional command, the monitor augments ctx with the label of the guard.
 - When execution exits a conditional command, ctx is restored.





Assume a dynamic enforcement mechanism with fixed Γ , where $\Gamma(\mathbf{h}) = \mathbf{H}$ and $\Gamma(\mathbf{l}) = \mathbf{L}$.

The diagram illustrates the execution of the following C-like code:

```
if h=0 then h:=2 else h:=3; l:=h
```

The code is divided into three execution blocks based on the value of h :

- Block 1 (ctx = L, h = 0):** The condition $h=0$ is true. The statement $h:=2$ is executed, updating the context to $ctx = H$. The check $ctx \sqcup \Gamma(2) \sqsubseteq \Gamma(h)$ passes, indicated by a green thumbs-up icon.
- Block 2 (ctx = H, h = 2):** The condition $h=0$ is false. The statement $h:=3$ is skipped. The check $ctx \sqcup \Gamma(h) \sqsubseteq \Gamma(l)$ fails, indicated by a red thumbs-down icon.
- Block 3 (ctx = L, h = 2):** The final statement $l:=h$ is executed, updating the context back to $ctx = L$.

andiamo a vedere quest'altro esempio per comprendere bene che cosa vuol dire fare analisi dinamica, in questo esempio vedete, abbiamo un ifthenelse, abbiamo una condizione sempre verificata if $0=0$ then I prende 2 else I prende h, se noi andassimo a verificare solo staticamente questo statement, il sistema non lo tiperebbe perché è evidente che abbiamo un flusso che viola il principio di non interferenza, I prende h è una variabile di basso livello, in un caso, cioè nel caso else permette di avere un flusso non corretto relativamente alla non interferenza tuttavia il ramo else non è mai preso perché l'espressione zero uguale a zero non darà mai il false quindi sicuramente questo è un falso positivo però se noi invece affrontiamo il problema del check dinamico con quello che abbiamo detto poc anzi, allora vedete che sicuramente questo programma passa dinamicamente e viene portato all'esecuzione completa perché quando vado a eseguire l'operazione di I prende due sicuramente l'operazione di assegnamento in un contesto dove i valori del contesto della variabile della guardia che sono gamma uguale a zero solo bottom, quindi a questo punto avremmo bottom su bottom che è minore di bottom e quindi questa operazione viene soddisfatta, quindi il controllo del contesto viene soddisfatto e questo statement viene portato a terminazione, viene eseguito completamente.

Discussion

Under dynamic analysis the command

if $0=0$ then $I:=2$ else $I:=h$

is always executed to completion,

1. because dynamic check $\Gamma(2) \sqcup \Gamma(0=0) \sqsubseteq \Gamma(I)$ always succeeds,
2. because branch $I:=h$ is never taken.

The static type system rejects this program before execution, even though the program is secure!

staticamente questo programma sarebbe rifiutato e dinamicamente andiamo a vedere che cosa succede nel caso che la condizione della guardia si è verificata nell'altro caso in cui la condizione della guardia non è verificata sotto, trovate in questa trasparenza, la simulazione di quello che succede nell' informazione contestuale e nei controlli che devono essere fatti quando viene eseguito il ramo else sopra, andate a vedere la stessa situazione quando viene eseguito il ramo else, allora supponiamo di eseguire l'ifthenelse quando l'informazione di contesto è un informazioni di basso livello e andiamo a controllare se l minore di due, a questo punto abbiamo detto supponiamo di essere nella situazione in cui la guardia è verificata allora a questo punto si estende l'informazione di contesto mettendo le informazioni relative a il valore della guardia, quindi sarebbe il contesto $I \sup H$ che ovviamente continua a essere I e a questo punto bisogna fare il controllo dell'informazione di non interference, ovvero che $I \sup H$ gamma di zero è minore del gamma associato a I , il gamma associato a I è I grande e l'informazione di contesto $I \sup H$ che mi da I a questo punto la condizione della guardia è verificata e vado avanti. Se invece avessi eseguito il ramo else, ovvero la condizione della guardia I minore di due non è verificata avrei fatto la stessa operazione relativa alla modifica del contesto, però a questo punto avrei avuto nella condizione di non interferenza da controllare, che $I \sup H$ gamma di I è minore di gamma di I , la parte destra di questa disequazione è uguale al valore low mentre la parte sinistra è uguale al valore high quindi sicuramente questo check non viene verificato. Quindi vuol dire che usando il meccanismo di enforcement dinamico il controllo dinamico mi permette di eseguire in parte questo comando.

