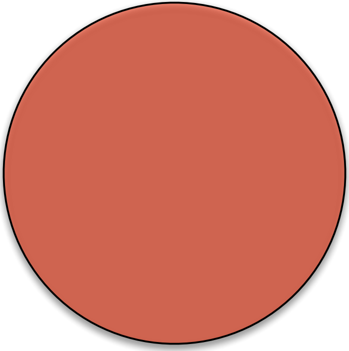
Post-Hoc Information Flow Analysis With Graph Algorithms

Lee Pike
leepike@galois.com

Galois Inc.

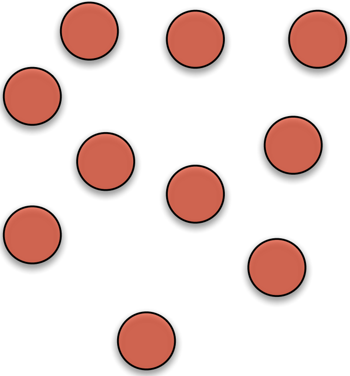
August 10, 2009

In The Beginning. . .



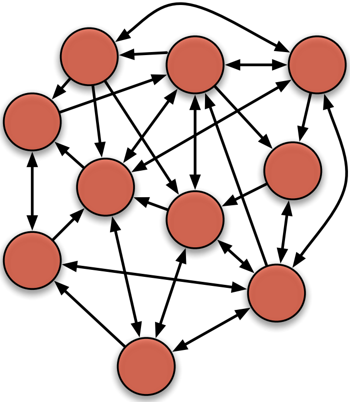
You have some monolithic system. . .

In The Beginning. . .



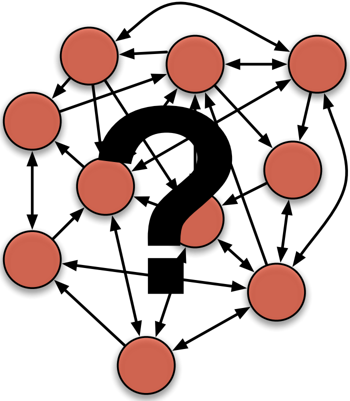
To make it more secure, you decompose it. . .

In The Beginning. . .



But then you connect the components to deliver functionality.

In The Beginning. . .



Wait: is this decomposition any more secure than what we started with?

Security by Decomposition

The answer depends:

1. Are the channels "better" (more secure APIs, more secure channels, etc.) in the decomposition than in the monolithic system?
2. Has our decomposed architecture reduced the possible information flow in the system?

This talk will address the 2nd issue by describing graph algorithms for analyzing basic information flow properties.

Motivations

Kinds of questions we'd like to answer:

- Which entities cannot transmit or receive data of some datatype?
- From an entity *e*, if *n* entities were to be compromised, to which other entities could data be leaked?
- Could data get from one entity to another in *n* or fewer steps?
- What channels between entities would have to be omitted to satisfy some separation policy?

Approach & Motivation

- Describe a simple model of information flow based on graphs.
- Describe how to combine graph algorithms for heuristics in information flow analyses.
- Why?
  - We've seen these ideas in the "folklore".
  - But not collected together coherently.
  - And they've proven useful to us.

Non-goals

Non-goals include modeling

- covert channels analysis
- information hiding/encoding
- cryptographic analysis

A Model for Information Flow

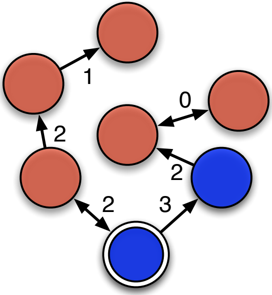
Details

- A system is a directed labeled graph $(V, L, \rightarrow)$.
  - V is a finite set of vertices (modeling computers, virtual machines, files, processes, threads, ports, etc.).
  - L is a finite set of edge labels (modeling data type restrictions).
  - $\rightarrow \subseteq L \times V \times V$: a nonempty relation between edges and labels (modeling shared bits, shared pages, firewalls with rules, ports, etc.).
- $\leq$ be a partial-order on edge labels giving an ordering to information flow.

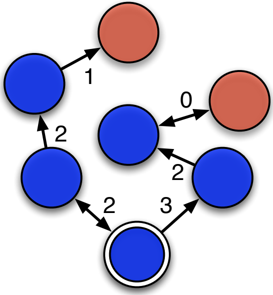
  For example, for channels c and c', $c \leq c'$ if c' expects integers and c expects natural numbers.

Transitive Closures

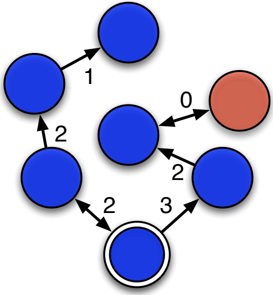
For analyses, we’ll be taking the *transitive closure of vertices with respect to labels*, including all dominating channels in the relation:



Label 3



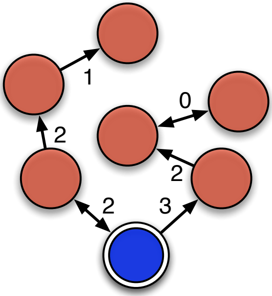
Label 2



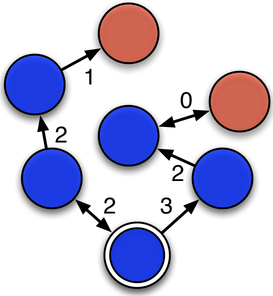
Label 1

k-Transitive Closures

We'll also take *k*-transitive closures, which are transitive closures (with respect to a label) to some depth:



$k = 0$, label 1



$k = 2$, Label 1

Confidentiality & Integrity

- Confidentiality: From a fixed vertex, to which other vertices can data of type *l* flow?
- Integrity: To a fixed vertex, from which other vertices can data of type *l* flow?

