Static Analysis of Security Properties by Abstract Interpretation

École normale supérieure, équipe ABSTRACTION

Mehdi Bouaziz

Friday, May 11 2012

Static Analysis of Security Properties by Abstract Interpretation

Static Analysis by Abstract Interpretation

$\longrightarrow$ course MPRI 2-6:

Abstract Interpretation: application to verification and static analysis

Security Properties

$\longrightarrow$?

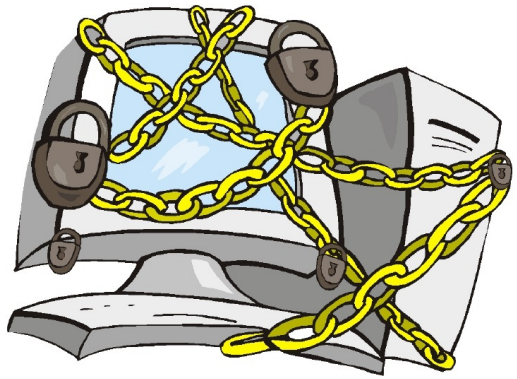
Security?


Security


Transmission Security
Financial Security
Complete
Security
Data Security
Staff Security

Security?

Information Security?



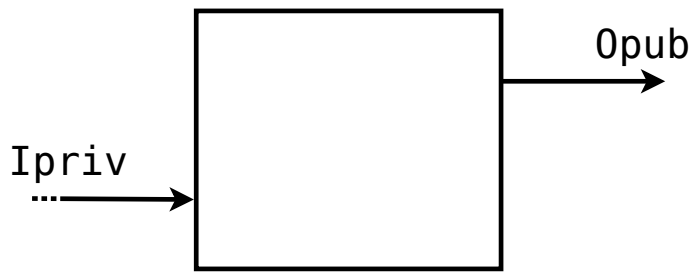
Security?

Access Control
Accountability
Attack
Authenticity
Authorization
Availability
Buffer Overflow
Bug
Classification
Confidentiality
Control-Flow
Covert Channels
Cross-Site Scripting
Cryptanalysis
Cryptography
Cryptology
Dangling Pointer
Data Race
Declassification
Deadlock
Earthquake
Encryption
Fire
Firewall
Flooding
Format String
Implicit Flow
Information-Flow
Input Validation
Integrity
Isolation
Language-Based
Least Privilege
Malicious Code
Memory Safety
Non-Interference
Non-Repudiation
Obfuscation
Phishing
Policy
Possession
Randomization
Reference Monitor
Risk
Runtime Check
Sandbox
SQL Injection
Stack Inspection
Stack Overflow
Symlink Race
Tainting
Theft
Threat
Type Safety
Utility
Vulnerability
Wild Jump

Key Concepts

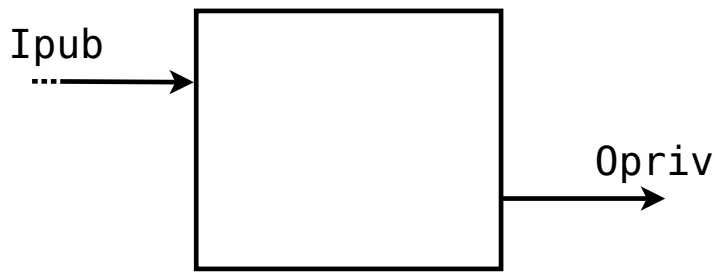
Key Concepts

- Confidentiality



Key Concepts

- Confidentiality
- Integrity



Key Concepts

- Confidentiality
- Integrity
- Disponibility

Key Concepts

- Confidentiality
- Integrity
- Disponibility

- Authenticity
- Accountability
- Possession
- Non-repudiation
- Utility

Security Policy

A statement that partition the states of the system into a set of authorized (secure) states and a set of unauthorized (nonsecure) states.

Security Policy

A statement that partition the states of the system into a set of authorized (secure) states and a set of unauthorized (nonsecure) states.

Specify who can read/write what data, execute what command, under which condition.

Security Policy

A statement that partition the states of the system into a set of authorized (secure) states and a set of unauthorized (nonsecure) states.

Specify who can read/write what data, execute what command, under which condition.

- Natural language (law, documentation)
- Encoded text (`755 root root /bin`)

Security Policy

A statement that partition the states of the system into a set of authorized (secure) states and a set of unauthorized (nonsecure) states.

Specify who can read/write what data, execute what command, under which condition.

- Natural language (law, documentation)
- Encoded text (`755 root root /bin`)
- Code (`if (x.isPrivate()) exit(1); //avoid leak`)

Security Policy

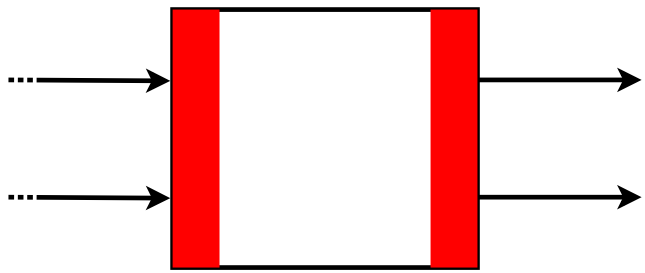
A statement that partition the states of the system into a set of authorized (secure) states and a set of unauthorized (nonsecure) states.

