

End-to-End Availability Policies and Noninterference

Lantian Zheng and Andrew C. Myers

presented by Michael Carl Tschantz

Overview

- Confidentiality – the values of high inputs should not affect the low outputs
- Integrity – the values of low inputs should not affect not affect high outputs
- Availability – low-level inputs should not result in high outputs being unavailable
- Want to statically ensure all three

Outline

- Define a programming language, security levels, and an attacker model
- Express confidentiality, integrity, and availability as noninterference properties
- Provide a type system to infer if these properties hold of a program for assignment of security levels to inputs and outputs

Aimp

Aimp Syntax

Values	v	::=	$n \mid \text{none}$
Expressions	e	::=	$n \mid !m \mid e_1 + e_2$
Statements	s	::=	$\text{skip} \mid m := e \mid s_1; s_2$ \mid $\text{if } e \text{ then } s_1 \text{ else } s_2$ \mid $\text{while } e \text{ do } s$

- none is a value but not an expression, represents unavailable data

Aimp Semantics for Expressions

$$\frac{m \in \text{dom}(M)}{\langle !m, M \rangle \Downarrow M(m)}$$

$$\frac{\langle e_1, M \rangle \Downarrow v_1 \quad \langle e_2, M \rangle \Downarrow v_2 \quad v = v_1 + v_2}{\langle e_1 + e_2, M \rangle \Downarrow v}$$

- where v_1+v_2 is `none` if $v_1=\text{none}$ or $v_2=\text{none}$
and n_1+n_2 if $v_1=n_1$ and $v_2=n_2$

Aimp Semantics for Statements

$$\frac{\langle e, M \rangle \Downarrow n}{\langle m := e, M \rangle \longmapsto \langle \text{skip}, M[m \mapsto n] \rangle}$$

$$\frac{\langle s_1, M \rangle \longmapsto \langle s'_1, M' \rangle}{\langle s_1; s_2, M \rangle \longmapsto \langle s'_1; s_2, M' \rangle}$$

$$\langle \text{skip}; s, M \rangle \longmapsto \langle s, M \rangle$$

- assignment will get stuck if e reduces to none

Aimp Semantics for Statements

$$\frac{\langle e, M \rangle \Downarrow n \quad n > 0}{\langle \text{if } e \text{ then } s_1 \text{ else } s_2, M \rangle \mapsto \langle s_1, M \rangle}$$

$$\frac{\langle e, M \rangle \Downarrow n \quad n \leq 0}{\langle \text{if } e \text{ then } s_1 \text{ else } s_2, M \rangle \mapsto \langle s_2, M \rangle}$$

$$\langle \text{while } e \text{ do } s, M \rangle \mapsto \langle \text{if } e \text{ then } s; \text{while } e \text{ do } s \text{ else skip}, M \rangle$$

- if statements will get stuck if e reduces to none

I/O in Aimp

- Aimp has no read or print statements
- The initial value of the memory may be affected by users to simulate input
- Users may observe the memory as the computation progresses to simulate output

I/O in Aimp

- An input will be none if a user did not supply an input
- Users cannot distinguish none from any other value
- Let $v_1 \approx v_2$ iff $v_1=v_2$ or $v_1=\text{none}$ or $v_2=\text{none}$

Security Levels

Contexts

- Which memory references are for high-confidentiality data; which, for low-confidentiality?
- Let Γ assigns memory references a label
- A label ℓ is $(\beta_C, \beta_I, \beta_A)$
 - $C((\beta_C, \beta_I, \beta_A)) = \beta_C$ confidentiality level of m
 - $I((\beta_C, \beta_I, \beta_A)) = \beta_I$ integrity level of m
 - $A((\beta_C, \beta_I, \beta_A)) = \beta_A$ availability level of m

Level Lattice: Confidentiality

- $\beta_C \leq \beta'_C$ Information of level β_C is less or equally confidential as that of β'_C
- Only if e_1+e_2 is kept confidential will e_1 and e_2 remain confidential
- $C(e_1) \leq C(e_1+e_2)$ and $C(e_2) \leq C(e_1+e_2)$
- $C(e_1+e_2) = C(e_1) \sqcup C(e_2)$

Level Lattice: Integrity

- $\beta_1 \leq \beta'_1$ Information of level β_1 has less or equal integrity as that of β'_1
- $e_1 + e_2$ only has integrity if both e_1 and e_2 has integrity
- $I(e_1 + e_2) \leq I(e_1)$ and $I(e_1 + e_2) \leq I(e_2)$
- $I(e_1 + e_2) = I(e_1) \wedge I(e_2)$

