SICUREZZA BASATA SUI LINGUAGGI

NON-INTERFERENZA

Isabella Mastroeni

Language-based security

PROBLEMA CRITICO: PROTEZIONE DELL'INFORMAZIONE LEGATO ALLA SICUREZZA NEI COMPUTER!

Language-based security

PROBLEMA CRITICO: PROTEZIONE DELL'INFORMAZIONE LEGATO ALLA SICUREZZA NEI COMPUTER!

La sicurezza può essere distinta in tre categorie

- DISPONIBILITÀ: Possibilità di usare le risorse o l'informazione desiderata:

  Una politica di sicurezza per garantire la disponibilità deve garantire che, dato un insieme di entità X e una risorsa I, tutti i membri di X possono accedere a I.

Language-based security

PROBLEMA CRITICO: PROTEZIONE DELL'INFORMAZIONE LEGATO ALLA SICUREZZA NEI COMPUTER!

La sicurezza può essere distinta in tre categorie

- DISPONIBILITÀ: Possibilità di usare le risorse o l'informazione desiderata:
- INTEGRITÀ: Affidabilità di dati e risorse:

  Una politica di sicurezza per garantire l'integrità deve garantire che dato un insieme di entità X ed una risorsa o informazione I, tutti i membri di X reputano I affidabile.

Language-based security

PROBLEMA CRITICO: PROTEZIONE DELL'INFORMAZIONE LEGATO ALLA SICUREZZA NEI COMPUTER!

La sicurezza può essere distinta in tre categorie

- DISPONIBILITÀ: Possibilità di usare le risorse o l'informazione desiderata:
- INTEGRITÀ: Affidabilità di dati e risorse:
- CONFIDENZIALITÀ: Protezione di informazione o risorse considerate private:

  Una politica di sicurezza, per garantire la confidenzialità, deve garantire che dato un insieme di entità X e qualche informazione I, allora nessun membro di X può ottenere alcuna informazione riguardante I.

Confidenzialità come Non-Interferenza

COME SI PROTEGGE LA CONFIDENZIALITÀ?

Confidenzialità come Non-Interferenza

COME SI PROTEGGE LA CONFIDENZIALITÀ?

- CONTROLLO DEGLI ACCESSI: Gestisce chi può accedere all'informazione protetta;

Confidenzialità come Non-Interferenza

COME SI PROTEGGE LA CONFIDENZIALITÀ?

- CONTROLLO DEGLI ACCESSI: Gestisce chi può accedere all'informazione protetta;
- CRITTOGRAFIA: Protegge l'informazione rendendola incomprensibile

Confidenzialità come Non-Interferenza

COME SI PROTEGGE LA CONFIDENZIALITÀ?

- CONTROLLO DEGLI ACCESSI: Gestisce chi può accedere all'informazione protetta;
- CRITTOGRAFIA: Protegge l'informazione rendendola incomprensibile
- AUTENTICAZIONE: Processo che verifica quando unutente vuole convincere un verificatore della sua identità

Confidenzialità come Non-Interferenza

COME SI PROTEGGE LA CONFIDENZIALITÀ?

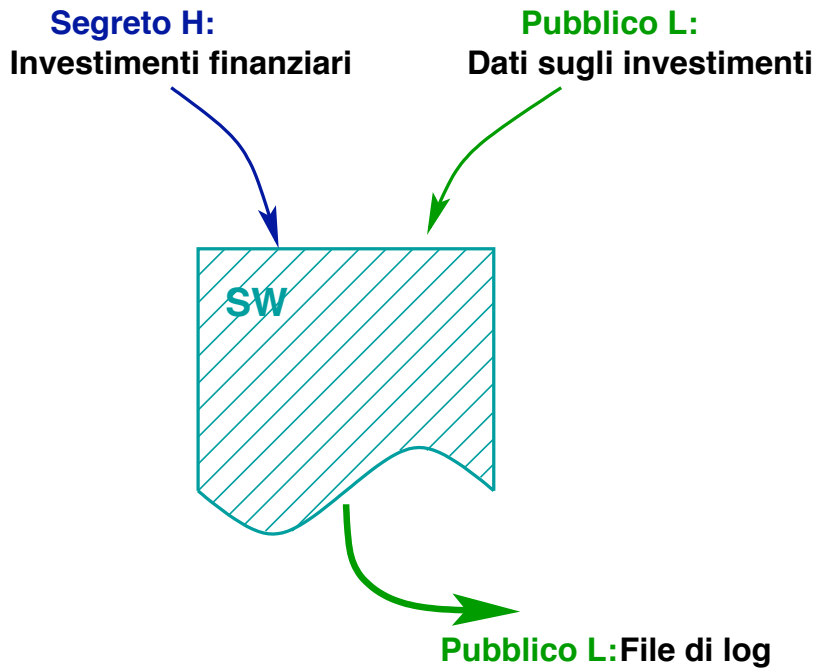
- CONTROLLO DEGLI ACCESSI: Gestisce chi può accedere all'informazione protetta;
- CRITTOGRAFIA: Protegge l'informazione rendendola incomprensibile
- AUTENTICAZIONE: Processo che verifica quando unutente vuole convincere un verificatore della sua identità

Queste tecniche possono vincolare i diritti di un utente, ma non possono vincolare il *flusso di informazione* nel sistema!

