Information Flow Security (2)

DD2460 Software Safety and Security: Part III, lecture 3

Gurvan Le Guernic

DD2460 (III, L3)
February 24th, 2012

Outline

Information Flow Security deals with *Confidentiality* and *Integrity* related security policies.

1. Noninterference Variants
2. Enforcement Techniques
3. Conclusion / Wrap-up

Noninterference Variants

Termination (In)sensitive Noninterference

Main idea: attacker is (un)able to observe ($\mathscr{O}$) if execution terminated or not

$$\forall \sigma_1, \sigma_2 : \ \sigma_1 =_L \sigma_2 \Rightarrow \mathscr{O}[\![\sigma_1 \vdash \mathtt{P}]\!] = \mathscr{O}[\![\sigma_2 \vdash \mathtt{P}]\!]$$

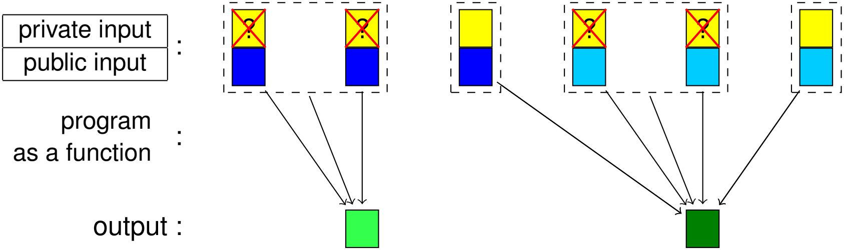
- Sensitive: tag termination into observables
- Insensitive (1): observable prefixes of nonterminating executions
- Insensitive (2): discard non-terminating executions (σ)

Declassification

Definition 1 (Noninterference modulo declassification ϕ)

A program is safe if and only if any executions, started with the same public inputs *and agreeing on* ϕ, output the same sequence.

ϕ = secret is or is not yellow



Taint Analysis

Takes into account only (direct) explicit flows

Weaker security guarantees, but more efficient enforcement mechanisms

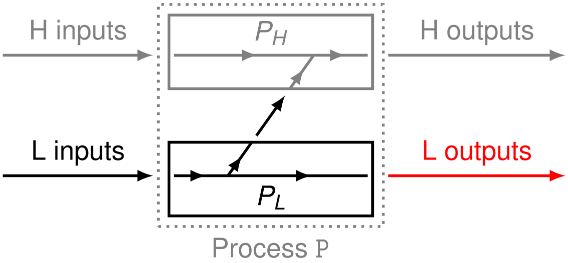
- not efficient against malicious code, but OK against buggy code

Examples:

- Python's taint library
- Perl taint mode
- ...

Enforcement Techniques

Noninterference Enforcement: Main Idea



Old Security Mechanism: Confined Processes

Lampson's 1973 notion of *confinement*

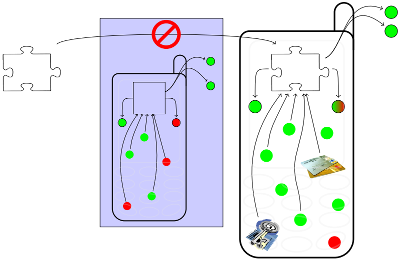
Confined processes:

- are memoryless ($\Rightarrow$ side-effect free)
- call only confined processes, but can be called by unconfined processes
- have masked output belonging to a predefined set
  - could extend to label verification

Main concepts underlying *sandboxing*

- one of Java's main security mechanisms

Static Information Flow Analysis



Principles:

- analyze IF before execution
- do nothing during execution

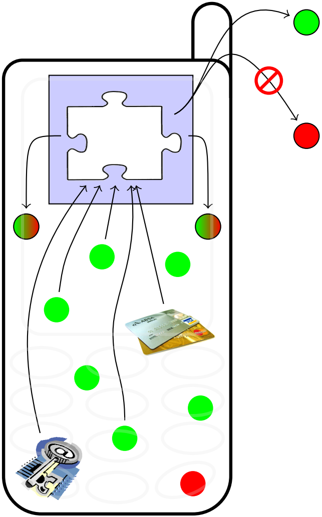
Advantages:

- no runtime overhead
- run iff NI is proved
- old strong soundness culture

Main drawback:

- can be too restrictive

Dynamic Information Flow Analysis



Principles:

- track flows at execution
- prevent data leak just before it occurs

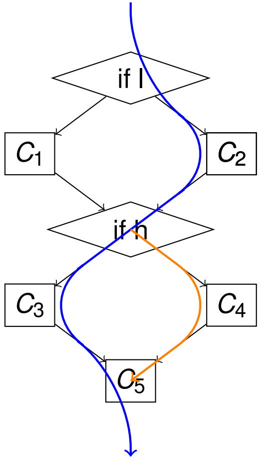
Advantages:

- enforce runtime policies more easily
- allow *safe* executions of unsafe programs
- may be more precise in some cases
  - reduced space (not all executions)
  - access to runtime values

Main drawback:

- hard to spot all flows (implicit flows)

Hybrid Information Flow Analysis



Principles: mix of static and dynamic analyses

- dynamically analyze C_2 and C_3
  - for *direct* and *explicit indirect* flows
- statically analyze C_4
  - for *implicit indirect* flows
- dynamically analyze C_5 with results of C_3 and C_4 mixed

Advantages:

- best of both worlds

Main drawback:

- worst of both worlds
- higher complexity

Is Detection Enough?

What happens with an analysis which is *sound* with regard to information flow detection?

- Static analysis:

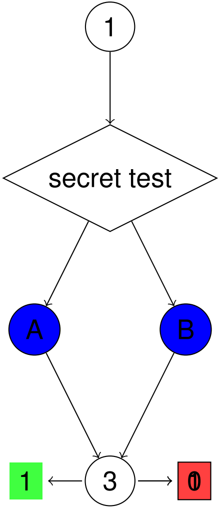
  Expert: "You should not use this program!"

- Dynamic analysis:

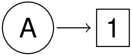
  ATM: "Oh, by the way, I probably sent your PIN code all over the web."

A user expects dynamic IF analyses to detect *and correct* information flows.

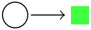
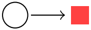
The Correction Pitfall



Code block A outputs value 1:

- 

Analysis concludes:

- public data: 
- secret data: 

Sound detection *does not* imply sound (detection + correction)

- dynamic analysis + "stop" correction
- "stop" correction with termination insensitive NI proof

Conclusion / Wrap-up

3 Most Important Points

- ∃ many information flow security policy variants
  - termination sensitivity
  - declassification
  - ...
  - taint analyses

- Enforcement
  - Static analyses: (**+**) soundness (**-**) usability (often too restrictive)
  - Dynamic analyses: (**+**) usability (**-**) soundness
  - Hybrid analyses: (**+**/**-**) soundness & usability (**-**) complexity

- Correction pitfall
  - dynamic and hybrid analyses require correction mechanism
  - sound detection $\not\Rightarrow$ sound (detection + correction)

IF Workshop

Goal: simulate review of some existing IF security techniques

- you *do not* need to *defend* or *kill* your paper
- you need to:
  - describe the enforcement technique used [and its implementation] (for reproducibility)
  - evaluate the level of security provided
  - describe advantages and limitations of the technique
  - compare with other known techniques:
    - workshop: type system + taint analysis
    - report: type system + taint analysis + workshop techniques

After the workshop and report, I/you should be able to pick up the best adapted tool/technique for a particular IF problem.

Grading

Workshop presentation is not graded per se (report is) [due 12/3]

- E:
  - give a decent presentation (or at least additions/corrections session)
  - be able to give an accurate description/summary of the paper at the course level
- C: (subsumes E)
  - detail specific advantages and limitations of the paper's technique
- A: (subsumes A)
  - compare with the relevant techniques presented in class and in the other papers

Level of learning of course material also reflected in the final grade

- if/where possible, report should contain proof of knowledge of channels, flows, labels, noninterference, enforcement, . . .

Information Flow Wrap-up

	Concepts Definitions	Enforcement		
		Type System	Taint	Others
basic	lectures 1 & 3: IF policies, channels, flows, labels, correction, . . . E C	lecture 2: type systems, noninterference, . . . A		lecture 3: static, dynamic, hybrid, . . .
deeper	exercises 1 & 2: IF policies, timing channels, flows, . . .	exercises 2: type systems, Jif, . . .	exercises 1: taint, . . .	workshop

Course Wrap-up

Software safety and security:

- prevent bad behaviors causing system (base) and data (load) damage
- due to specification and/or implementation errors and/or weaknesses

Formal methods:

- precise correctness guarantees
- often complex and expensive
- for critical systems and/or data

3 different techniques for software safety and security

- Temporal logic and model checking
- Hoare logic and VCG/symbolic execution
- Information flow and type system

Announcements and Questions?

Soon online:

- lab 2 booking
- course evaluation

Questions?