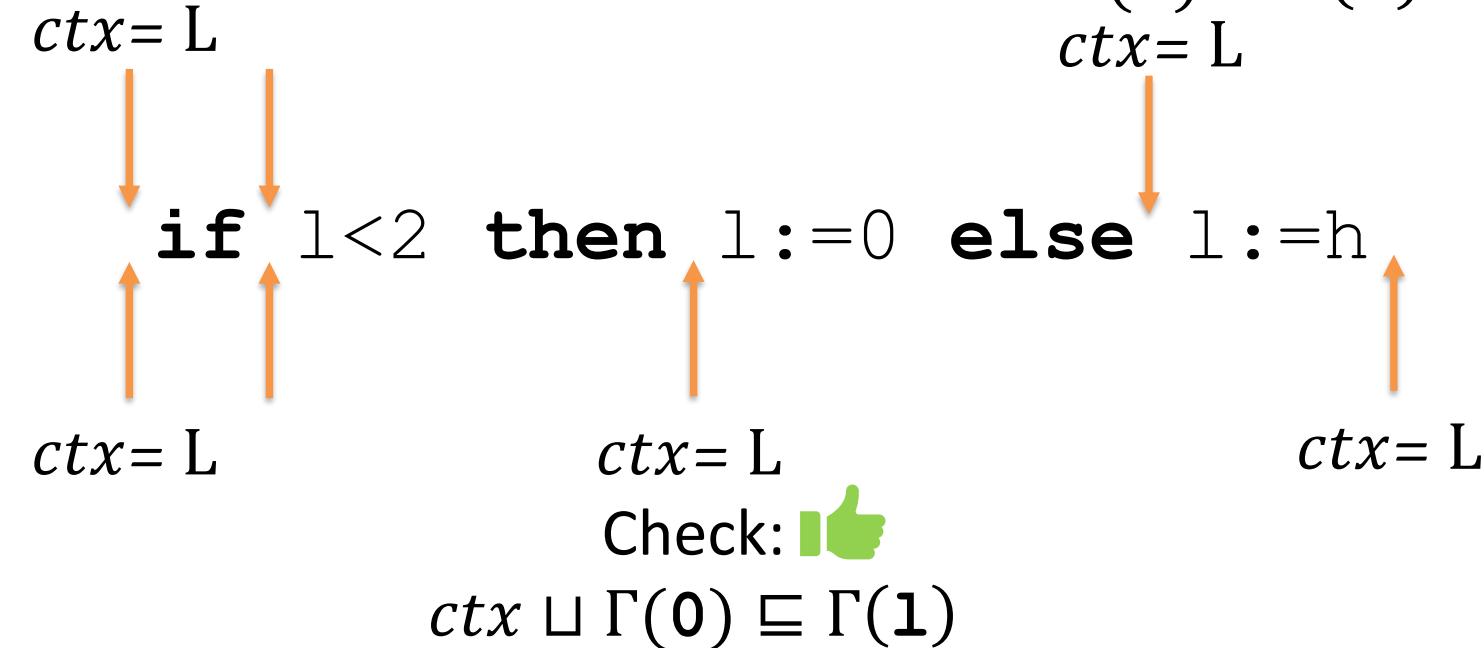
Assume a dynamic enforcement mechanism with fixed Γ ,
where $\Gamma(h) = H, \Gamma(1) = L$.

Execution blocks!

Check:

$$ctx \sqcup \Gamma(h) \sqsubseteq \Gamma(1)$$

$$ctx = L$$



For program

if $I < 2$ then $I := 0$ else $I := h$

If $I < 2$ holds, then command is executed to termination, because
 $\Gamma(0) \sqcup \Gamma(I < 2) \sqsubseteq \Gamma(I)$ succeeds.

If $I < 2$ does not hold, then command is blocked before executing
 $I := h$, because $\Gamma(h) \sqcup \Gamma(I < 2) \sqsubseteq \Gamma(I)$ does not succeed.

Is this program accepted by the static type system?

The static type system rejects this program before execution: all
executions of this program are rejected

Esattamente riscritta in formule è quello che abbiamo detto a voce e c'è anche scritto che se noi avessimo come abbiamo detto in precedenza, analizzato questo programma staticamente il sistema dei tipi ci avrebbe dati picche, proprio per il fatto che il ramo else quando viene eseguito evidentemente viola la condizione di non interference, dinamicamente invece lo esegue il più possibile e quando sono in una situazione di andare a violare la condizione di non interferenza, do una bacchettata nella mano al programma e termina l'esecuzione del programma stesso.



A dynamic mechanism can be more permissive than a static mechanism.



But, there is a
caveat...

A dynamic mechanism may leak
information
when deciding to halt an
execution due to a failed check

abbiamo la possibilità di eseguire più programmi, però abbiamo la possibilità di mandare fuori dei flussi di informazioni che potrebbero essere sgradevoli. Allora per comprendere questo problema, consideriamo questo semplice programmino. Abbiamo questo assegnamento di valori di sicurezza alle variabili che sono presenti nel programma, **h** piccolo è di tipo high, **l** piccolo è di tipo low, il programma semplicemente assegna **p** uguale a zero, poi va a controllare una condizione in un if then else, se il valore della variabile di alto livello è maggiore di zero, allora assegna alla variabile di basso livello il valore uno, altrimenti assegna il valore della variabile di alto livello il valore **h** uguale 1 e poi conclude il programma assegnando 2 alla variabile di basso livello. Allora adesso andiamo a esaminare che cosa succede in questo programma, andando a considerare quando la guardia viene soddisfatta e non. Quando la guardia è minore di zero avremo il ramo else, cioè che **h** viene modificato e il programma termina, e nell'altra condizione invece, che è la condizione corretta, il programma termina normalmente.