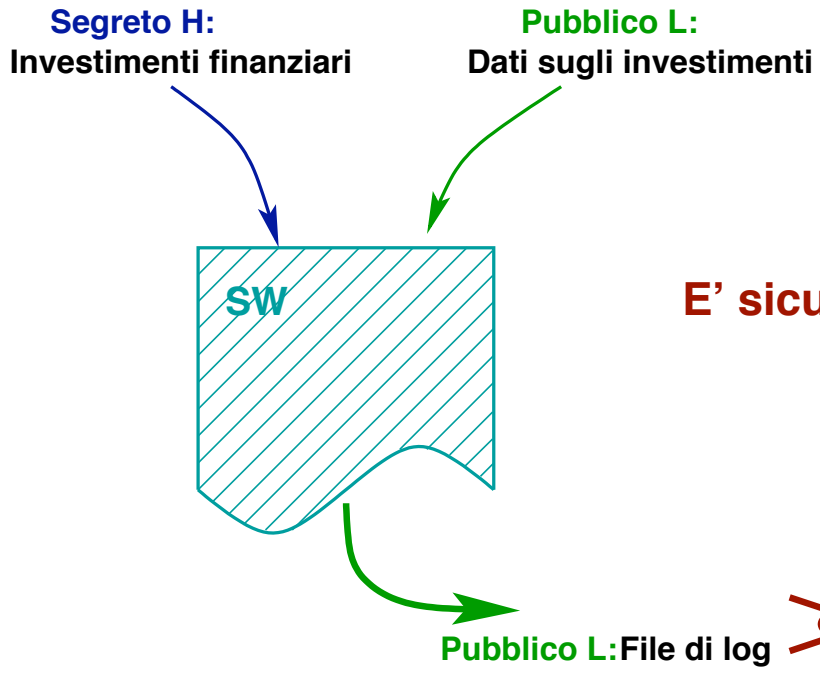
⇓

ABBIAMO BISOGNO DI UNA POLITICA CHE VINCOLA IL FLUSSO DI INFORMAZIONE DEFINENDO IL MODO IN CUI L'INFORMAZIONE SI PUÒ MUOVERE ALL'INTERNO DEL SISTEMA

Il problema: la Non-Interferenza



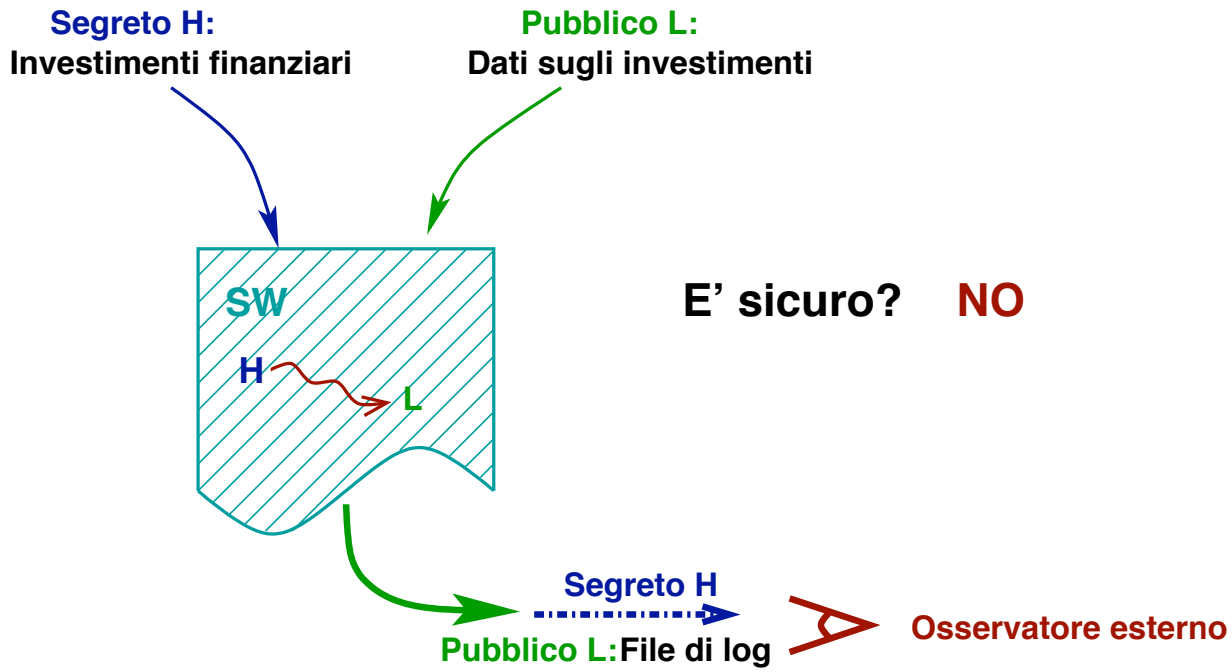
Il problema: la Non-Interferenza



E' sicuro?

Osservatore esterno

Il problema: la Non-Interferenza



DEFINIRE

LA

NON-INTERFERENZA

Definire la Non-Interferenza

PROPRIETÀ DI SICUREZZA: Determina quali classi non devono interferire con quali altre classi.

Definire la Non-Interferenza

PROPRIETÀ DI SICUREZZA: Determina quali classi non devono interferire con quali altre classi.

- Problema di confinamento [Lampson'73]: *Prevenire che i risultati delle computazioni rilascino informazioni anche parziali sugli input confidenziali.*

Definire la Non-Interferenza

PROPRIETÀ DI SICUREZZA: Determina quali classi non devono interferire con quali altre classi.

- Problema di confinamento [Lampson'73]: *Prevenire che i risultati delle computazioni rilascino informazioni anche parziali sugli input confidenziali.*

$\Longrightarrow$ Nozione intuizionista... poco formale!

Dipendenza forte [Cohen'77]

Sia $\mathtt{P}$ un programma e $[\![\mathtt{P}]\!]$ la sua semantica, siano $\mathtt{L}$ e $\mathtt{H}$ rispettivamente l'insieme *pubblico* e *privato* delle variabili.

- Valori differenti di un input a possono risultare in un insieme di valori differenti di un output b dopo l'esecuzione di $\mathtt{P}$:
  $\Rightarrow$ b dipende fortemente da a
- Se $a = \mathtt{H}$ dato privato e $b = \mathtt{L}$ dato pubblico allora abbiamo un problema di confidenzialità.

