# A Sound Type System for Secure Flow Analysis

Dennis Volpana, Geoffrey Smith, Cynthia Irvine

Jason Belt
CIS 890

09.20.2010

# Secure Information Flow Analysis

- Static analysis is used to ensure sensitive information is not leaked
- Define a lattice of security levels and prove information only flows upwards

  e.g. $\text{if } L \leq H \text{ then } L \leadsto L,\ H \leadsto H,\ L \leadsto H,\ H \not\leadsto L$

# Secure Information Flow Analysis

- Static analysis is used to ensure sensitive information is not leaked
- Define a lattice of security levels and prove information only flows upwards

  **e.g.** if $L \leq H$ then $L \rightsquigarrow L$, $H \rightsquigarrow H$, $L \rightsquigarrow H$, $H \not\rightsquigarrow L$

**Definition (Noninterference)** Program $c$ satisifies noninterference if, for any memories $\mu$ and $v$ that agree on $L$ variables, the memories produced by running $c$ on $\mu$ and on $v$ also agree on $L$ variables (provided both runs terminate successfully)

# Type-Based Approach

- Security levels ≈ Types
- Lattice order on security levels ≈ Subtyping
- Program certification ≈ Type checking

# Type-Based Approach

- Security levels ≈ Types
- Lattice order on security levels ≈ Subtyping
- Program certification ≈ Type checking

welltyped($P$) ⇒ noninterference($P$)

# The Core Language

$$
\begin{array}{rccl}
\text{Phrases} & p & ::= & e \mid c \\
\text{Expressions} & e & ::= & x \mid l \mid n \mid e + e' \mid e - e' \mid \\
 & & & e = e' \mid e < e' \\
\text{Commands} & c & ::= & e := e' \mid c; c' \mid \textbf{if } e \textbf{ then } c \textbf{ else } c' \mid \\
 & & & \textbf{while } e \textbf{ do } c \mid \textbf{letvar } x := e \textbf{ in } c
\end{array}
$$

# The Core Language

$$
\begin{array}{rccl}
\text{Phrases} & p & ::= & e \mid c \\
\text{Expressions} & e & ::= & x \mid l \mid n \mid e + e' \mid e - e' \mid \\
 & & & e = e' \mid e < e' \\
\text{Commands} & c & ::= & e := e' \mid c; c' \mid \textbf{if } e \textbf{ then } c \textbf{ else } c' \mid \\
 & & & \textbf{while } e \textbf{ do } c \mid \textbf{letvar } x := e \textbf{ in } c \\
\text{Security classes} & s & \in & SC \text{ (partially ordered by } \leq) \\
\text{Type} & \tau & ::= & s \\
\text{Phrase types} & \rho & ::= & \tau \mid \tau\ var \mid \tau\ cmd
\end{array}
$$

# Typing Judgements

$$\lambda; \gamma \vdash p : \rho$$

# Typing Judgements

$$\lambda; \gamma \vdash p : \rho$$

- $\lambda : l \to \tau$ location typing

# Typing Judgements

$$\lambda; \gamma \vdash p : \rho$$

- $\lambda : l \to \tau$ location typing
- $\gamma : x \to \rho$ identifier typing

# Typing Rules

(INT) $\lambda; \gamma \vdash n : \tau$

(VAR) $\lambda; \gamma \vdash x : \tau\ var \quad \text{if } \gamma(x) = \tau\ var$

(VARLOC) $\lambda; \gamma \vdash l : \tau\ var \quad \text{if } \lambda(l) = \tau$

(ARITH)
$$\frac{\lambda; \gamma \vdash e : \tau, \quad \lambda; \gamma \vdash e' : \tau}{\lambda; \gamma \vdash e + e' : \tau}$$

(R-VAL)
$$\frac{\lambda; \gamma \vdash e : \tau\ var}{\lambda; \gamma \vdash e : \tau}$$

(ASSIGN)
$$\frac{\lambda; \gamma \vdash e : \tau\ var, \quad \lambda; \gamma \vdash e' : \tau}{\lambda; \gamma \vdash e := e' : \tau\ cmd}$$

# Typing Rules

(INT) $\quad \lambda;\gamma \vdash n : \tau$

(VAR) $\quad \lambda;\gamma \vdash x : \tau\ var \quad \text{if } \gamma(x) = \tau\ var$