Leaking through halting

- Assume fixed Γ : $\Gamma(\mathbf{h})=H$ and $\Gamma(\mathbf{l})=L$.
- Consider program:

```
p:=0;  
if h>0 then l:=1 else h:=1;  
l:=2
```

- At termination with $!(\mathbf{h}>0)$ is *true*, $\Gamma(\mathbf{y}) = \Gamma(\mathbf{x}) = L$.
 - Two public outputs.
- At termination with $\mathbf{h}>0$ is *true*, then execution terminates normally with no public output
- Problem: Even though **h** flows to **x**, **x** is tagged with H only when **h>0**. So, **h>0** is leaked to public outputs

From fixed labels to flow-sensitive labels

A flow-sensitive label on a variable can change during the analysis of the program.

Flow-sensitive labels can be used both in a static or dynamic mechanism.

succede che abbiamo quindi una condizione che dipende dal flusso di esecuzione del programma, cioè abbiamo una condizione che ha un valore della guardia, che ha delle caratteristiche che dipendono dalle condizioni che possono avvenire durante l'esecuzione nel programma, quindi sono delle condizioni che dipendono da come il programma viene eseguito.

Flow sensitive labels

Assume $\Gamma(h) = H$ and $\Gamma(l) = L$. Consider the program

$x := h; x := 0; l := x$

Is this program safe?

- If $\Gamma(x)$ is fixed to H , then the program is rejected, because the type analysis of $l := x$ fails.

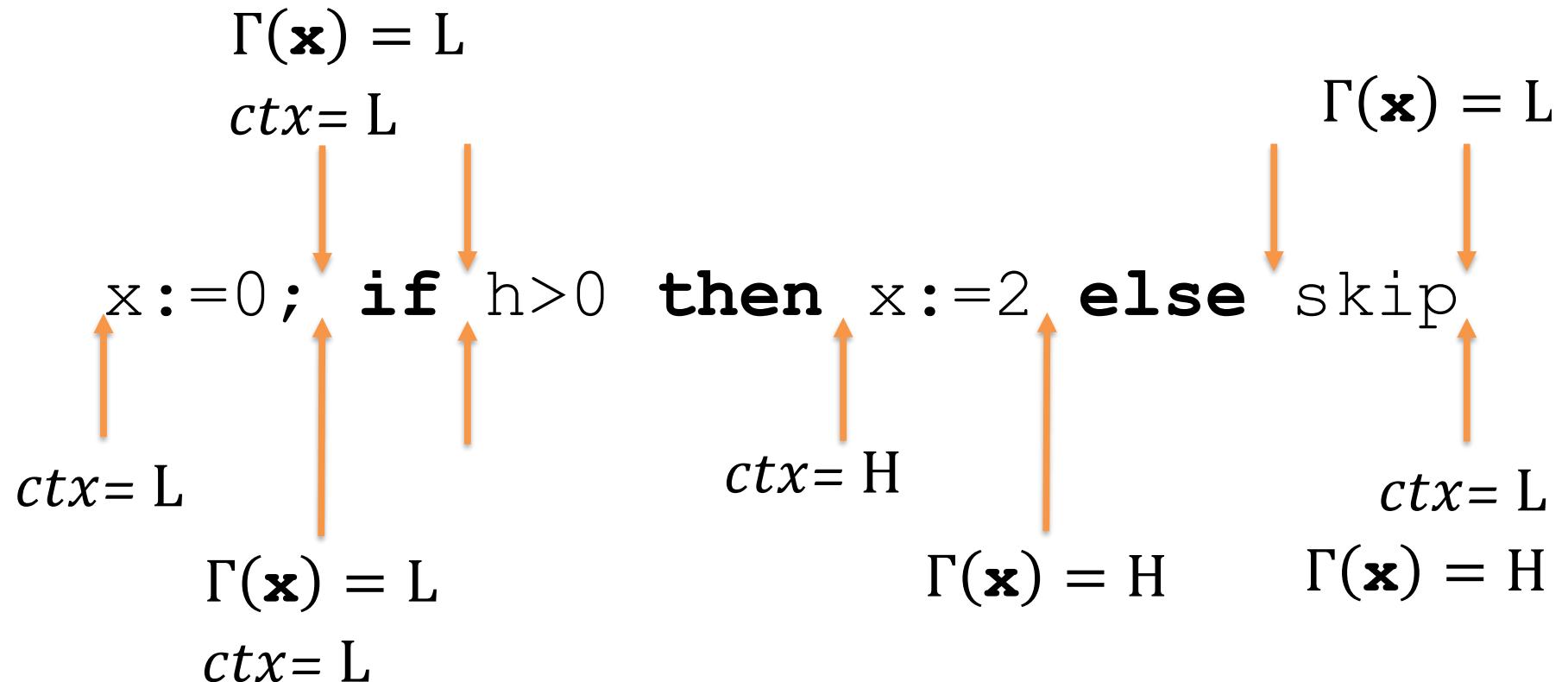
If $\Gamma(x)$ is flow-sensitive, then

- $\Gamma(x)$ becomes H after $x := h$,
- $\Gamma(x) =$ becomes L after $x := 0$,
- the analysis of $l := x$ succeeds.

Flow-sensitive labels can enhance permissiveness even further.



- Assume fixed $\Gamma(\mathbf{h}) = \mathbf{H}$ and flow-sensitive $\Gamma(\mathbf{x})$.