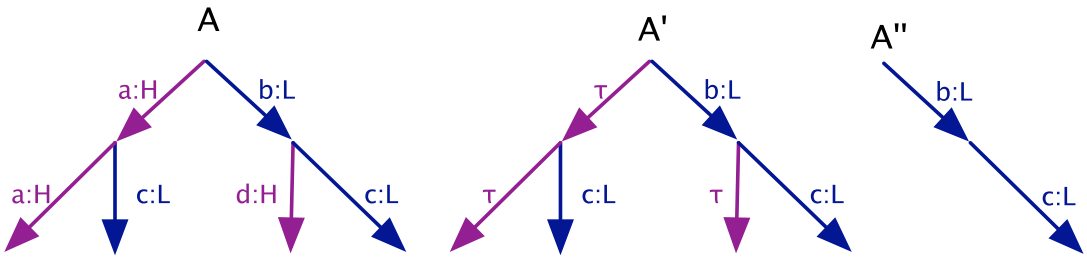
$$\mathtt{H} \rhd^{\mathtt{P}} \mathtt{L} \text{ iff } \exists s_1, s_2 \,.\, {s_1}^{\mathtt{L}} = {s_2}^{\mathtt{L}} \wedge [\![\mathtt{P}]\!](s_1)^{\mathtt{L}} \neq [\![\mathtt{P}]\!](s_2)^{\mathtt{L}}$$

Non-interferenza [Goguen & Meseguer'82]

Un certo gruppo di utenti G che esegue un certo insieme H di operazioni non interferesce con, i.e., non può essere individuato da, un altro gruppo di utenti G' che esegue un insieme L di comandi di output.

$$G, H :| G', L \text{ iff}$$
$$\forall w \in (U \times C)^*, v \in G', l \in L.\ \textbf{out}([\![w]\!], v, l) = \textbf{out}([\![P_{G,H}(w)]\!], v, l)$$

dove $P_{G,H}(w)$ è la sequenza ottenuta da w eliminando tutte le occorrenze delle coppie (u, c) con $u \in G$ e $c \in H$.



Approccio semantico [Joshi & Leino'00]

Un programma è sicuro se ogni osservazione dei valori iniziali e finali delle variabili pubbliche ($l : L$) non fornisce alcuna informazione sui valori iniziali delle variabili private ($h : H$).

Sia HH ="assegna ad h un valore arbitrario", e P un programma

P è *sicuro* sse $HH\ ;\ P;\ HH \doteq P\ ;\ HH$

⇒ *I due programmi sono I/O equivalenti se il valore finale di l non dipende da quello iniziale di h.*

PER model [Sabelfeld & Sands'01]

Interpretiamo ora *tipi di sicurezza* come *relazioni di equivalenza parziali* (PER) sulle funzioni semantiche.
Let $\forall x, x'.\ x\ \mathit{All}\ x'$ e $x\ \mathit{Id}\ x' \Leftrightarrow x = x'$.

$$P \text{ è } \textit{sicuro} \quad \text{sse}$$
$$\forall s, t\ .\ \langle s^H, s^L\rangle\ \mathit{All} \times \mathit{Id}\ \langle t^H, t^L\rangle \Rightarrow [\![P]\!](s)\ \mathit{All} \times \mathit{Id}\ [\![P]\!](t)$$

- P e Q relazioni di equivalenza:
  $f\ (P \rightarrowtriangle Q)\ g \Leftrightarrow \forall x, x' \in D\ .\ x\ P\ x' \Rightarrow f(x)\ Q\ g(x')$;
- $(\mathit{All} \times \mathit{Id} \rightarrowtriangle \mathit{All} \times \mathit{Id})$ è *riflessiva* su $[\![P]\!]$ sse P sicuro.

CONVALIDARE

LA

NON-INTERFERENZA

Verificare la Non-Interferenza

Il confidare sul fatto che un sistema sia sicuro, rispetto alla confidenzialità, dovrebbe derivare da un'analisi rigorosa che mostra che il sistema nella sua interezza, soddisfa le politiche di sicurezza dei suoi utenti.

Verificare la Non-Interferenza

Il confidare sul fatto che un sistema sia sicuro, rispetto alla confidenzialità, dovrebbe derivare da un'analisi rigorosa che mostra che il sistema nella sua interezza, soddisfa le politiche di sicurezza dei suoi utenti.

- Meccanismi standard di sicurezza: Controllo degli accessi.

  Ogni *soggetto* ha un *diritto* e ogni *oggetto* ha una *classificazione*. Ogni oggetto ha un livello di sicurezza che non deve superare i diritti del soggetto [Bell & LaPadula'73].

⇓

NON C'È CONTROLLO DELLA PROPAGAZIONE DELL'INFORMAZIONE!

Analisi statica dei flussi di informazione [Denning & Denning'76]

Si consideri un modello dei flussi di informazione $\mathcal{F}$ definito come $\mathcal{F} = \langle \mathrm{N}, \mathrm{P}, \mathcal{S}, \oplus, \rightarrow \rangle$, dove $\mathrm{N} = \{\mathrm{a}, \mathrm{b}, \ldots\}$ è un onsieme di oggetti, $\mathrm{P} = \{\mathrm{p}, \mathrm{q}, \ldots\}$ è un insieme di processi, che sono agenti attivi responsabili del flusso di informazione.

L'informazione fluisce dalla classe A *alla classe* B *quando l'informazione associata ad* A *ha effetti sul valore dell'informazione associata a* B

$\Downarrow$

Un modello di flusso $\mathcal{F}$ è sicuro se e solo se l'esecuzione di una sequenza di operazioni non può dare origine ad un flusso che viola la relazione $\rightarrow$.