(VARLOC) $\quad \lambda;\gamma \vdash l : \tau\ var \quad \text{if } \lambda(l) = \tau$

(ARITH)
$$\frac{\lambda;\gamma \vdash e : \tau, \quad \lambda;\gamma \vdash e' : \tau}{\lambda;\gamma \vdash e + e' : \tau}$$

Upward flow from $e$ to $e'$ allowed if $e : H$, $e' : L$, and $e'$ can be coerced to $H$, then with the rule applied with $\tau = H$

(R-VAL)
$$\frac{\lambda;\gamma \vdash e : \tau\ var}{\lambda;\gamma \vdash e : \tau}$$

(ASSIGN)
$$\frac{\lambda;\gamma \vdash e : \tau\ var, \quad \lambda;\gamma \vdash e' : \tau}{\lambda;\gamma \vdash e := e' : \tau\ cmd}$$

# Typing Rules

$$(\textsc{compose})\quad \frac{\lambda;\gamma \vdash c : \tau\ cmd,\quad \lambda;\gamma \vdash c' : \tau\ cmd}{\lambda;\gamma \vdash c;\, c' : \tau\ cmd}$$

$$(\textsc{if})\quad \frac{\lambda;\gamma \vdash e : \tau,\quad \lambda;\gamma \vdash c : \tau\ cmd,\quad \lambda;\gamma \vdash c' : \tau\ cmd}{\lambda;\gamma \vdash \textbf{if}\ e\ \textbf{then}\ c\ \textbf{else}\ c' : \tau\ cmd}$$

$$(\textsc{while})\quad \frac{\lambda;\gamma \vdash e : \tau,\quad \lambda;\gamma \vdash c : \tau\ cmd}{\lambda;\gamma \vdash \textbf{while}\ e\ \textbf{do}\ c : \tau\ cmd}$$

$$(\textsc{letvar})\quad \frac{\lambda;\gamma \vdash e : \tau,\quad \lambda;\gamma[x : \tau\ var] \vdash c : \tau'\ cmd}{\lambda;\gamma \vdash \textbf{letvar}\ x := e\ \textbf{in}\ c : \tau'\ cmd}$$

# Typing Rules

(COMPOSE)
$$\frac{\lambda; \gamma \vdash c : \tau\ cmd, \quad \lambda; \gamma \vdash c' : \tau\ cmd}{\lambda; \gamma \vdash c; c' : \tau\ cmd}$$

(IF)
$$\frac{\lambda; \gamma \vdash e : \tau, \quad \lambda; \gamma \vdash c : \tau\ cmd, \quad \lambda; \gamma \vdash c' : \tau\ cmd}{\lambda; \gamma \vdash \textbf{if}\ e\ \textbf{then}\ c\ \textbf{else}\ c' : \tau\ cmd}$$

Suppose $\gamma(x) = \gamma(y) = H\ var$ and $\tau = H$

(WHILE)
$$\frac{\lambda; \gamma \vdash e : \tau, \quad \lambda; \gamma \vdash c : \tau\ cmd}{\lambda; \gamma \vdash \textbf{while}\ e\ \textbf{do}\ c : \tau\ cmd}$$

(LETVAR)
$$\frac{\lambda; \gamma \vdash e : \tau, \quad \lambda; \gamma[x : \tau\ var] \vdash c : \tau'\ cmd}{\lambda; \gamma \vdash \textbf{letvar}\ x := e\ \textbf{in}\ c : \tau'\ cmd}$$

# Typing Rules

(COMPOSE)
$$\frac{\lambda; \gamma \vdash c : \tau\ cmd, \quad \lambda; \gamma \vdash c' : \tau\ cmd}{\lambda; \gamma \vdash c; c' : \tau\ cmd}$$

(IF)
$$\frac{\lambda; \gamma \vdash e : \tau, \quad \lambda; \gamma \vdash c : \tau\ cmd, \quad \lambda; \gamma \vdash c' : \tau\ cmd}{\lambda; \gamma \vdash \textbf{if}\ e\ \textbf{then}\ c\ \textbf{else}\ c' : \tau\ cmd}$$

(WHILE)
$$\frac{\lambda; \gamma \vdash e : \tau, \quad \lambda; \gamma \vdash c : \tau\ cmd}{\lambda; \gamma \vdash \textbf{while}\ e\ \textbf{do}\ c : \tau\ cmd}$$

