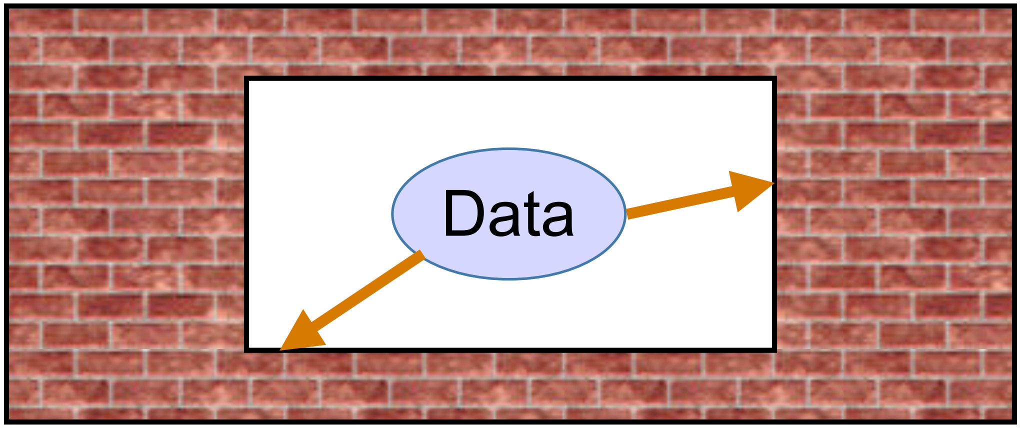
Information-flow Security ? Provenance

Steve Zdancewic

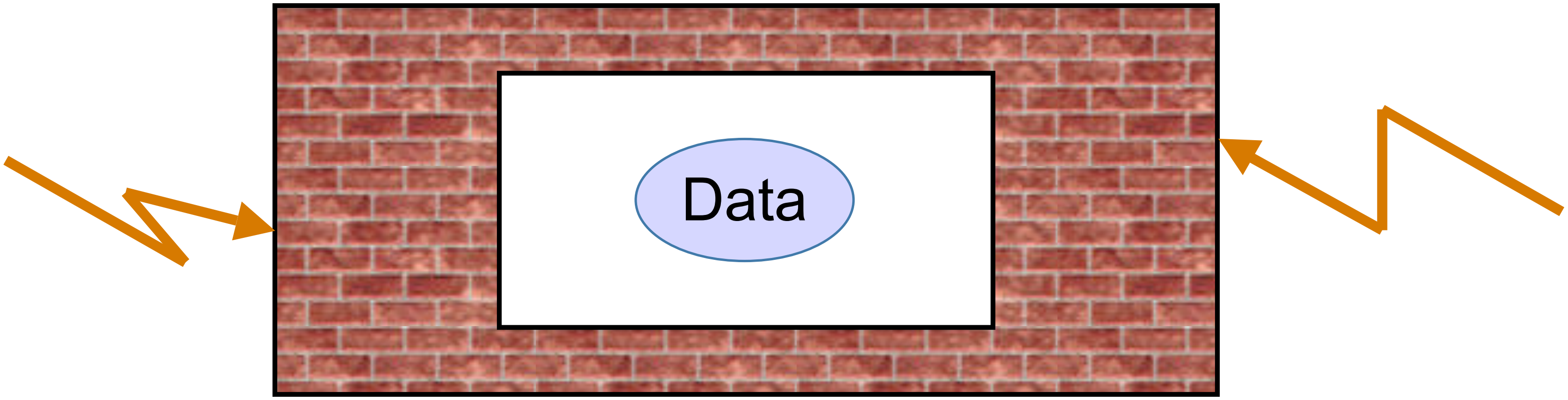
University of Pennsylvania

Quality 1: *Confidentiality*



- Keep data or actions *secret.*
- Related to: Privacy, Anonymity, Secrecy
- Authorized *reading* of data
- Examples:
  - Pepsi secret formula
  - Medical information
  - Personal records (e.g. credit card information)
  - Military secrets (Unclassified, Classified, Secret, Top Secret)