Specify who can read/write what data, execute what command, under which condition.

- Natural language (law, documentation)
- Encoded text (`755 root root /bin`)
- Code (`if (x.isPrivate()) exit(1); //avoid leak`)
- $\emptyset$

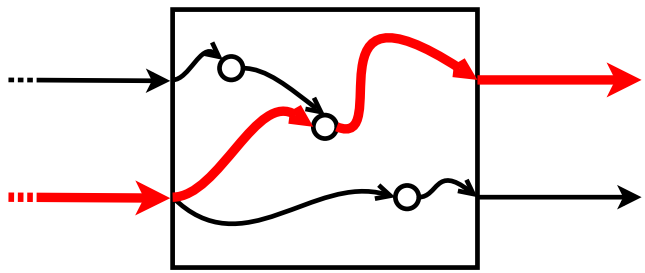
Information Security Controls

- Access Control
- Information-Flow Control
- Control-Flow Integrity
- Encryption



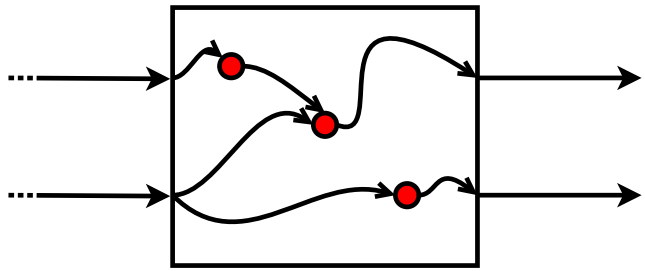
Information Security Controls

- Access Control
- Information-Flow Control
- Control-Flow Integrity
- Encryption



Information Security Controls

- Access Control
- Information-Flow Control
- Control-Flow Integrity
- Encryption



Information Security Controls

- Access Control
- Information-Flow Control
- Control-Flow Integrity
- Encryption

$\longrightarrow$ courses
MPRI 1-13: Initiation to cryptology
MPRI 2-12-1: Cryptanalysis
MPRI 2-12-2: Arithmetic algorithms for cryptology
MPRI 2-13-2: Error correcting codes and applications to cryptography
MPRI 2-30: Cryptographic protocols: computational and symbolic proofs

Threats

- Physical: Earthquake, Fire, Flooding, Theft
- In the code:
  - Memory Safety:
    - Buffer Overruns
    - Stack Overflow
    - Dangling pointers
  - Concurrency:
    - Deadlocks
    - Data races
    - Symlink races
  - Input Validation:
    - SQL injection
    - Cross-Site Scripting (XSS)
    - Format String
  - Control/Data-Flow:
    - Type Safety
    - Wild Jumps
    - Self Modifying Code

Language-Based Mechanisms

- Runtime Checks: Reference Monitor (OS, Interpreter, Firewall), Inlined Reference Monitor
- Programming Languages: Type-Safe Languages, Typed Assembly Language (TAL)
- Executing Model: Isolation, Sandboxing, Stack Inspection
- Static Analysis: Information-Flow Typing, Abstract Interpretation
- Exotic: Obfuscation, Randomization

Security Policy (2)

- Authorization
- History-Based
- Control-Flow
- Information-Flow
- Classification (private/public)
- Declassification (when, where, by who and what private information can be considered public)

Information-Flow Security

Non-Interference: *No two executions are observably different if they differ solely by confidential inputs.*

Explicit Flows: from assignments

Implicit Flows: from Indirect Flows and Covert Channels:

- Termination Channel
- Timing Channel
- Probabilistic Channel
- Resource Exhaustion Channel
- Power Channel

Information-Flow Security Type System

$$\vdash exp : high \qquad \frac{h \notin Vars(exp)}{\vdash exp : low}$$

$$[pc] \vdash \texttt{skip} \qquad [pc] \vdash h := exp \qquad \frac{\vdash exp : low}{[low] \vdash l := exp}$$

$$\frac{[pc] \vdash C_1 \qquad [pc] \vdash C_2}{[pc] \vdash C_1; C_2} \qquad \frac{\vdash exp : pc \qquad [pc] \vdash C}{[pc] \vdash \texttt{while}\ exp\ \texttt{do}\ C}$$

$$\frac{\vdash exp : pc \qquad [pc] \vdash C_1 \qquad [pc] \vdash C_2}{[pc] \vdash \texttt{if}\ exp\ \texttt{then}\ C_1\ \texttt{else}\ C_2} \qquad \frac{[high] \vdash C}{[low] \vdash C}$$

Issues

Non-interference is too restrictive. Most real-world programs need exceptions to non-interference: declassification.

Examples?

Issues

Non-interference is too restrictive. Most real-world programs need exceptions to non-interference: declassification.

Examples?

Other issues:

- Expressiveness: first-class functions, exceptions, objects
- Concurrency: threads, nondeterminism, distribution
- Covert channels: termination, timing, probability
- Security policies: declassification, quantitative security, dynamic policies
- Certification: proven compilers, proof-carrying codes

Thank you for listening

Questions are welcome