Language Based Information Flow Security

Andrei Sabelfield, Andrew C. Myers

Presentation: Ashish Kundu
ashishk@cs.purdue.edu

Outline

• Security requirements
• Information flow – background
• Language-based information flow
• Open challenges
• Discussion
• Conclusion

Information flow?

![Diagram of information flow with confidentiality and openness]

Data flow

Information flow?

![Diagram of information flow with encryption and confidentiality]

Encrypted: $h \rightarrow l$

E.g. password sharing

Information flow?

![Diagram of information flow with trusted and non-trusted]

Data flow

Information flow?

![Diagram of information flow with confidentiality and openness]

Data flow
Information flow?

- **Confidentiality**: A rigorous requirement
  - can confidentiality guarantee of a system be proven?

Explicit Information Flow

Implicit Information Flow

Implicit Information Flow
**Property-I of IFlow**

- Confidentiality: A rigorous requirement
  - can confidentiality guarantee of a system be proven?
  - can explicit and implicit flows be controlled?
- Relationship with data and control dependency ???

**Covert channels**

- Implicit flows
  - covert
- Termination channel
  - termination-sensitive confidentiality
- Timing channels
  - subsumes termination channel
- Probabilistic channel
  - PDF of output data
- Resource exhaustion channel
  - memory or disk space: high value for malloc()
- Power channels
  - related: recent work about the age of running system
  - thus attack vulnerability

**Properties of IFlow**

- No propagation of high confidential data to low confidential container
- Rigor: On all paths - no leak
  - makes it easy for static-time solutions

**Mechanisms**

- Access control
  - controls release of information, not propagation
  - no control on “how data is used”
- Language-based techniques
  - Runtime: JVM – applets, sandbox
  - Bytecode verifier
    - no control on propagation
- **Type systems**

**Type systems**

- Compositional reasoning
  - incremental construction: from a correct system to a larger and correct system
  - structural induction (will return to this later)
  - objective: correct computation
  - modified objective: correct confidentiality-preserving computation

**Type systems**

- Compositional reasoning
  - incremental construction: from a correct system to a larger and correct system
  - structural induction (will return to this later)
- Objective: correct computation
  - modified objective: correct confidentiality-preserving computation
Explicit Information Flow

- Confidential
- Leak
- Open

Static Information Flow Control

- Program analysis: Denning and Denning
- Theorem provers
- Type checking

Type checking

- Security type systems
  - Ordinary type: int, char
  - Label: static labeling on its confidentiality semantics
- Static type checking detects leaks
  - Conservative: so false positive
  - Structural induction
  - Cannot completely control covert channels
    - Semantics – values \( \rightarrow \) Undecidability

Explicit Information Flow

- Good for static analysis
- Partial order
- Lattice model of confidentiality
- MAC

Non-interference

- high
- high
- high
- high
- high
- higher

\[ \{ \text{high} \} \rightarrow \text{non-interference} \]

\[ \{ \text{low} \} \]

No explicit or implicit path from any high to any low

Semantics-based security

- variation of high input does NOT lead to (observable) variation on low output

\[ \text{input state } s = (s_h, s_l) \]

\[ [C] : S \rightarrow S_1 \quad \widehat{S}_1 = S \cup \{ \perp \} \text{ and } \perp \notin S \]

\[ \text{output state } s' = (s'_{h}, s'_{l}) \]

Semantics-based security

- Two inputs are equivalent if they agree on low output values

\[ \forall s_1, s_2 \in S, s_1 =_{L} s_2 \Rightarrow [C] s_1 =_{L} [C] s_2 \]

\[ (\text{if } l = 5 \text{ then } h := h + 1 \text{ else } l := l + 1) \]
Semantics-based security

• \( l := h \)
• if \((h=3)\) then \(l:=5\) else skip

\[ \forall s_1, s_2 \in S. \ s_1 =_L s_2 \implies [C]s_1 =_L [C]s_2 \]

Security Type System

[1] \( \vdash \ exp \ : \ high \quad \vdash \ exp \ : \ low \)

[C1-3] \( [pc] \vdash \skip \quad [pc] \vdash \text{if } h := \exp \) \[\vdash \exp \ : \ low \]

[C4-5] \( [pc] \vdash \ \exp \ : \ pc \quad [pc] \vdash \ C_1 \)

[C6-7] \( [pc] \vdash \ \exp \ : \ pc \quad [pc] \vdash \ C_1 \quad [pc] \vdash \ C_2 \)

Directions

• Expressiveness
• Concurrency
• Covert channels
• Refining security policies

Expressiveness

• Functions
  – SLam: First-class functions [Heintze et al]
    • non-interference
  – First-class continuations [Zdancewic et al]
    • non-interference

• Exceptions
  – explicit and implicit flows
  – path labeling by Myers

• JFlow by Myers: Java – Jif compiler
Concurrency

• Nondeterminism

Concurrency

• Nondeterminism: possibilistic security condition
  – set of high inputs may not affect set of low outputs
  – dependencies between variables

Concurrency

• Nondeterminism: possibilistic security condition
  – equational security property

\[ \forall s \in S. [\text{HH}; C; \text{HH}]s \approx [C; \text{HH}]s \]

Concurrency

• Thread concurrency
  – non-atomicity

• Non-interference requirements:
  – no “high” guard in a while loop
  – no if with “high” guard having a while loop in its branch
• termination leak
• timing leak

Concurrency

• Thread concurrency
  – non-atomicity

• Non-interference requirements:
  – no “high” guard in a while loop
  – no if with “high” guard having a while loop in its branch
• termination leak
• timing leak

\[(\text{if } h = 1 \text{ then } C_{\text{long}} \text{ else skip; } l := 1 \parallel l := 0)\]
Concurrency

- Thread concurrency
  - non-atomicity

- Scheduler-independent security
  - uniform scheduler (Sabelfield and Sands)

- Type systems: rule out synchronization on “high” data.
  - Sabelfield

Distributed programs

- non-trusted parties
- parties’ concurrency property
- failures

- Secure program partitioning: high and low

Discussion

- Illustrated Security type system: simple yet powerful
  - expressive
  - precise
  - easily extensible to a lattice model of access control

- Organization of the survey addresses
  - all language-level factors clearly and precisely
  - illustrates important issues and challenges with simple examples
  - considers both formal approaches and informal approaches in the light of the
    - hardness
    - undecidability of the general nature of the problem

Critique

- Presentation very compact: lacking
  - useful illustration and explanation of the concepts and approaches
  - relation between various approaches need to be established

- How to make the approaches such as security type systems part of pragmatic languages

- Needed to address program certification more detailed in a compositional framework

Some Ideas

- Slicing towards proving non-interference

- Use of SSA in checking policy-violations

Some Ideas

- Error Handling: an error violation of integrity policy
  - dual of confidentiality: <high, low> :: <low’, high’>

- Exceptions resulting in termination
  - illegal flow of information?
  - self-healing systems