Abbiamo questo sequenza di programma che ci dice x prende zero e poi il comando successivo da eseguire è il costrutto condizionale che fa il controllo sulla variabile h di alto livello e se h maggiore di zero assegna a x due, altrimenti non fa niente, supponiamo di avere un ambiente dei tipi che ci permette di essere fissato su il valore della variabile h piccolo che supponiamo essere di tipo high mentre supponiamo che il valore della variabile x è dipendente dal flusso dell'esecuzione. A questo punto usiamo una strategia di modifica del contesto di run time, quindi supponiamo di considerare il contesto al tempo di esecuzione, andare a vedere che cosa succede nel contesto quando operiamo con l'operazione di if then else e andiamo a eseguire questo comando. Allora supponiamo di partire in una situazione dove eseguiamo x prende zero e con un contesto che è LOW, quando eseguiamo x prende zero, vedete il contesto continua a essere LOW e a causa dell'assegnamento della costante zero alla variabile x, la variabile di sicurezza associata via gamma alla x diventa low perché prende il valore di sicurezza dalla costante che è assegnata. Adesso, ancora una volta nella nostra diapositiva sotto abbiamo il ramo then e sopra abbiamo il ramo else, quindi supponiamo di eseguire la guardia h maggiore di zero e supponiamo che l'h maggiore di zero viene valutata a true, dato che il contesto dipende dal valore di sicurezza della guardia, il contesto diventa HIGH e a questo punto quando facciamo x uguale 2 dobbiamo modificare indipendentemente dall'informazione contestuale il valore di sicurezza dell'etichetta associata a x che a quel punto diventa HIGH. Se andiamo a fare la stessa operazione nel caso then vedete che qui l'informazione di contesto, indipendentemente da cosa sia, skip viene sicuramente eseguito sempre, sia che sia true, high o low sicuramente quando usciamo abbiamo che dobbiamo ripristinare le informazioni su gamma, quindi nel caso del then l'informazione su gamma associata alla x diventa HIGH, nel secondo caso diventa LOW e quindi abbiamo eseguito questa ifthenelse e vedete che in un caso o nell'altro abbiamo dell'informazione della variabile che cambia dipendentemente dal flusso

x:=0; if h>0 then x:=2 else skip

- If **h>0** holds, then after **x:=2**, $\Gamma(x)$ becomes H.
- If **h>0** does not hold, then $\Gamma(x)$ remains L.

This label is not sound!

Problem: Even though **h** flows to **x**,

1. **x** is tagged with H only when **h>0**;
2. **x** is tagged with L otherwise



Solution #1

- Make purely dynamic flow-sensitive mechanism more conservative:
- Block execution before entering conditional commands with high guards.

Back to our example:

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

All execution would stop after $x := 0$.



Solution #2

- **Exploit on-the-fly static analysis to update the labels of target variables in untaken branches to capture implicit flow.**
- **The mechanism is no longer purely dynamic.**

l'altra alternativa, è che attiviamo dinamicamente un controllo dei tipi, quindi facciamo una analisi statica on the fly sul ramo `then` e `else`, per andare a caratterizzare quali sono i valori delle variabili di sicurezza, perché la nostra analisi statica affronta questo. Vuol dire che stiamo mescolando all'interno di un meccanismo dinamico, un meccanismo statico, on the fly, quindi cerchiamo di prendere il beneficio dell'analisi statica che cerca di caratterizzare in blocco tutti i flussi di comportamenti con un meccanismo di analisi dinamica che è più permissivo ma allo stesso tempo vogliamo evitare quelle situazioni di non omogeneità dove la dipendenza dal flusso dell'esecuzione ci causa delle situazioni in cui non siamo in grado di determinare in modo univoco qual è il valore di sicurezza associato a una variabile.

Use on-the-fly static analysis

Problem: **x** was tagged with H only when **h>0** was true, even though **h** always flow to **x**.

Goal: **x** should be tagged with H at every execution.

```
x:=0;
```

```
if h>0 then x:=1 else skip
```

h>0 is evaluated to *false*.

Execute taken branch.

allora facciamo un esempio, l'esempio, è quello che abbiamo visto poc anzi e supponiamo di fare volere avere un analisi statica e vogliamo avere un analisi statiche che ci permette di determinare che effettivamente in tutte le esecuzioni il valore della x è di alto livello perchè skip non serve a niente, cioè ci permette di caratterizzare in modo unico qual è il valore di sicurezza da associare alla variabile x. Allora, se vogliamo che h è valutata a falsa, se h è valutato a falso dobbiamo prendere il ramo else e quindi eseguire l'operazione di skip, allora questo punto, quello che noi dobbiamo fare, invece dobbiamo fare una un'analisi statica on the fly, dobbiamo andare a vedere qual è il valore che noi avremmo assunto per quanto riguarda il livello di sicurezza associato alla x, se avessimo anche eseguito il ramo then e dalla semplice analisi che usa esattamente le condizioni di contesto del sistema di tipi che abbiamo visto per il nostro linguaggio, viene fuori che il valore di sicurezza, quindi l'etichetta che avremmo dovuto associare è h

Use on-the-fly static analysis

Problem: **x** was tagged with H only when **h>0** was true, even though **h** always flow to **x**.