Suppose $\gamma(x) = \gamma(y) = H\ var$ and $\tau = H$

$$\gamma \vdash \textbf{if}\ x = 1\ \textbf{then}\ y := 1\ \textbf{else}\ y := 0$$

(LETVAR)
$$\frac{\lambda; \gamma \vdash e : \tau, \quad \lambda; \gamma[x : \tau\ var] \vdash c : \tau'\ cmd}{\lambda; \gamma \vdash \textbf{letvar}\ x := e\ \textbf{in}\ c : \tau'\ cmd}$$

# Typing Rules

(COMPOSE)
$$\frac{\lambda;\gamma \vdash c : \tau\ cmd, \quad \lambda;\gamma \vdash c' : \tau\ cmd}{\lambda;\gamma \vdash c;\, c' : \tau\ cmd}$$

(IF)
$$\frac{\lambda;\gamma \vdash e : \tau, \quad \lambda;\gamma \vdash c : \tau\ cmd, \quad \lambda;\gamma \vdash c' : \tau\ cmd}{\lambda;\gamma \vdash \textbf{if}\ e\ \textbf{then}\ c\ \textbf{else}\ c' : \tau\ cmd}$$

(WHILE)
$$\frac{\lambda;\gamma \vdash e : \tau, \quad \lambda;\gamma \vdash c : \tau\ cmd}{\lambda;\gamma \vdash \textbf{while}\ e\ \textbf{do}\ c : \tau\ cmd}$$

(LETVAR)
$$\frac{\lambda;\gamma \vdash e : \tau, \quad \lambda;\gamma[x : \tau\ var] \vdash c : \tau'\ cmd}{\lambda;\gamma \vdash \textbf{letvar}\ x := e\ \textbf{in}\ c : \tau'\ cmd}$$

Suppose $\gamma(x) = \gamma(y) = H\ var$ and $\tau = H$

$$\gamma \vdash \textbf{if}\ \underbrace{x = 1}_{H}\ \textbf{then}\ \underbrace{y := 1}_{H\ cmd}\ \textbf{else}\ \underbrace{y := 0}_{H\ cmd}$$

# Typing Rules

(COMPOSE)

$$\frac{\lambda; \gamma \vdash c : \tau\ cmd, \quad \lambda; \gamma \vdash c' : \tau\ cmd}{\lambda; \gamma \vdash c;\ c' : \tau\ cmd}$$

(IF)

$$\frac{\lambda; \gamma \vdash e : \tau, \quad \lambda; \gamma \vdash c : \tau\ cmd, \quad \lambda; \gamma \vdash c' : \tau\ cmd}{\lambda; \gamma \vdash \textbf{if}\ e\ \textbf{then}\ c\ \textbf{else}\ c' : \tau\ cmd}$$

Suppose $\gamma(x) = \gamma(y) = H\ var$ and $\tau = H$

$$\overset{H}{\ } \qquad \overset{H\ cmd}{\ } \qquad \overset{H\ cmd}{\ }$$

$$\gamma \vdash \textbf{if}\ x = 1\ \textbf{then}\ y := 1\ \textbf{else}\ y := 0 : H\ cmd$$

# Typing Rules

(COMPOSE)
$$\frac{\lambda;\gamma \vdash c : \tau\ cmd, \quad \lambda;\gamma \vdash c' : \tau\ cmd}{\lambda;\gamma \vdash c; c' : \tau\ cmd}$$

(IF)
$$\frac{\lambda;\gamma \vdash e : \tau, \quad \lambda;\gamma \vdash c : \tau\ cmd, \quad \lambda;\gamma \vdash c' : \tau\ cmd}{\lambda;\gamma \vdash \textbf{if}\ e\ \textbf{then}\ c\ \textbf{else}\ c' : \tau\ cmd}$$

Suppose $\gamma(x) = L\ var$, $\gamma(y) = H\ var$, $\tau = L$, and $L \leq H$ so that $H\ cmd \subseteq L\ cmd$

$$\gamma \vdash \textbf{if}\ x = 1\ \textbf{then}\ y := 1\ \textbf{else}\ y := 0$$

