CS52600: Information Security

Information Flow

September 27, 2010

Prof. Ninghui Li

What is the point? *Information Flow*

- Policy governs flow of information
  - *How do we ensure information flows only through governed channels?*
- State transition attempts to capture this
  - We may return to this later
- Next: How do we measure/capture flow?
  - Entropy-based analysis
    - Change in entropy ⇒ flow
  - Confinement
    - “Cells” where information does not leave
  - Language/compiler based mechanisms?
    - Type-based tracking of flow
  - Guards

2

Information Flow

- Information Flow: Where information can move in the system
- How does this relate to confidentiality policy?
  - Confidentiality: What subjects can see what objects
  - Flow: Controls what subjects actually see
- Variable *x* holds information classified *S*
  - $\underline{x}$, information flow class of *x*, is *S*
- Confidentiality specifies what is allowed
- Information flow describes how this is enforced

3

Formal Definition

- Problem: capturing *all* information flow
  - Files
  - Memory
  - Page faults
  - CPU use
  - ?
- Definition: Based on *entropy*
  - Flow from *x* to *y* (times *s* to *t*) if $H(x_s \mid y_t) < H(x_s \mid y_s)$

4

What is Entropy?

- Idea: Entropy captures *uncertainty*
  – $H(X) = -\Sigma_j P(X=x_j) \lg P(X=x_j)$
- Entropy of a coin flip
  – $H(X) = -\Sigma_{j=heads,tails} P(X=x_j) \lg P(X=x_j)$
  – = -(P(heads) lg P(heads) + P(tails) lg P(tails))
  – = - (.5 lg .5 + .5 lg .5) = - (.5 * -1 + .5 * -1) = 1

  Complete uncertainty!
- Conditional Entropy:
  – $H(X|Y) = -\Sigma_j P(Y=y_j)[\Sigma_i P(X=x_i|Y=y_j) \lg P(X=x_i|Y=y_j)]$

5

Formal Definition

- Flow from *x* to *y* if $H(x_s \mid y_t) < H(x_s \mid y_s)$
  – $-\Sigma_i P(y_t=y_j)[\Sigma_i P(x_s=x_i| y_t=y_j) \lg P(x_s=x_i| y_t=y_j)] <$
  $-\Sigma_j P(y_s=y_j)[\Sigma_i P(x_s=x_i| y_s=y_j) \lg P(x_s=x_i| y_s=y_j)]$
- Has the uncertainty of x_s gone down from knowing y_t?
- Examples showing possible flow from *x* to *y*:
  – y := x
    - No uncertainty – $H(x|y) = 0$
  – y := x / z
    - Greater uncertainty (we only know *x* for some values of *y*)
  – Why *possible*?
  – Does information flow from *y* to *x*?
- What if y_s not defined?
  – Flow if $H(x_s \mid y_t) < H(x_s)$

6

Implicit flow

- Implicit flow: flow of information without assignment
- Example:
  - if (x =1) then y :=0 else y := 1
- *This is why the entropy definition is necessary!*

7

How do we Manage Information Flow?

- Information flow policy
  - Captures security levels
  - Often based on *confinement*
  - Principles: Reflexivity, transitivity
- Compiler-based mechanisms
  - Track potential flow
  - Enforce legality of flows
- Execution-based mechanisms
  - Track flow at runtime
  - Validate correct

8

Confinement Flow Model

- (I, O, *confine*, $\rightarrow$)
  - $I = (SC_I, \leq_I, join_I)$: Lattice-based policy
  - O: set of entities
  - $\rightarrow$: $O \times O$ indicates possible flows
  - $confine(o)$: $SC_I \times SC_I$ is allowed flow levels
- Security requirement
  - $\forall\, a,b \in O: a \rightarrow b \Rightarrow a_L \leq_I b_U$
- Similar definitions possible for more general levels
  - non-lattice
  - non-transitive

9

Compiler Mechanisms

- Declaration approach
  - x: *integer* **class** { *A*,*B* }
  - Specifies what security classes of information are allowed in x
- Function parameter: class = argument
- Function result: class = $\cup$ parameter classes
  - Unless function verified stricter
- Rules for statements
  - Assignment: LHS must be able to receive all classes in RHS
  - Conditional/iterator: then/else must be able to contain if part
  - Composition
- *Verifying a program is secure becomes type checking!*

10

Execution Mechanisms

- Problem with compiler-based mechanisms
  - *May be too strict*
  - Valid executions not allowed
- Solution: run-time checking
- Difficulty: *implicit* flows
  - **if** x=1 **then** y:=0;
  - When x:=2, does information flow to y?
- Solution: Data mark machine
  - Tag variables
  - *Tag Program Counter*
  - Any branching statement affects PC security level
    - Affect ends when “non-branched” execution resumes

12

Data Mark: Example

- Statement involving only variables *x*
  - If PC ≤ *x* then *statement*
- Conditional involving *x*:
  - Push PC, PC = *lub*(PC,*x*), execute inside
  - When done with conditional statement, Pop PC
- Call: Push PC
- Return: Pop PC
- Halt
  - if *stack empty* then *halt execution*

13

Flow Control: Specialized Processor

- Security Pipeline Interface
  - Independent entity that checks flow
  - Could this manage confidentiality?
  - *Useful for integrity!*

15

Confinement

- Confinement Problem
  - Prevent a server from leaking confidential information
- Covert Channel
  - Path of communication not designed as communication path
- Transitive Confinement
  - If a confined process invokes a second process, invokee must be as confined as invoker

17

Isolation

- Virtual machine
  - Simulates hardware of an (abstract?) machine
  - Process confined to virtual machine
    - Simulator ensures confinement to VM
  - Real example: IBM VM/SP
    - Each user gets “their own” IBM 370
- Sandbox
  - Environment where actions restricted to those allowed by policy

18

Covert Channels

- Storage channel
  - Uses attribute of shared resource
- Timing channel
  - Uses temporal/ordering relationship of access to shared resource
- Noise in covert channel
  - Noiseless: Resource only available to sender/receiver
  - Noisy: Other subjects can affect resource

19

Modeling Covert Channels

- Noninterference
  - Bell-LaPadula approach
  - All shared resources modeled as subjects/objects
  - Let $\sigma \in \Sigma$ be states. Noninterference secure if $\forall s$ at level $l(s)$ $\exists \equiv : \Sigma \times \Sigma$ such that
    - $\sigma_1 \equiv \sigma_2 \Rightarrow \text{view}(\sigma_1) = \text{view}(\sigma_2)$
    - $\sigma_1 \equiv \sigma_2 \Rightarrow \text{execution}(i,\sigma_1) \equiv \text{execution}(i,\sigma_2)$
    - if i only contains instructions from subjects dominating s, $\text{view}(\text{execution}(i, \sigma)) = \text{view}(\sigma)$
- Information Flow analysis
  - Again model all shared resources

20

Covert Channel Mitigation

- Can covert channels be eliminated?
  - Eliminate shared resource?
- *Severely* limit flexibility in using resource
  - Otherwise we get the halting problem
  - Example: Assign fixed time for use of resource
    - *Closes timing channel*
- Not always realistic
  - *Do we really need to close every channel?*

21

Covert Channel Analysis

- Solution: Accept covert channel
  - But analyze the capacity
    - *How many bits/second can be "leaked"*
- Allows cost/benefit tradeoff
  - Risk exists
  - Limits known
- Example: Assume data time-critical
  - Ship location classified until next commercial satellite flies overhead
  - Can covert channel transmit location before this?

22

Example: Covert Channel Analysis

23