Goal: **x** should be tagged with H at every execution.

```
x:=0;  
if h>0 then x:=1 else skip
```

On-the-fly static analysis:
 $\Gamma(x) = \Gamma(1) \sqcup \Gamma(h>0) = H$

Apply on-the-fly static analysis to the untaken branch.



Use on-the-fly static analysis

Problem: **x** was tagged with H only when **h>0** was true,
even though **h** always flow to **x**.

Goal: **x** should be tagged with H at every execution.

```
x:=0 ;  
if h>0 then x:=1 else skip
```

$$\Gamma(\mathbf{x}) = \mathbf{H}$$

Quali sono i vantaggi e gli svantaggi delle analisi statica e dell'analisi dinamica? Allora l'analisi statica è evidente che ha un basso overhead al run time perché tutti i controlli vengono fatti prima e non vengono creati ad esempio dei nuovi covert channel dovuti a flussi di esecuzione dinamica ed è evidente che l'analisi statica è molto conservativa, cioè in tutte quelle situazioni in cui non riesce a decidere sui valori delle condizioni delle guardie di alto livello blocca perché dice qui ci potrebbe essere la potenzialità di un flusso implicito dovuto al valore di verità che può assumere in una guardia una condizione di alto livello.

Static vs Dynamic

- Static:
 - Low run time overhead.
 - No new covert channels.
 - More conservative.
- Dynamic
 - Increased run time overhead.
 - Possible new covert channels.
 - Less conservative.
- Ongoing research for both static and dynamic.
 - Different expressiveness of policies, different NI versions, different mechanisms

L'avete visto già nel while ifthenelse. Un meccanismo di controllo dinamico, ovviamente, incrementa l'overhead a tempo di esecuzione perché quantomeno va tenuto il contesto, anche se abbiamo gli assegnamenti ai livelli di sicurezza delle variabili fisse. Se vogliamo avere delle informazioni dei livelli di sicurezza delle variabili che dipendono dal flusso dell'esecuzione, dobbiamo non solo mantenere a runtime il contesto, ma dobbiamo dedurre a tempo di esecuzione il valore di sicurezza della variabile in dipendenza delle situazioni in cui andiamo a risolvere delle guardie. Non solo è sicuramente meno conservativo e possono esserci dei leaking, di informazione dovuta alla terminazione, quello che è in questo momento una linea di ricerca che viene seguita da diverse persone e tra poco andremo a esaminare esattamente nel dettaglio una di queste situazioni è quella di cercare di amalgamare controlli statici di information flow con controlli dinamici in modo tale da avere e potere esprimere delle politiche di information flow particolarmente espansive.

Information Flow in Practice

Jif [Myers 1999] Java + Information Flow (originally JFlow)

Jif

andiamo a vedere un esempio velocemente di un sistema che adotta dei meccanismi di information flow, esplicitamente in modo particolare è un'estensione di java che si chiamava jflow e qual era l'idea questo vedete, è il codice è vedete che si sta definendo un metodo che permette di leggere una password, allora questo punto vedete, si mette un valore di sicurezza relativamente al fatto che chi può leggere la password deve essere associato a il livello di sicurezza root,

```
class passwordFile authority(root) {  
    public boolean  
        check (String user, String password)  
    where authority(root) {  
        // Return whether password is correct  
        boolean match = false;  
        try {  
            for (int i = 0; i < names.length; i++) {  
                if (names[i] == user &&  
                    passwords[i] == password) {  
                    match = true;  
                    break;  
                }  
            }  
        }  
        catch (NullPointerException e) {}  
        catch (IndexOutOfBoundsException e) {}  
        return declassify(match, {user; password});  
    }  
    private String [] names;  
    private String {root:} [] passwords;  
}
```

Jif

```
class passwordFile authority(root) {  
    public boolean  
        check (String user, String password)  
    where authority(root) {  
        // Return whether password is correct  
        boolean match = false;  
        try {  
            for (int i = 0; i < names.length; i++) {  
                if (names[i] == user &&  
                    passwords[i] == password) {  
                    match = true;  
                    break;  
                } } }  
        catch (NullPointerException e) {}  
        catch (IndexOutOfBoundsException e) {}  
        return declassify(match, {user; password});  
    }  
    private String [ ] names;  
    private String {root: } [ ] passwords;  
}
```

Security type:
only **root** may
learn
information in
this field

Jif

Declassification:
okay to leak
whether
password
matches

```
class passwordFile authority(root) {  
    public boolean  
        check (String user, String password)  
    where authority(root) {  
        // Return whether password is correct  
        boolean match = false;  
        try {  
            for (int i = 0; i < names.length; i++) {  
                if (names[i] == user &&  
                    passwords[i] == password) {  
                    match = true;  
                    break;  
                }  
            }  
            catch (NullPointerException e) {}  
            catch (IndexOutOfBoundsException e) {}  
            return declassify(match, {user; password});  
        }  
        private String [ ] names;  
        private String { root: } [ ] passwords;  
    }
```

non solo, perchè ci dice anche che esiste un meccanismo di declassificazione, perchè vi sono alcune situazioni in cui io voglio declassificare l'informazione, perché voglio fare fluire delle informazioni dal livello alto al livello basso nei casi in cui è controllato che questa informazione è sensitiva, in modo particolare, ad esempio vedete sto declassificando questo accesso di informazione quando effettivamente è l'utente che è il legittimo proprietario della password che sta andando a fare il login.