# Typing Rules

(COMPOSE)
$$\frac{\lambda;\gamma \vdash c : \tau\ cmd,\quad \lambda;\gamma \vdash c' : \tau\ cmd}{\lambda;\gamma \vdash c;\, c' : \tau\ cmd}$$

(IF)
$$\frac{\lambda;\gamma \vdash e : \tau,\quad \lambda;\gamma \vdash c : \tau\ cmd,\quad \lambda;\gamma \vdash c' : \tau\ cmd}{\lambda;\gamma \vdash \mathbf{if}\ e\ \mathbf{then}\ c\ \mathbf{else}\ c' : \tau\ cmd}$$

(WHILE)
$$\frac{\lambda;\gamma \vdash e : \tau,\quad \lambda;\gamma \vdash c : \tau\ cmd}{\lambda;\gamma \vdash \mathbf{while}\ e\ \mathbf{do}\ c : \tau\ cmd}$$

(LETVAR)
$$\frac{\lambda;\gamma \vdash e : \tau,\quad \lambda;\gamma[x : \tau\ var] \vdash c : \tau'\ cmd}{\lambda;\gamma \vdash \mathbf{letvar}\ x := e\ \mathbf{in}\ c : \tau'\ cmd}$$

Suppose $\gamma(x) = L\ var$, $\gamma(y) = H\ var$, $\tau = L$, and $L \leq H$ so that $H\ cmd \subseteq L\ cmd$

$$\gamma \vdash \mathbf{if}\ \overset{L}{x = 1}\ \mathbf{then}\ y := 1\ \mathbf{else}\ y := 0$$

# Typing Rules

(COMPOSE)
$$\frac{\lambda;\gamma \vdash c : \tau\ cmd,\quad \lambda;\gamma \vdash c' : \tau\ cmd}{\lambda;\gamma \vdash c;\, c' : \tau\ cmd}$$

(IF)
$$\frac{\lambda;\gamma \vdash e : \tau,\quad \lambda;\gamma \vdash c : \tau\ cmd,\quad \lambda;\gamma \vdash c' : \tau\ cmd}{\lambda;\gamma \vdash \textbf{if}\ e\ \textbf{then}\ c\ \textbf{else}\ c' : \tau\ cmd}$$

(WHILE)
$$\frac{\lambda;\gamma \vdash e : \tau,\quad \lambda;\gamma \vdash c : \tau\ cmd}{\lambda;\gamma \vdash \textbf{while}\ e\ \textbf{do}\ c : \tau\ cmd}$$

(LETVAR)
$$\frac{\lambda;\gamma \vdash e : \tau,\quad \lambda;\gamma[x : \tau\ var] \vdash c : \tau'\ cmd}{\lambda;\gamma \vdash \textbf{letvar}\ x := e\ \textbf{in}\ c : \tau'\ cmd}$$

Suppose $\gamma(x) = L\ var$, $\gamma(y) = H\ var$, $\tau = L$, and $L \leq H$ so that $H\ cmd \subseteq L\ cmd$

$$\gamma \vdash \textbf{if}\ \underset{L}{x = 1}\ \textbf{then}\ \underset{H\ cmd \rightsquigarrow L\ cmd}{y := 1\ \textbf{else}\ y := 0}$$

# Typing Rules

(COMPOSE)
$$\frac{\lambda; \gamma \vdash c : \tau\ cmd, \quad \lambda; \gamma \vdash c' : \tau\ cmd}{\lambda; \gamma \vdash c; c' : \tau\ cmd}$$

(IF)
$$\frac{\lambda; \gamma \vdash e : \tau, \quad \lambda; \gamma \vdash c : \tau\ cmd, \quad \lambda; \gamma \vdash c' : \tau\ cmd}{\lambda; \gamma \vdash \textbf{if } e \textbf{ then } c \textbf{ else } c' : \tau\ cmd}$$

(WHILE)
$$\frac{\lambda; \gamma \vdash e : \tau, \quad \ldots}{\lambda; \gamma \vdash \textbf{while } e \textbf{ do } c : \tau\ cmd}$$

(LETVAR)
$$\frac{\ldots}{\lambda; \gamma \vdash \textbf{letvar } x := e \textbf{ in } c : \tau'\ cmd}$$