Level Lattice: Availability

- $\beta_A \leq \beta'_A$ Information of level β_A has less or equal availability as that of β'_A
- e_1+e_2 is only available if e_1 and e_2 are available
- $A(e_1+e_2) \leq A(e_1)$ and $A(e_1+e_2) \leq A(e_2)$
- $A(e_1+e_2) = A(e_1) \sqcap A(e_2)$

Label Lattice

- Let $(\beta_C, \beta_I, \beta_A) \sqcup (\beta'_C, \beta'_I, \beta'_A)$ be
 $(\beta_C \sqcup \beta'_C, \beta_I \sqcup \beta'_I, \beta_A \sqcup \beta'_A)$
- Let $(\beta_C, \beta_I, \beta_A) \sqcap (\beta'_C, \beta'_I, \beta'_A)$ be
 $(\beta_C \sqcap \beta'_C, \beta_I \sqcap \beta'_I, \beta_A \sqcap \beta'_A)$

Attacker Model

Attacker Model: Abilities

- Interested in protecting information above a level L
- Presume that the attacker has control over level L and below
 - outputs to a level below L have no confidentiality
 - inputs of a level below L have no integrity
 - inputs of a level below L might not be available

Attacker Model: Limitations

- The confidentiality of other outputs is not compromised
- The integrity and availability of other inputs is not compromised
- The attacker may only compromise the integrity or availability of an output by affecting the integrity or availability of compromised inputs
- No attacks based on timing, termination, etc.

Confidentiality Noninterference

Intuition

- The program given two memories with equivalent low-confidentiality inputs, should produce equivalent low-confidentiality outputs

Confidentiality Indistinguishable Memories

- Memories M_1 and M_2 are indistinguishable at or below confidentiality-level L iff for every reference m ,

$$C(\Gamma(m)) \leq L \text{ implies } M_1(m) \approx M_2(m)$$

- We write $\Gamma \vdash M_1 \approx_{C \leq L} M_2$

Indistinguishable Traces

- An execution yields a sequence of memories [M1, M2, M3, ...] we call a trace
- No timing: [M1, M1, M2] looks like [M1, M2, M2]
- $T1 \approx T2$ iff T1 and T2 are equal modulo stuttering
- T1 looks like T2 if T1 is a prefix of T2:
Observers cannot be sure T1 won't become T2

Confidentiality Indistinguishable Traces

- Traces T_1 and T_2 are indistinguishable at or below confidentiality-level L iff there exists $T'_1 = [M_1, \dots, M_n]$ and $T'_2 = [M'_1, \dots, M'_m]$ s.t.
 - $T'_1 \approx T_1$,
 - $T'_2 \approx T_2$, and
 - $\Gamma \vdash M_i \approx_{C \leq L} M'_i$ for all i in $\{1, \dots, \min(n, m)\}$
- We write $\Gamma \vdash T_1 \approx_{C \leq L} T_2$

Confidentiality Noninterference

- A program s has confidentiality noninterference w.r.t Γ iff every two traces T_1 and T_2 generated by (s, M_1) and (s, M_2) where $\Gamma \Vdash M_1 \approx_{C \leq L} M_2$, we have $\Gamma \Vdash T_1 \approx_{C \leq L} T_2$

Integrity Noninterference

Intuition

- The program given two memories with equivalent high-integrity inputs, should produce equivalent high-integrity outputs

Integrity Indistinguishable Memories

- Memories M_1 and M_2 are indistinguishable above integrity-level L iff for every reference m ,
 $I(\Gamma(m)) > L$ implies $M_1(m) \approx M_2(m)$
- We write $\Gamma \vdash M_1 \approx_{I>L} M_2$

Integrity Indistinguishable Traces

- Traces T_1 and T_2 are indistinguishable above integrity-level L iff there exists $T'_1 = [M_1, \dots, M_n]$ and $T'_2 = [M'_1, \dots, M'_m]$ s.t.
 - $T'_1 \approx T_1$,
 - $T'_2 \approx T_2$, and
 - $\Gamma \vdash M_i \approx_{|>L} M'_i$ for all i in $\{1, \dots, \min(n, m)\}$
- We write $\Gamma \vdash T_1 \approx_{|>L} T_2$