l'idea è di avere delle variabili delle annotazione di tipo anche dette label che sono le informazioni che mi dicono com'è fatto il livello di sicurezza associato a una variabile, in modo particolare, vedete in questa trasparenza, ad esempio noi stiamo dichiarando una variabile x intera e diciamo anche qual è il livello di sicurezza associato a questa variabile che ho dichiarato, quindi vuol dire da programma si riesce a definire il livello di sicurezza esattamente high o low a seconda delle caratteristiche che siamo considerando. In modo particolare, queste informazioni sulle variabili possono vincolare il flusso dell'informazione, ad esempio uno potrebbe descrivere un flusso di informazione che va da Alice verso Bob verso Charlie, ovvero che a questo punto l'idea è che uno riesce a creare dei flussi di informazione mettendoci dei vincoli nell'esempio precedenti il vincolo era root verso root con una declassificazione quando era l'utente e similmente uno può avere usando le etichette dei vincoli di scrittura che permettono di descrivere quali sono i vincoli di flusso di informazione che permettono, ad esempio, in questo caso ad alice di scrivere può anche mettere dei meccanismi di declassificazione come nel caso precedente, quando l'utente vuole controllare se effettivamente fare l'accesso con la sua password, quindi questo cosa vuol dire? Vuol dire che complicando un pochino il meccanismo di struttura delle etichette e complicando il sistema di runtime e il sistema dei tipi di java, si riesce a caratterizzare delle proprietà di flusso di informazioni su java.

JIF Type Checking

- Variables (fields, methods, etc.) may have additional **label** as part of their type, e.g., `int {lbl} x;`
- Label constrains information flow to and from variable
 - **reader label:** `alice -> bob, charlie`
 - Alice owns this constraint; her permission required to violate it
 - Alice permits the information to flow **to** Bob and Charlie
 - On previous slide: `root`: is short for `root -> root`
 - **writer label:** `alice <- bob, charlie`
 - Alice owns this constraint;
 - Alice permits the information to flow **from** Bob and Charlie
 - can have multiple such constraints as part of label
 - can read these arrows as the may flow relation →
 - **Decentralized label model (DLM)** [Myers and Liskov 1997]

Information Flow in Practice

JSFlow (Sabelfeld)

<http://www.jsflow.net>

Sperimentation next lecture

JSFlow

- JSFlow adopts **dynamic information flow control**:
 - all runtime values are augmented with a security label taken from L-H lattice.
 - the concept of *no sensitive upgrade (NSU)* is also considered NSU states that security labels are not allowed to change under secret control.
- Tracking the information flow at run-time exploits a *program counter (pc)* abstraction reflecting the current *security context*, i.e. the security level, of the current computation.
 - The security context is upgraded when the program branches on a secret variable.

Andremo a vedere nel dettaglio è un meccanismo di information flow in un caso particolare di un sistema di un linguaggio di scripting javascript e quindi affrontare il problema di information flow non solo nell'applicazione object oriented come Java ma nel caso di applicazioni tipiche, di programmi tipici delle applicazioni web, allora cos'è già javascript flow? E' un'estensione di javascript che adotta un meccanismo di information flow dinamico esattamente nello spirito del meccanismo dinamico che abbiamo visto in precedenza, quindi vuol dire che tutti i valori che sono presenti a tempo di esecuzione sono etichettati con un valore dinamico della etichetta di sicurezza che viene preso da reticolo low o high quindi stiamo prendendo delle etichette prese da un reticolo, low o high e il valore di sicurezza associato ai valori a run time sono presi da questo reticolo. Adotta inoltre una strategia conservativa in modo tale che non possono andare a modificare il valore di sicurezza delle etichette se dipende dalle condizioni di guardia che sono di alto livello, esattamente la strategia che abbiamo discusso in precedenza e questa nel contesto di javascript, viene chiamata non sensitive upgrade, poi vedremo tra poco un esempio nel dettaglio. Il meccanismo sostanzialmente ha un astrazione a tempo di esecuzione che astrae il livello di sicurezza che ho a quel punto dell'esecuzione del programma, quindi ha un'astrazione del flusso di controllo del programma, sostanzialmente è un'astrazione del program counter e ad ogni istante dell'esecuzione, il valore del program counter riflette il livello di sicurezza di quell'esecuzione a quel punto dell'esecuzione, quindi, che cosa ha a tempo di esecuzione? In questa astrazione del flusso viene sempre valorizzato il livello di sicurezza attuale della computazione, notate che questa è esattamente la stessa informazione che quando abbiamo visto la taint analysis e quando la usavamo per andare a caratterizzare dei flussi di informazione implicita, utilizzavamo esattamente la stessa astrazione dove però vi ricordate mettavamo nel program counter chiaramente il livello di taintness cioè il livello di corruzione nel reticolo che stavamo considerando della taintness per caratterizzare se effettivamente stavamo passando nella computazione corrente, in un punto dove c'era una possibilità di corrompere dei dati e quindi l'analogia e il meccanismo è sicuramente simile notate che il contesto viene modificato usando il meccanismo che abbiamo visto in precedenza. Si entrava in un ramo then o in un ramo else e lo si seguiva a seconda del valore di verità della guardia

no sensitive upgrade (example)

```
var lo = true;  
var tm = true;  
if(hi === true)  
  { tm = false; }  
if(tm === true)  
  { lo = false; }
```