Analisi statica dei flussi di informazione [Denning & Denning'76]

L'informazione fluisce dalla classe A *alla classe* B *quando l'informazione associata ad* A *ha effetti sul valore dell'informazione associata a* B

$\Downarrow$

Un modello di flusso $\mathcal{F}$ è sicuro se e solo se l'esecuzione di una sequenza di operazioni non può dare origine ad un flusso che viola la relazione $\rightarrow$.

Si crea un programma astratto e lo si valuta ricorsivamente:

- Un assegnamento è sicuro se tutti i flussi dalle variabili usate a quella assegnata sono ammessi dalla politica
- $S_1; S_2$ è sicuro se lo sono S_1 ed S_2
- In presenza di una guardia, il programma è sicuro se il flusso dalle variabili della guardia ad ogni variabile modificata è ammesso dalla politica.

Tipi per la sicurezza [Volpano et al.'96]

Un *sistema di tipi per la sicurezza* è una collezione di regole di inferenza e assiomi per derivare tipi di sicurezza:

$$\gamma \vdash p : \tau$$

Tipi per la sicurezza [Volpano et al.'96]

Un *sistema di tipi per la sicurezza* è una collezione di regole di inferenza e assiomi per derivare tipi di sicurezza:

$$\gamma \vdash p : \tau$$

$$\gamma \vdash n : \tau \qquad \gamma \vdash x : \tau\ var \qquad \frac{\gamma \vdash e : \tau\ var}{\gamma \vdash e : \tau} \qquad \frac{\gamma \vdash x : \tau\ var \quad \gamma \vdash e : \tau}{\gamma \vdash x := e : \tau\ cmd}$$

$$\frac{\gamma \vdash c_1 : \tau\ cmd \quad \gamma \vdash c_2 : \tau\ cmd}{\gamma \vdash c_1 ; c_2 : \tau\ cmd} \qquad \frac{\gamma \vdash e : \tau \quad \gamma \vdash c : \tau\ cmd \quad \gamma \vdash c' : \tau\ cmd}{\gamma \vdash \textbf{if } e \textbf{ then } c \textbf{ else } c' : \tau\ cmd}$$

$$\frac{\gamma \vdash e : \tau \quad \gamma \vdash c : \tau\ cmd}{\gamma \vdash \textbf{while } e \textbf{ do } c}$$

Approccio assiomatico

- Derivazione di dimostrazioni di flusso [Andrews & Reitman'80]: Questo approccio deriva asserzioni $\{P\}\ S\ \{Q\}$ costruite applicando assiomi di flusso e regole di inferenza:

$$\frac{A_1, \ldots, A_n}{B}$$

Approccio assiomatico

- Derivazione di dimostrazioni di flusso [Andrews & Reitman'80]
- Logica tipo Hoare [Amtoft & Banerjee'04,'06]: La confidenzialità è trattata come *indipendenza* di variabili, e le tracce del programma, potenzialmente infinite, sono astratte mediante interpretazione astratta da un insieme finito di indipendenze:

$$G \vdash \{T_1^\#\}\ C\ \{T_2^\#\}$$

$$[\text{Assign}]\ G \vdash \{T_0^\#\} x := e \{T^\#\}$$

If $\forall [y \# w] \in T^\#$.
$(x \neq y \ \Rightarrow\ [y \# w] \in T_0^\#)$,
$(x = y \ \Rightarrow\ w \notin G \ \wedge\ \forall z$ free variable in e. . $[z \# w \in T_0^\#])$

LA

NON-INTERFERENZA

NEI DIVERSI SISTEMI DI CALCOLO

Linguaggi imperativi deterministici

- Linguaggi imperativi deterministici: *Un programma* P *soddisfa la proprietà di* non interferenza *se per tutte le memorie* μ *e* ν *tali che* $\mu(l) = \nu(l)$ *per tutte le variabili* l*, e tali che* P *termina partendo da* μ *and* ν*, portando, rispettivamente, a* μ' *e a* ν'*, allora abbiamo* $\mu'(l) = \nu'(l)$ *per tutte le variabili* l [Volpano'96]:

while $\neg(mask = 0)$
  if $\neg(\text{PIN} \;\&\; mask = 0)$ (bitwise *and*)
    $y := y \mid mask;$ (bitwise *or*)
  $mask := mask/2;$

$$\forall s_1, s_2 \in \Sigma \,.\, s_1 =_L s_2 \;\Rightarrow\; [\![P]\!](s_1) =_L [\![P]\!](s_2)$$

Linguaggi non-deterministici e multi-threaded

Un programma non-deterministico P *soddisfa la proprietà di* non-interferenza possibilistica *se per tutte le memorie* μ *e* ν *tali che* $\mu(l) = \nu(l)$ *per tutte le variabli pubbliche* l*, e* P *può terminare a partire da* μ *portando allo stato finale* μ'*, allora esiste uno stato* ν' *tale che* P *può terminare partendo da* ν *e portando a* ν' *con* $\mu(l) = \nu'(l)$ *per tutte le variabili pubbliche* l [Volpano'96]:

Thread α:	Thread β:	Thread γ:
$y := x;$	$y := 0;$	$y := 1;$

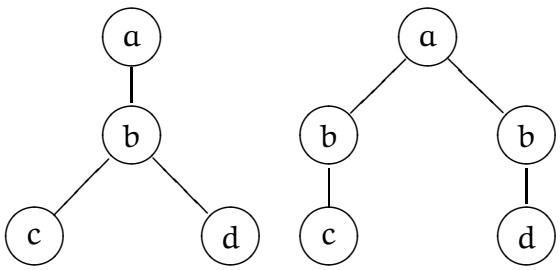
dove $x : H$ binaria, $y : L$

Algebre dei processi

Siano G *e* G' *due gruppi di utenti, data una qualunque sequenza di input* γ*, sia* γ' *la sua sottosequenza ottenuta cancellando tutte le azioni degli utenti in* G*;* G *soddisfa la non-interferenza con* G' *sse per ogni sequenza di input* γ*, gli utenti di* G' *ottengono lo stesso output dopo l'esecuzione di* γ *e di* γ' [Focardi & Gorrieri'95]:

$$A/Act_H \approx_T A \backslash Act_H$$

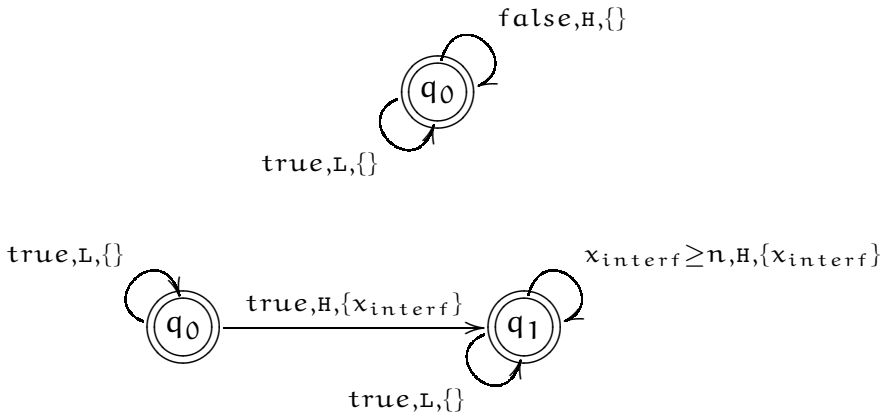
- Proprietà di sicurezza basate sulle tracce: NNI, SNNI, NDC;
- Proprietà di sicurezza basate sulla bisimulazione: BNNI, BSNNI, BNDC.



Automi temporizzati

Azioni private non interferiscono con il sistema, considerando un ritardo minimo n, se il comportamento del sistema in assenza di azioni private è equivalente al comportamento del sistema, osservato sulle azioni pubbliche, quando le azioni private possono essere eseguite con un ritardo minimo tra loro di almento n [Barbuti et al.'02]:

$$(A||Interf_H^n)/H \approx A||Inhib_H \Leftrightarrow [\![A]\!]_H^n/H = [\![A]\!]\backslash L$$



NON-INTERFERENZA

E

COVERT CHANNELS

Canali coperti

CANALE COPERTO: *Consiste in un canale la cui funzione non è quella di trasferire informazione ma di fatto lo fa*.
[Lampson'73]

Canali coperti

CANALE COPERTO: *Consiste in un canale la cui funzione non è quella di trasferire informazione ma di fatto lo fa*.
[Lampson'73]

- Canali impliciti: Canali di informazione dovuti alle strutture di controllo dei programmi:

  while $\mathtt{h}$ **do** $\mathtt{l} := \mathtt{l} + 1; \mathtt{h} := \mathtt{h} - 1$ **endw**

Sicurezza basata sui linguaggi – p.22/32

Canali di terminazione

Canali di informazione dovuti allo stato di terminazione o meno del programma:

while $\mathtt{h}$ **do** $\mathtt{h} := \mathtt{h} + 1$ **endw**

- [Sabelfeld & Sands'01]: Si consideri, nel PER model, una semantica denotazionale termination-sensitive: $[\![\mathtt{P}]\!] = \bot$ se $\mathtt{P}$ non termina;
- [Volpano & Smith'97]: Sistemi di tipi con tipo minimo.

Canali di tempo

Canali di informazione dovuti al momento in cui una azione occorre invece che al contenuto dell'azione stessa. L'azione può essere la terminazione:

while $\mathtt{h}$ **do** $\mathtt{l} := 2\mathtt{l} - \mathtt{l}; \mathtt{h} := \mathtt{h} - 1$ **endw**

- [Volpano & Smith'99]: restrizione dei condizionali con guardia privata;
- [Agat'00]: La *trasformazione di programmi* viene usata in modo da chiudere eventuali canali di tempo.

Canali di probabilistici

Canali di informazione dovuti al cambiamento della distribuzione di probabilità dei dati osservabili. Questi cambiamenti sono pericolosi quando l'attaccante può eseguire ripetutamente una computazione ed osservare il suo comportamento stocastico:

Thread α:	Thread β:
$y := x;$	$y :=$ random$(100);$

- [Volpano & Smith'99]: Sistema di tipi per programmi multi-threaded probabilistici;
- [Sabelfeld & Sands'00]: PER model su powerdomain probabilistici.

INDEBOLIRE LA NON-INTERFERENZA

Indebolire la Non-Interferenza

Le politiche di richiedono che ogni *variazione dei dati confidenziali non sia rivelato attraverso l'osservazione dei dati pubblici.*

Indebolire la Non-Interferenza

Le politiche di richiedono che ogni *variazione dei dati confidenziali non sia rivelato attraverso l'osservazione dei dati pubblici.*

- Molti sistemi reali rilasciano *volontariamente* dell'informazione riguardante i dati privati

Indebolire la Non-Interferenza

Le politiche di richiedono che ogni *variazione dei dati confidenziali non sia rivelato attraverso l'osservazione dei dati pubblici.*

- Molti sistemi reali rilasciano *volontariamente* dell'informazione riguardante i dati privati
- Anche se un sistema soddisfa la non-interferenza, alcuni test possono rigettarlo come insicuro

Indebolire la Non-Interferenza

Le politiche di richiedono che ogni *variazione dei dati confidenziali non sia rivelato attraverso l'osservazione dei dati pubblici.*

- Caratterizzazione dell'informazione rilasciata (declassificazione)
- Vincolare gli attaccanti