Suppose $\gamma(x) = L\ var$, $\gamma(y) = H\ var$, $\tau = L$, and $L \leq H$ so that $H\ cmd \subseteq L\ cmd$

$$\gamma \vdash \textbf{if } \overset{L}{x = 1} \textbf{ then } \overset{L\ cmd}{y := 1} \textbf{ else } \overset{L\ cmd}{y := 0} : L\ cmd$$

# Typing Rules

$$(\textsc{compose})\quad \frac{\lambda;\gamma \vdash c : \tau\ cmd,\quad \lambda;\gamma \vdash c' : \tau\ cmd}{\lambda;\gamma \vdash c;\, c' : \tau\ cmd}$$

$$(\textsc{if})\quad \frac{\lambda;\gamma \vdash e : \tau,\quad \lambda;\gamma \vdash c : \tau\ cmd,\quad \lambda;\gamma \vdash c' : \tau\ cmd}{\lambda;\gamma \vdash \mathbf{if}\ e\ \mathbf{then}\ c\ \mathbf{else}\ c' : \tau\ cmd}$$

$$(\textsc{while})\quad \frac{\lambda;\gamma \vdash e : \tau,\quad \lambda;\gamma \vdash c : \tau\ cmd}{\lambda;\gamma \vdash \mathbf{while}\ e\ \mathbf{do}\ c : \tau\ cmd}$$

$$(\textsc{letvar})\quad \frac{\lambda;\gamma \vdash e : \tau,\quad \lambda;\gamma[x : \tau\ var] \vdash c : \tau'\ cmd}{\lambda;\gamma \vdash \mathbf{letvar}\ x := e\ \mathbf{in}\ c : \tau'\ cmd}$$

# Subtyping Rules

$$(\textsc{base}) \quad \frac{\tau \leq \tau'}{\vdash \tau \subseteq \tau'}$$

$$(\textsc{reflex}) \quad \vdash \rho \subseteq \rho$$

$$(\textsc{trans}) \quad \frac{\vdash \rho \subseteq \rho', \ \vdash \rho' \subseteq \rho''}{\vdash \rho \subseteq \rho''}$$

$$(\textsc{cmd}^-) \quad \frac{\vdash \tau \subseteq \tau'}{\vdash \tau' \ cmd \subseteq \tau \ cmd}$$

$$(\textsc{subtype}) \quad \frac{\lambda; \gamma \vdash p : \rho, \quad \vdash \rho \subseteq \rho'}{\lambda; \gamma \vdash p : \rho'}$$

# Operational Semantics

- Evaluation is performed relative to a memory $\mu : l \rightarrow n$

# Operational Semantics

- Evaluation is performed relative to a memory $\mu : l \to n$

$$\mu \vdash e \Rightarrow n \qquad \mu \vdash c \Rightarrow \mu'$$

# Operational Semantics

$$(\textsc{base}) \qquad \mu \vdash n \Rightarrow n$$

$$(\textsc{contents}) \qquad \mu \vdash l \Rightarrow \mu(l) \qquad \text{if } l \in dom(\mu)$$

$$(\textsc{add}) \qquad \frac{\mu \vdash e \Rightarrow n, \quad \mu \vdash e' \Rightarrow n'}{\mu \vdash e + e' \Rightarrow n + n'}$$

$$(\textsc{update}) \qquad \frac{\mu \vdash e \Rightarrow n, \quad l \in dom(\mu)}{\mu \vdash l := e \Rightarrow \mu[l := n]}$$

$$(\textsc{sequence}) \qquad \frac{\mu \vdash c \Rightarrow \mu', \quad \mu' \vdash c' \Rightarrow \mu''}{\mu \vdash c; \, c' \Rightarrow \mu''}$$

$$(\textsc{branch}) \qquad \frac{\mu \vdash e \Rightarrow 1, \quad \mu \vdash c \Rightarrow \mu'}{\mu \vdash \textbf{if } e \textbf{ then } c \textbf{ else } c' \Rightarrow \mu'}$$

$$\frac{\mu \vdash e \Rightarrow 0, \quad \mu \vdash c' \Rightarrow \mu'}{\mu \vdash \textbf{if } e \textbf{ then } c \textbf{ else } c' \Rightarrow \mu'}$$