NSU is set to false

allora andiamo a vedere che cosa vuol dire no sensitiva upgrade. Supponiamo che non vogliamo adottare la no sensitive upgrade e abbiamo questa sequenza di programmi, abbiamo una variabile di basso livello che è inizializzata true, una variabile temporanea inizializzata true e poi abbiamo una variabile di alto livello che va a controllare se il valore è true, nel caso esegue l'assegnamento temporaneo mettendogli false e poi continua andando a vedere il valore della temporanea true e se il valore della temporanea è true associo alla variabile di basso livello, + il valore false. Allora assumiamo di non volere fare il sensitive upgrade allora a questo punto assumiamo nell'ipotesi di esecuzione che la variabile di alto livello è uguale a true. Se il valore della variabile di alto livello è uguale a true, quando noi andiamo ad eseguire il ramo then che è esattamente il ramo che viene eseguito per l'ipotesi che stiamo considerando sul valore di high, allora l'astrazione che noi abbiamo sul flusso di esecuzione a questo punto ci dice che il valore del program counter, quindi della astrazione di controllo a quel punto è high perché viene aumentato dal valore dell' etichetta, l'etichetta è l'etichetta della guardia. Se vi ricordate, quando entro in un contesto dopo che ho fatto la valutazione, uso il valore della guardia della valutazione e allora a questo punto, ho il valore di sicurezza, notate che quando a questo punto vado avanti, a questo punto tm è uguale a false e quindi viene eseguito questo ramo e quindi a questo punto tm non viene eseguito quindi il valore della variabile low è esattamente uguale al valore di h.

no sensitive upgrade (example)

```
var lo = true;                                NSU is set to false
var tm = true;
if(hi === true) // assume hi == true
  { tm = false; }. // pc = H (upgrade to the label of h
if(tm === true)
  { lo = false; } // lo is equal to the value of h
```

no sensitive upgrade (example)

```
var lo = true;                                NSU is set to false
var tm = true;
if(hi === true) // assume hi == false
  { tm = false; }. // pc = H (upgrade to the label of h
if(tm === true)
  { lo = false; } // lo is equal to the value of h
```

In both cases, the value of **hi** is leaked
to **lo**, with **lo**

allora se noi andiamo a considerare anche il ramo else e vedete è quello che viene eseguito quando questo non viene considerato e quindi si mette a false il valore della variabile di basso livello low e quindi vuol dire che il valore in entrambi i casi il valore della variabile di basso livello corrisponde a il valore high della guardia che viene eseguita qui nel mezzo, quindi abbiamo un leak di informazione, proprio per il fatto che andando a vedere il valore della variabile di basso livello low riesco a vedere qual è il valore dell'h.

no sensitive upgrade (example)

```
var lo = true;                                NSU is set to true
var tm = true;
if(hi === true) // assume hi == true
  { tm = false; }. // this statement is not executed
                    // as it depends on the value of a secret variable
if(tm === true)
  { lo = false; }
```

// If hi is false, lo would be false and public.
// This is a information leak that cannot be prevented with NSU.

a questo punto facciamo la stessa operazione però assumendo non sensitive upgrade, allora dato che noi assumiamo il non sensitive upgrade vuol dire che stiamo eseguendo un ifthenelse che dipende dal valore della guardia che è di alto livello allora vuol dire che questo statement non è eseguito perché dipende dal valore della variabile segreta, quindi a questo punto questo viene bloccato mentre l'altro viene eseguito è evidente che se h è uguale a false e quindi viene eseguito questa parte qui del ramo c'è un piccolo flusso di informazione, ma questo è quello che si deve pagare per avere una maggiore precisione a tempo di esecuzione.