Indebolire la Non-Interferenza

Le politiche di richiedono che ogni *variazione dei dati confidenziali non sia rivelato attraverso l'osservazione dei dati pubblici.*

- Caratterizzazione dell'informazione rilasciata (declassificazione)
  - Dipendenza selettiva [Cohen'77]: $\mathtt{L}$ è *selettivamente indipendente* da $\mathtt{H}$ in $\mathtt{P}$,rispetto all'asserzione ϕ, i.e., $\mathtt{H} \not\rhd^{\mathtt{P}}_{\phi} \mathtt{L}$, se:

$$\forall s_1, s_2 \,.\, (\phi(s_1^{\mathtt{H}}) \;\wedge\; \phi(s_2^{\mathtt{H}}) \;\wedge\; s_1^{\mathtt{L}} = s_2^{\mathtt{L}}) \;\Rightarrow\; [\![\mathtt{P}]\!](s_1)^{\mathtt{L}} = [\![\mathtt{P}]\!](s_2)^{\mathtt{L}}$$

- Vincolare gli attaccanti

Indebolire la Non-Interferenza

Le politiche di richiedono che ogni *variazione dei dati confidenziali non sia rivelato attraverso l'osservazione dei dati pubblici.*

- Caratterizzazione dell'informazione rilasciata (declassificazione)
  - Dipendenza selettiva [Cohen'77]
  - Declassificazione robusta [Zdancewic & Myers'01]: Si trasforma la capacità osservazione dell'attaccante per determinare cosa il programma rilascia:

$$\forall \sigma, \sigma' \in \Sigma \,.\, \langle \sigma, \sigma' \rangle \in S[\approx] \;\Leftrightarrow\; \mathrm{Obs}_{\sigma}(S, \approx) \equiv \mathrm{Obs}_{\sigma'}(S, \approx)$$

- Vincolare gli attaccanti

Indebolire la Non-Interferenza

Le politiche di richiedono che ogni *variazione dei dati confidenziali non sia rivelato attraverso l'osservazione dei dati pubblici.*

- Caratterizzazione dell'informazione rilasciata (declassificazione)
  - Dipendenza selettiva [Cohen'77]
  - Declassificazione robusta [Zdancewic & Myers'01]
  - Delimited Release [Sabelfeld & Myers'04]: Si verifica se l'informazione declassificata è sufficiente a catturare tutti i flussi:

$$\forall s_1, s_2 \in \Sigma.\ s_1^L = s_2^L \wedge s_1 \approx_E s_2 \ \Rightarrow\ [\![P]\!](s_1)^L = [\![P]\!](s_2)^L$$

- Vincolare gli attaccanti

Indebolire la Non-Interferenza

Le politiche di richiedono che ogni *variazione dei dati confidenziali non sia rivelato attraverso l'osservazione dei dati pubblici.*

- Caratterizzazione dell'informazione rilasciata (declassificazione)
  - Dipendenza selettiva [Cohen'77]
  - Declassificazione robusta [Zdancewic & Myers'01]
  - Delimited Release [Sabelfeld & Myers'04]
  - Non-interferenze rilassata [Li &Zdancewic'05]: Si determinano le politiche di declasificazione necessarie a rendere un λ-termine sicuro:

$$P \equiv f(n_1\sigma_1)(n_2\sigma_2)\ldots(n_k\sigma_k)$$

- Vincolare gli attaccanti

Indebolire la Non-Interferenza

Le politiche di richiedono che ogni *variazione dei dati confidenziali non sia rivelato attraverso l'osservazione dei dati pubblici.*

- Caratterizzazione dell'informazione rilasciata (declassificazione)
  - Dipendenza selettiva [Cohen'77]
  - Declassificazione robusta [Zdancewic & Myers'01]
  - Delimited Release [Sabelfeld & Myers'04]
  - Non-interferenze rilassata [Li &Zdancewic'05]
  - Approcci basati sulla teoria dell'informazione:
    - [Clark et al.'04]: La teoria dell'informaizone di Shannon è usata per quantificare la quantità di informazione che un programma può rilasciare e analizzare in che modo questa dipende dalla distribuzione degli input ;
- Vincolare gli attaccanti

Indebolire la Non-Interferenza

Le politiche di richiedono che ogni *variazione dei dati confidenziali non sia rivelato attraverso l'osservazione dei dati pubblici.*

- Caratterizzazione dell'informazione rilasciata (declassificazione)
  - Dipendenza selettiva [Cohen'77]
  - Declassificazione robusta [Zdancewic & Myers'01]
  - Delimited Release [Sabelfeld & Myers'04]
  - Non-interferenze rilassata [Li &Zdancewic'05]
  - Approcci basati sulla teoria dell'informazione :
    - [Lowe'02]: La *quantità di flusso di informazione* che può effetivamente fluire è misurata in termini del numero di comportamenti a livello privato possono essere accuratamente distinti dall'osservazione pubblica.
- Vincolare gli attaccanti

Indebolire la Non-Interferenza

Le politiche di richiedono che ogni *variazione dei dati confidenziali non sia rivelato attraverso l'osservazione dei dati pubblici.*

- Caratterizzazione dell'informazione rilasciata (declassificazione)
  - Dipendenza selettiva [Cohen'77]
  - Declassificazione robusta [Zdancewic & Myers'01]
  - Delimited Release [Sabelfeld & Myers'04]
  - Non-interferenze rilassata [Li &Zdancewic'05]
  - Approcci basati sulla teoria dell'informazione
- Vincolare gli attaccanti
  - Un approccio probabilistico [Di Pierro et al.'02]: La nozione di indistinguibilità nella non-interferenza viene sostituita dalla nozione di *similarità*.