# Operational Semantics

$$(\textsc{loop}) \quad \frac{\mu \vdash e \Rightarrow 0}{\mu \vdash \textbf{while } e \textbf{ do } c \Rightarrow \mu}$$

$$\frac{\mu \vdash e \Rightarrow 1, \quad \mu \vdash c \Rightarrow \mu', \quad \mu' \vdash \textbf{while } e \textbf{ do } c \Rightarrow \mu''}{\mu \vdash \textbf{while } e \textbf{ do } c \Rightarrow \mu''}$$

$$(\textsc{bindvar}) \quad \frac{\mu \vdash e \Rightarrow n, \quad l \text{ is the first location not in } dom(\mu), \quad \mu[l := n] \vdash [l/x]c \Rightarrow \mu'}{\mu \vdash \textbf{letvar } x := e \textbf{ in } c \Rightarrow \mu' - l}$$

# Type Soundness

- Altering the initial values of locations of type $\tau$ cannot affect the initial values of any locations of type $\tau'$, provided that $\tau \not\leq \tau'$

# Simple Security

**Lemma 6.3** If $\lambda \vdash e : \tau$, then for every $l$ in $e$, $\lambda(l) \vdash \tau$

- Secrecy

  Only locations at level $\tau$ or lower will have their contents read when $e$ is evaluated (no read up)

- Confinement

  If $e$ has integrity level $\tau$, then every location in $e$ stores information at integrity level $\tau$

# Confinement

**Lemma 6.4** If $\lambda; \gamma \vdash c : \tau\ cmd$, then for every $l$ assigned to in $c$, $\lambda(l) \geq \tau$

- Secrecy

  No location below level $\tau$ is updated in $c$
  (no write down)

- Confinement

  Every location assigned to in $c$ can be updated by information at integrity level $\tau$

# Type Soundness

Theorem 6.8 (*Type Soundess*) Suppose

(a) $\lambda \vdash c : \rho$, $\quad$ *c is well typed*

# Type Soundness

Theorem 6.8 (*Type Soundess*) Suppose

| | | |
|---|---|---|
| (a) | $\lambda \vdash c : \rho,$ | *c is well typed* |
| (b) | $\mu \vdash c \Rightarrow \mu',$ | *execution one* |

# Type Soundness

Theorem 6.8 (*Type Soundess*) Suppose

| | | |
|---|---|---|
| (a) | $\lambda \vdash c : \rho,$ | *c is well typed* |
| (b) | $\mu \vdash c \Rightarrow \mu',$ | *execution one* |
| (c) | $v \vdash c \Rightarrow v',$ | *execution two* |

# Type Soundness

Theorem 6.8 (*Type Soundess*) Suppose

| | | |
|---|---|---|
| (a) | $\lambda \vdash c : \rho,$ | *c is well typed* |
| (b) | $\mu \vdash c \Rightarrow \mu',$ | *execution one* |
| (c) | $v \vdash c \Rightarrow v',$ | *execution two* |
| (d) | $\mathsf{dom}(\mu) = \mathsf{dom}(v) = \mathsf{dom}(\lambda)$, and | |

# Type Soundness

Theorem 6.8 (*Type Soundess*) Suppose

| | | |
|---|---|---|
| (a) | $\lambda \vdash c : \rho$, | *c is well typed* |
| (b) | $\mu \vdash c \Rightarrow \mu'$, | *execution one* |
| (c) | $v \vdash c \Rightarrow v'$, | *execution two* |
| (d) | $\mathsf{dom}(\mu) = \mathsf{dom}(v) = \mathsf{dom}(\lambda)$, and | |
| (e) | $v(l) = \mu(l)$ for all $l$ such that $\lambda(l) \leq \tau$ | *the same low inputs* |

# Type Soundness

Theorem 6.8 (*Type Soundess*) Suppose

| | | |
|---|---|---|
| (a) | $\lambda \vdash c : \rho$, | *c is well typed* |
| (b) | $\mu \vdash c \Rightarrow \mu'$, | *execution one* |
| (c) | $v \vdash c \Rightarrow v'$, | *execution two* |
| (d) | $\mathsf{dom}(\mu) = \mathsf{dom}(v) = \mathsf{dom}(\lambda)$, and | |
| (e) | $v(l) = \mu(l)$ for all $l$ such that $\lambda(l) \leq \tau$ | *the same low inputs* |

Then $v'(l) = \mu'(l)$ for all $l$ such that $\lambda(l) \leq \tau$ *the same low outputs*