Quality 2: *Integrity*



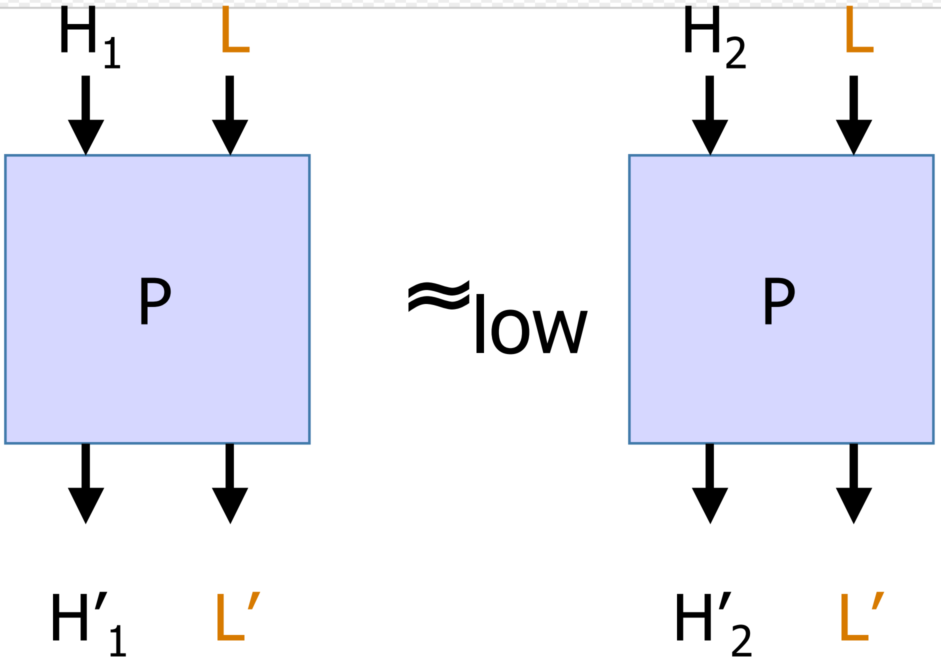
- Protect the *reliability* of data against unauthorized tampering
- Related to: Corruption, Forgery, Consistency
- Authorized *writing/creation* of data
- Example:
  - Bank statement agrees with ATM transactions
  - The mail you send is what arrives
  - No system call is passed untrusted inputs (e.g. in Perl)

Information-flow Security

- *Not access control*
- Concern is tracking *flow* of information through a program or a system.
- Main idea: *Label* data with security levels and restrict use based on those levels.
  - Labels are ordered: $L \leq H$

    (Higher = more "confidential" or more "tainted")
  - Dynamically: tag values, propagate them
  - Statically: In a type system: $(\mathrm{bool}_H \to \mathrm{bool}_L)_L$
  - Noninterference Theorem implies:
    A function of type $(\mathrm{bool}_H \to \mathrm{bool}_L)_L$ is constant;
    no information is leaked from H to L.

Noninterference



- Every notion of program equivalence yields a viable definition of "information flow"
  - There is no single definition that applies universally
- Proof techniques:
  - Fundamentally, noninterference is a property that relates pairs of evaluations
  - Logical relations or Bisimulation techniques

Historical Context

- Label Models:
  - Bell & LaPadula 1975: military's "no read up, no write down"
  - Biba 1977: integrity variant
- Original formulation: Trace models of computation
  - Goguen & Meseguer 1982
  - McClean – late 1980's early 1990's
- Dorothy Denning's program analysis techniques
  - Proposed a "lattice model" for secure program analysis
  - Mid-late 1970's (but no proofs of correctness)
- Volpano & Smith 1996
  - Type system (static analysis) for noninterference
- Much, much more recent work
  - See Sabelfeld & Myers 2003 for survey of ~150 papers.

PL Focus w.r.t. Information Flow

- Label models:
  - Theory: typically assumes a join semi-lattice (often with meets too)
  - Practice: Myers & Liskov's Decentralized Label Model
  - Variants: e.g. "dynamic labels" -- labels that are themselves program values
- Programming features:
  - Label inference
  - Label-generic functions (i.e. label polymorphism)
  - Declassification / Endorsement

Programming Language Results

- Information-flow analysis is known to be undecidable in the worst case
- In the presence of side effects (mutable state, nontermination, I/O, etc.) *static analysis* is essential for precise reasoning about information flow.
  - Most approaches approximate control flow and side effects
- Most analyses have focused on confidentiality
  - Integrity is usually treated as the dual to confidentiality
  - Integrity is probably closer to "provenance"
  - But… the duality is not completely satisfactory
    [Li, Mao, Zdancewic. Information Integrity Policies. FAST 2003]

Relevance to Provenance?

- Hypothesis: Integrity analysis == Provenance
- PL research might yield:
  - Precise definitions for a variety of models: probabilistic/nondeterministic/etc.
  - Formalization techniques
  - Ideas for static analysis of queries
- More connections???
  - I don't know -- that's why I'm here!