Indebolire la Non-Interferenza

Le politiche di richiedono che ogni *variazione dei dati confidenziali non sia rivelato attraverso l'osservazione dei dati pubblici.*

- Caratterizzazione dell'informazione rilasciata (declassificazione)
  - Dipendenza selettiva [Cohen'77]
  - Declassificazione robusta [Zdancewic & Myers'01]
  - Delimited Release [Sabelfeld & Myers'04]
  - Non-interferenze rilassata [Li &Zdancewic'05]
  - Approcci basati sulla teoria dell'informazione
- Vincolare gli attaccanti
  - Un approccio probabilistico [Di Pierro et al.'02]
  - Un approccio basato sulla complessità [Laud'01]: vengono considerati solo gli attaccanti che lavorano in tempo probabilistico polinomiale.

Dipendenza selettiva

[Cohen '78]

POTREMMO NON ESSERE INTERESSATI AL FATTO CHE LA VARIABILE b IN OUTPUT RIFLETTA SE LA VARIABILE DI INPUT a SIA PARI O DISPARI. D'ALTRA PARTE POTREMMO VOLER DIMOSTRARE CHE b NON DIPENDE DA a IN NESSUN ALTRO MODO.

- DIPENDENZA FORTE: $\exists s_1, s_2$ such that

$$\forall x \neq a. s_1(x) = s_2(x) \ \wedge \ [\![P]\!](s_1)(b) \neq [\![P]\!](s_2)(b)$$

- DIPENDENZA SELETTIVA: $\exists s_1, s_2$ such that

$$\forall x \neq a. s_1(x) = s_2(x) \wedge \phi(s_1) \ \wedge \ \phi(s_2) \ \wedge \ [\![P]\!](s_1)(b) \neq [\![P]\!](s_2)(b)$$

Dipendenza selettiva

[Cohen '78]

POTREMMO NON ESSERE INTERESSATI AL FATTO CHE LA VARIABILE b IN OUTPUT RIFLETTA SE LA VARIABILE DI INPUT a SIA PARI O DISPARI. D'ALTRA PARTE POTREMMO VOLER DIMOSTRARE CHE b NON DIPENDE DA a IN NESSUN ALTRO MODO.

- DIPENDENZA FORTE: $\exists s_1, s_2$ such that

$$\forall x \neq a. s_1(x) = s_2(x) \ \wedge \ [\![P]\!](s_1)(b) \neq [\![P]\!](s_2)(b)$$

- DIPENDENZA SELETTIVA: $\exists s_1, s_2$ such that

$$\forall x \neq a. s_1(x) = s_2(x) \wedge \phi(s_1) \ \wedge \ \phi(s_2) \ \wedge \ [\![P]\!](s_1)(b) \neq [\![P]\!](s_2)(b)$$

ESEMPIO:

Sia $l := x + (h \ mod \ 4)$ e $\phi = h \ mod \ 4 = 3$

$\Rightarrow \phi$ elimina tutta l'interferenza tra le variabili.

Declassificazione Robusta

[Myers et al. '04]

Linguaggio=IMP+*declassify*(e)+$[\bullet]$

$P[a]$ è il programma P sotto l'attacco di a!

Declassificazione Robusta

[Myers et al. '04]

Linguaggio=IMP+*declassify*(e)+$[\bullet]$

$P[a]$ è il programma P sotto l'attacco di a!

$P[\bullet]$ è ROBUSTO se

$$\forall s_1, s_2 \in \Sigma. \forall a, a' : \ [\![P[a]]\!](s_1)^L = [\![P[a]]\!](s_2)^L \ \Rightarrow \ [\![P[a']]\!](s_1)^L = [\![P[a']]\!](s_2)^L$$

Declassificazione Robusta

[Myers et al. '04]

Linguaggio=IMP+*declassify*(e)+$[\bullet]$

$P[a]$ è il programma P sotto l'attacco di a!

$P[\bullet]$ è ROBUSTO se

$$\forall s_1, s_2 \in \Sigma. \forall a, a' : \ [\![P[a]]\!](s_1)^L = [\![P[a]]\!](s_2)^L \ \Rightarrow \ [\![P[a']]\!](s_1)^L = [\![P[a']]\!](s_2)^L$$

ESEMPIO:

$$P \stackrel{\text{def}}{=} [\bullet]; l := l + \textit{declassify}(h \textit{ mod } 3)$$

P soddisfa la *declassificazione robusta*!

Delimited release

[Sabelfeld & Myers '04]

Linguaggio=IMP+*declassify*(e)

$s_1 \approx_E s_2$ iff $\forall e \in E. \ [\![e]\!](s_1) = [\![e]\!](s_2)$

Delimited release

[Sabelfeld & Myers '04]

Linguaggio=IMP+*declassify*(e)

$s_1 \approx_E s_2$ iff $\forall e \in E.\ [\![e]\!](s_1) = [\![e]\!](s_2)$

$\Downarrow$

P soddisfa DELIMITED RELEASE, $E = \left\{ e \;\middle|\; \textit{declassify}(e) \text{ in } P \right\}$

$$\boxed{\forall s_1, s_2 \in \Sigma.\ s_1^L = s_2^L \wedge s_1 \approx_E s_2 \ \Rightarrow\ [\![P]\!](s_1)^L = [\![P]\!](s_2)^L}$$

Delimited release

[Sabelfeld & Myers '04]

Linguaggio=IMP+*declassify*(e)

$s_1 \approx_E s_2$ iff $\forall e \in E.\ [\![e]\!](s_1) = [\![e]\!](s_2)$

$\Downarrow$

P soddisfa DELIMITED RELEASE, $E = \left\{ e \;\middle|\; \textit{declassify}(e) \text{ in } P \right\}$

$$\boxed{\forall s_1, s_2 \in \Sigma.\ s_1^L = s_2^L \wedge s_1 \approx_E s_2 \ \Rightarrow\ [\![P]\!](s_1)^L = [\![P]\!](s_2)^L}$$