Integrity Noninterference

- A program s has integrity noninterference w.r.t Γ iff every two traces T_1 and T_2 generated by (s, M_1) and (s, M_2) where $\Gamma \vdash M_1 \approx_{I>L} M_2$, we have $\Gamma \vdash T_1 \approx_{I>L} T_2$

Availability Noninterference

Example

$$m_2 := !m_1; \quad m_o := 1;$$

- If m_1 is not available, then $m_2 := !m_1$ will get stuck making m_o unavailable
- Thus, we need the availability of m_1 to be at least as high as that of m_o

First Attempt

- If the high-availability inputs are available, then the high-availability outputs will become available

First Attempt

```
while (! $m_1$ ) do skip;       $m_o := 1$ ;
```

```
if (! $m_1$ ) then while (1) do skip; else skip;  
 $m_o := 1$ ;
```

- The availability of m_o depends not only on the availability of m_1 but also the integrity of m_1

Second Attempt

- If the high-availability inputs are available and the high-integrity references un-compromised, then the high-availability outputs will become available

Second Attempt

- If the high-availability inputs are available and the high-integrity references un-compromised, then the high-availability outputs will become available
- But this requires termination checking

Intuition

- “With all high-availability inputs available, equivalent high-integrity inputs will eventually result in equally available high-availability outputs.”

Availability Indistinguishable Memories

- Memories M_1 and M_2 are indistinguishable above availability-level L iff for every reference m ,
 $A(\Gamma(m)) > L$ implies
 $M_1(m)=\text{none}$ iff $M_2(m)=\text{none}$
- We write $\Gamma \vdash M_1 \approx_{A>L} M_2$

Unassigned References

- Must make clear which references are for output and still need to be assigned
- Let R denote the set of unassigned output references
- All high-availability inputs are available:
for all m , $A(\Gamma(m)) > L$ and $m \notin R$,
implies $M(m) \neq \text{none}$

Availability Noninterference

- s has A.N. w.r.t. Γ and R if for all M_1 and M_2 :
 - $\Gamma \vdash M_1 \approx_{I>L} M_2$
 - All high-availability inputs are available in M_1 & M_2
 - $(s, M_1) \rightarrow (s'_1, M'_1)$ and $(s, M_2) \rightarrow (s'_2, M'_2)$
- then exists (s''_1, M''_1) and (s''_2, M''_2) such that
 - $(s'_1, M'_1) \rightarrow (s''_1, M''_1)$, $(s'_2, M'_2) \rightarrow (s''_2, M''_2)$
 - and $\Gamma \vdash M''_1 \approx_{A>L} M''_2$

Type System

Types

$$\tau ::= \text{int}_\ell \mid \text{int}_\ell \text{ ref} \mid \text{stmt}_{\mathcal{R}}$$

Expression Typing

$$\Gamma; \mathcal{R} \vdash n : \text{int}_\ell$$

$$\Gamma; \mathcal{R} \vdash \text{none} : \text{int}_\ell$$

$$\frac{\Gamma(m) = \text{int}_\ell}{\Gamma; \mathcal{R} \vdash m : \text{int}_\ell \text{ ref}}$$

$$\frac{m \notin \mathcal{R} \quad \Gamma(m) = \text{int}_\ell}{\Gamma; \mathcal{R} \vdash !m : \text{int}_\ell}$$

$$\frac{\Gamma; \mathcal{R} \vdash e_1 : \text{int}_{l_1} \quad \Gamma; \mathcal{R} \vdash e_2 : \text{int}_{l_2}}{\Gamma; \mathcal{R} \vdash e_1 + e_2 : \text{int}_{l_1 \sqcup l_2}}$$

Statement Typing Judgment

$$\Gamma ; \mathcal{R} ; pc \vdash s : \text{stmt}_{\mathcal{R}'}$$

- Means that in the context Γ and with unassigned output references R , and program counter label pc , s levels the output references R' unassigned after termination