[allora anche quello che noi abbiamo fatto è andare a vedere sul sito del javascript flow, dove ci sono tutorial e un po' di esempi e abbiamo visto un challenge che è sostanzialmente andare a comprendere come l'esecuzione e come nel modello dell' information flow in questi 10 esempi, quindi vi fa vedere in pratica cosa succede relativamente all'uso di tecniche di information flow. abbiamo 12 variabili, sei variabili di alto livello e sei variabili di basso livello, inizialmente solo due label che sono high e low. Tutte le variabili che iniziano con la h sono high, tutte le variabili che iniziano con un'altra lettera sono low. Ad ogni livello cambia il type system. Nel primo livello il type system non ha regole, ha solo questo assioma |- c che ci dice che sostanzialmente tutto è lecito e possibile e nei prossimi step questo system cambierà quindi impedendo, ad esempio, explicit flow, implicit flow e così via. Lo scopo è quello di, tra virgolette, rompere questo type system e assegnare il contenuto di ogni variabile di tipo high a quelle rispettive di tipo low, quindi il contenuto di h1 a l1, di h2 ad l2 e così via. Il linguaggio di programmazione è molto semplice, la sintassi è qui sotto comunque è semplicissimo. Il type system in questo caso non ha regole granché particolari, quindi mi è permesso fare una cosa del genere, assegnare direttamente h1 a l1, h2 a l2 e così via. Nel momento in cui mando il codice vengono effettuati due test con valori casuali e effettivamente viene fatto il confronto fra la memoria low finale e quella iniziale e vengono confrontati per vedere che sono effettivamente uguali. Lo scopo dello step successivo è invece impedire l' explicit flow che avevamo prima. Usiamo un if per ogni variabile h e assegniamolo in base al valore di h, cioè facciamo if h1 then l1 = true else l1 = false per ogni h. Qui non cambia solo il typesystem ma anche il linguaggio, viene aggiunta una nuova operazione declassify che quello che fa è estrarre il valore di h1, solo h1 c'è nella regola del type system. Il valore di h1 può essere estratto tramite declassify, assegnare un valore declassificato non è consentito nel if e nei loop con le guardie di tipo high, entrambi i branch del then e dell'else dell' if devono avere lo stesso numero di step e lo skip e l'assignment contano come uno step. Se declassify fosse più generale, nel senso che potessi usarlo per tutte le h, allora la cosa più naturale sarebbe fare l1 declassify h1, l2 declassify h2 e così via. E questo funzionerebbe se, appunto declassify funzionasse solo per tutte le h non solo per h1. Posso assegnare di volta in volta il valore di hx alla variabile di h1. Sfrutto il fatto che posso declassificare solo h1 quindi ad h1 assegno h2.](http://ifc-challenge.appspot.com/steps/start)

```

l1 = declassify(h1);
h1 = h2; l2 = declassify(h1);
h1 = h3; l3 = declassify(h1);
h1 = h4; l4 = declassify(h1);
h1 = h5; l5 = declassify(h1);
h1 = h5; l6 = declassify(h1);
h1 = h6;

```

Nel 4 step oltre a cambiare il type system cambia anche il linguaggio, in questo caso abbiamo aggiunto le eccezioni, quindi meccanismi try catch e così via. Se faccio una throw dentro un if oppure un while con una guardia alta allora il throw è considerato alto quindi high, dentro un blocco catch non posso assegnare variabile low se il try block ha una high throw. Se h1 è true faccio skip e quindi l1 resta true altrimenti faccio throw e assegno false a l1 nel blocco catch. Però il typesystem ci dice che nel catch block non posso assegnare alle variabili low se nel try block ho una throw di tipo high perché è in un if con una guardia high.

```

try{
    l1 = false
    if(h1){
        skip;
    }else{
        throw;
    }
    l2 = true;//COSÌ DEVE ANDARE!!!
}catch {
    //l1 = true; non va bene!!!
    skip
}

```

Il punto qual è? Dentro il blocco catch non posso fare assegnamento, quindi l1=false non va bene. L'assegnamento a l1 lo faccio nel blocco try e nel blocco catch faccio uno skip. Perché questo è valido? Perché inizializza l1 a true, se h1 è true allora faccio skip, e l1 ha il valore di h1, se false faccio lo skip che invece è consentito. Nello step 5 il senso del Let è che posso dare un valore alla x, che è quello specificato dal valore dell'espressione e usare questa x nello statement che è dopo l'in. Possiamo scrivere:

```
let (x=h1) in {l1 = x;}
```

Il typesystem ci dice che gli statement nel let block sono type checked, quindi il controllo dei tipi degli statement del let, viene fatto come viene fatto per il resto di altri linguaggi. Gli assignment nel let block sono sempre consentiti e il controllo di questo comando è come avveniva prima. Nello step 6 la sintassi del linguaggio viene estesa con le procedure, quindi è possibile dichiarare le procedure in questo modo: declare proc che prende una variabile di input con il livello, e una variabile di output sempre con una label e poi il corpo della procedura. I livelli sono high e low in corrispondenza delle label che abbiamo e poi possiamo chiamare la procedura con il nome della procedura e un valore, un'espressione che indica un valore per la variabile di ingresso e una variabile di uscita. Le procedure possono essere dichiarate all'inizio del programma perché diciamo una grammatica lo impone, le variabili dentro le procedure sono locali. Nel corpo delle procedure le variabili che possiamo dichiarare sono locali e seguono la regola che sono di tipo high se iniziano con la h, mentre il livello, il tipo delle variabili di out sono specificati nella dichiarazione e le procedure possono restituire un valore tramite un assegnamento alla variabile di uscita. È possibile leggere con un if all'interno del programma la procedura restituisce qualcosa di low, e in input prende una variabile high di cui vogliamo estrarre il valore. Avevo pensato di usare una variabile da appoggio, ma penso che il risultato non cambi comunque si perché poi alla fine c'è da sempre l' assegnamento quindi non va. Il codice a cui avevo pensato era: declare proc p(in x : high, out y: low) {if (x) {y=true;} else {y=false;} } p(h1,l1) Non è consentito, è una regola che era stata introdotta negli step precedenti. Si potrebbe pensare a 2 procedure che ritornano una true e una false in base al valore di x:

```

declare proc t(in x: low, out y : low) {y = true;}
declare proc f(in x: low, out y : low) {y = false;}
if (h1) t(true,l1); else f(true,l1)

```

Sostanzialmente l' assegnamento che nello step 2 facevo nel ramo then o else adesso lo faccio tramite la procedura,