ESEMPIO:

I programmi P_i soddisfano delimited release mentre P_i' no:

$$P_1 \stackrel{\text{def}}{=} avg := \textit{declassify}((h_1 + h_2 + \ldots + h_n)/n);$$

$$P_1' \stackrel{\text{def}}{=} \left[\begin{array}{l} h_1 := h_1; h_2 := h_1; \ldots h_n := h_1; \\ avg := \textit{declassify}((h_1 + h_2 + \ldots + h_n)/n); \end{array} \right.$$

Delimited release

[Sabelfeld & Myers '04]

Linguaggio=IMP+*declassify*(e)

$s_1 \approx_E s_2$ iff $\forall e \in E.\ [\![e]\!](s_1) = [\![e]\!](s_2)$

$\Downarrow$

P soddisfa DELIMITED RELEASE, $E = \left\{ e \;\middle|\; \textit{declassify}(e) \text{ in } P \right\}$

$$\boxed{\forall s_1, s_2 \in \Sigma.\ s_1^L = s_2^L \wedge s_1 \approx_E s_2 \ \Rightarrow\ [\![P]\!](s_1)^L = [\![P]\!](s_2)^L}$$

ESEMPIO:

I programmi P_i soddisfano delimited release mentre P_i' no:

$$P_2 \stackrel{\text{def}}{=} \textbf{if } \textit{declassify}(h \geq k) \textbf{ then } (h := h - k; l := l + k) \textbf{ else nil}$$

$$P_2' \stackrel{\text{def}}{=} \left[\begin{array}{ll} l := 0; & \\ \textbf{while } n \geq 0 \textbf{ do} & k := 2^{n+1} \\ & \textbf{if } \textit{declassify}(h \geq k) \textbf{ then } (h := h - k; l := l + k) \textbf{ else nil} \\ & n := n - 1 \end{array}\right.$$

Delimited release

[Sabelfeld & Myers '04]

Linguaggio=IMP+*declassify*(e)

$s_1 \approx_E s_2$ iff $\forall e \in E.\ [\![e]\!](s_1) = [\![e]\!](s_2)$

$\Downarrow$

P soddisfa DELIMITED RELEASE, $E = \left\{ e \;\middle|\; \textit{declassify}(e) \text{ in } P \right\}$

$$\boxed{\forall s_1, s_2 \in \Sigma.\ s_1^L = s_2^L \wedge s_1 \approx_E s_2 \ \Rightarrow\ [\![P]\!](s_1)^L = [\![P]\!](s_2)^L}$$

ESEMPIO:

I programmi P_i soddisfano delimited release mentre P_i' no:

$$P_3 \stackrel{\text{def}}{=} \textbf{if } \textit{declassify}(h \textit{ mod } 2) \textbf{ then } l := 0; h := 0 \textbf{ else } l := 1; h := 1$$

$$P_3' \stackrel{\text{def}}{=} \left[\begin{array}{l} h := h \textit{ mod } 2; \\ \textbf{if } \textit{declassify}(h = 0) \textbf{ then } l := 0; h := 0 \textbf{ else } l := 1; h := 1 \end{array}\right.$$

Non-interferenza rilassata

[Li & Zdancewic '05]

Linguaggio=λ-calcolo (nessuna declassificazione esplicita)

$\Downarrow$

P soddisfa la NON-INTERFERENZA RILASSATA, se

$$\boxed{P \equiv f(n_1\sigma_1)(n_2\sigma_2)\ldots(n_k\sigma_k)}$$

dove

f è un λ-termine senza variabili segrete

σ_i sono tutte variabili segrete

n_i sono λ-termini che denotano politiche di downgrading tali che $(n_i\sigma_i)$ sono tutti pubblici.

Non-interferenza rilassata

[Li & Zdancewic '05]

Linguaggio=λ-calcolo (nessuna declassificazione esplicita)

$\Downarrow$

P soddisfa la NON-INTERFERENZA RILASSATA, se

$$\boxed{P \equiv f(n_1\sigma_1)(n_2\sigma_2)\ldots(n_k\sigma_k)}$$

ESEMPIO:

$$P_1 \stackrel{\text{def}}{=} \left[\begin{array}{l} x := \mathit{hash}(secret); \\ y := x \ \mathit{mod}\ 2^{64}; \\ \textbf{if } y = input \textbf{ then } output = 1 \textbf{ else } output := 0; \end{array}\right.$$

Il corrispondente λ-programma soddisfa la *non-interferenza rilassata* con politica di downgrading:

$$\lambda secret.\, \lambda input.\, (\mathit{hash}(secret) \ \mathit{mod}\ 2^{64}) = input$$

Non-interferenza rilassata

[Li & Zdancewic '05]

Linguaggio=λ-calcolo (nessuna declassificazione esplicita)

$\Downarrow$

P soddisfa la NON-INTERFERENZA RILASSATA, se

$$\boxed{P \equiv f(n_1\sigma_1)(n_2\sigma_2)\ldots(n_k\sigma_k)}$$

ESEMPIO: Controllo password:

$$P = \lambda in.\textbf{if } in = \sigma_{pw} \textbf{ then } out := 1 \textbf{ else } out := 0$$

soddisfa la *non-interferenza rilassata* potendo essere trasformato in:

$$\lambda x.\lambda g : \mathbb{Z} \longrightarrow \mathbb{Z}.(\textbf{if } g(x) \textbf{ then } out := 1 \textbf{ else } out := 0)in((\lambda x.\lambda y.x = y)\sigma_{pw})$$

Cosa fare?

- Abbandonare la nozione di non-interferenza con attaccanti *all-powerful* $\Rightarrow$ Indebolire la non-interferenza
- Combinare indebolimento attaccanti con caratterizzazione informazione rilasciata