Statement Typing: Skip and Seq

$$\Gamma ; \mathcal{R} ; pc \vdash \text{skip} : \text{stmt}_{\mathcal{R}}$$
$$\Gamma ; \mathcal{R} ; pc \vdash s_1 : \text{stmt}_{\mathcal{R}_1}$$
$$\Gamma ; \mathcal{R}_1 ; pc \vdash s_2 : \text{stmt}_{\mathcal{R}_2}$$

$$\Gamma ; \mathcal{R} ; pc \vdash s_1 ; s_2 : \text{stmt}_{\mathcal{R}_2}$$

Statement Typing: Assignment

$$\begin{array}{c} \Gamma; \mathcal{R} \vdash m : \text{int}_{\ell} \text{ ref} \quad \Gamma; \mathcal{R} \vdash e : \text{int}_{\ell'} \\ C(pc) \sqcup C(\ell') \leq C(\ell) \quad I(\ell) \leq I(pc) \sqcap I(\ell') \\ A_{\Gamma}(\mathcal{R}) \leq A(\ell') \end{array}$$

$$\Gamma; \mathcal{R}; pc \vdash m := e : \text{stmt}_{\mathcal{R} - \{m\}}$$

$$A_{\Gamma}(\mathcal{R}) = \bigsqcup_{m \in \mathcal{R}} A(\Gamma(m))$$

- The availability of e must be as high as the availability of every unassigned output reference

Statement Typing: If

$$\Gamma; \mathcal{R} \vdash e : \mathbf{int}_\ell \quad A_\Gamma(\mathcal{R}) \leq A(\ell)$$
$$\Gamma; \mathcal{R}; pc \sqcup \ell \vdash s_i : \tau \quad i \in \{1, 2\}$$

$$\Gamma; \mathcal{R}; pc \vdash \mathbf{if } e \mathbf{ then } s_1 \mathbf{ else } s_2 : \tau$$

Statement Typing: While

$$\Gamma \vdash e : \text{int}_\ell \quad \Gamma ; \mathcal{R} ; pc \sqcup \ell \vdash s : \text{stmt}_{\mathcal{R}}$$
$$A_\Gamma(\mathcal{R}) \leq I(\ell) \sqcap I(pc) \sqcap A(\ell)$$

$$\Gamma ; \mathcal{R} ; pc \vdash \text{while } e \text{ do } s : \text{stmt}_{\mathcal{R}}$$

Statement Typing: While

$$\frac{\Gamma \vdash e : \text{int}_\ell \quad \Gamma ; \mathcal{R} ; pc \sqcup \ell \vdash s : \text{stmt}_{\mathcal{R}} \quad A_\Gamma(\mathcal{R}) \leq I(\ell) \sqcap I(pc) \sqcap A(\ell)}{\Gamma ; \mathcal{R} ; pc \vdash \text{while } e \text{ do } s : \text{stmt}_{\mathcal{R}}}$$

`while (!m1) do skip; mo := 1;`

Statement Typing: While

$$\frac{\Gamma \vdash e : \text{int}_\ell \quad \Gamma ; \mathcal{R} ; pc \sqcup \ell \vdash s : \text{stmt}_{\mathcal{R}} \quad A_\Gamma(\mathcal{R}) \leq I(\ell) \sqcap I(pc) \sqcap A(\ell)}{\Gamma ; \mathcal{R} ; pc \vdash \text{while } e \text{ do } s : \text{stmt}_{\mathcal{R}}}$$

if (! m_1) then while (1) do skip; else skip;
 $m_0 := 1$;

Statement Typing: Subtyping

$$\frac{\Gamma ; \mathcal{R} ; pc \vdash s : \tau \quad \Gamma ; \mathcal{R} ; pc \vdash \tau \leq \tau'}{\Gamma ; \mathcal{R} ; pc \vdash s : \tau'}$$

$$\frac{\begin{array}{c} \mathcal{R}' \subseteq \mathcal{R}'' \subseteq \mathcal{R} \\ \forall m, m \in \mathcal{R}'' - \mathcal{R}' \Rightarrow A(\Gamma(m)) \leq I(pc) \end{array}}{\Gamma ; \mathcal{R} ; pc \vdash \text{stmt}_{\mathcal{R}'} \leq \text{stmt}_{\mathcal{R}''}}$$

Soundness

- If $\Gamma; R; pc \vdash s : \tau$, then
 - s has Confidentiality Noninterference w.r.t. Γ ,
 - s has Integrity Noninterference w.r.t. Γ , and
 - s has Availability Noninterference w.r.t. Γ and R

Extensions

- Timeouts
- New References

Summery

- Defined confidentiality, integrity, and availability as noninterference properties
- Provided a sound type system for inferring if a these properties hold given security levels for memory references

Concerns

- I/O model seems odd
- Availability noninterference does not imply availability

Musing

- If the high-availability inputs are available, the high-integrity references un-compromised, and the program terminates, then the high-availability outputs will become available
- Does something like $\Gamma; R; pc \vdash s : stmt \emptyset$ implies this?