So in our model, integrity is the dual of confidentiality (i.e., reverse the edge relation).

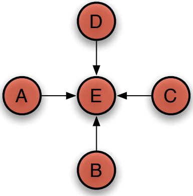
Transitive Closures

Interpretation

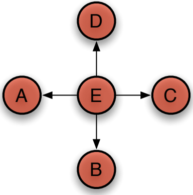
- (k)-transitive closures give us an intuition about what information can go where, from a specific vertex.
- But this is a local picture of information flow.
- We want graph abstractions.
  - Maximum confidentiality closures (Max CC) w.r.t. a label. Max CCs are the sets of transitive closures that have no strict supersets as transitive closures.
    Intuition: Max CCs represent the greatest distribution of information from sources.
  - Minimum confidentiality closures (Min CC) w.r.t. a label. Min CCs are the sets of transitive closures that have no strict subsets as transitive closures.
    Intuition: Min CCs represent the data sinks in a system.

Maximum Confidentiality Closures

- Max CC: $\{\{A, E\}, \{B, E\}, \{C, E\}, \{D, E\}\}$
- Min CC: $\{\{E\}\}$



- Max CC: $\{\{A, B, C, D, E\}\}$
- Min CC: $\{\{A\}, \{B\}, \{C\}, \{D\}\}$



Separation Policies

- In the beginning, we talked about system decomposition, ostensibly to make the system “more secure”.
- By “more secure”, we might mean federated according to some separation policy.
- Challenge: how can channels be modified to deliver functionality while ensuring separation?

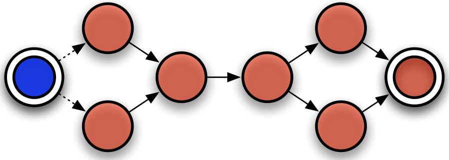
Separation Policies

Definition

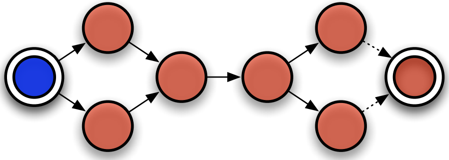
- Our notion of a separation policy is a relation between vertices, w.r.t. a lablel: $\not\rightarrow \subseteq L \times V \times V$.
- A graph respects a separation policy iff for all v, v', $v \not\rightarrow v'$ implies $v' \notin TC(v)$ (TC is the transitive closure), w.r.t. to a label.

Three Separation Algorithms

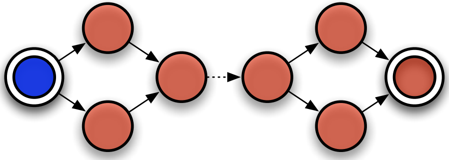
1. Separate the source.



2. Separate the sink.



3. Take a minimum cut.



Minimum Cut

Details

- Reducible to the *hitting set problem*:

  Given a set X and a collection of subsets $\mathcal{S}$ of X, find the subset of $\mathcal{S}$ exactly covering X, if one exists.
- The hitting set problem is NP-complete.
- A greedy approximation exists, polynomial in the size of the separation policy. The error is bounded by $\ln(x) + 1$, where x is the maximal number of paths violating a policy element.

Heuristics

For Describing Separation Policies

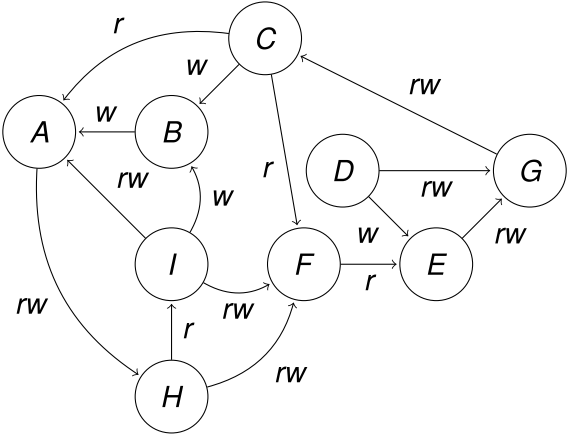
Idea: how to carve a system at the joints.

- Compute cliques and independent sets.
- Find graph partitions (NP-complete) to find subgraphs with minimized edges between subgraphs. Borrows from workload balancing.

Putting Things Together

Extended Example

- Set of processes with access to a shared file system.
- Labels: $r \leq rw$ and $w \leq rw$.



Discretionary access control: processes can provide other processes with read and/or write privileges on files for which they have those privileges.

Some Analyses

(1/4)

Let's take max confidentiality closures first:

- $\mathit{read}()$: $\{\{A, C, D, E, F, G, H, I\}, \{B\}\}$
- $\mathit{write}()$: $\{\{A, B, C, D, E, F, G, H\}, \{A, B, F, H, I\}\}$
- $\mathit{read} - \mathit{write}()$: $\{\{A, F, H, I\}, \{B\}, \{C, D, G\}, \{C, E, G\}\}$

Some Analyses

(1/4)

Let’s take max confidentiality closures first:

- $\mathit{read}()$: $\{\{A, C, D, E, F, G, H, I\}, \{B\}\}$
- $\mathit{write}()$: $\{\{A, B, C, D, E, F, G, H\}, \{A, B, F, H, I\}\}$
- $\mathit{read} - \mathit{write}()$: $\{\{A, F, H, I\}, \{B\}, \{C, D, G\}, \{C, E, G\}\}$

- So *B* cannot obtain read privileges from any other process.

Some Analyses
(1/4)

Let's take max confidentiality closures first:

- $\mathit{read}()$: $\{\{A, C, D, E, F, G, H, I\}, \{B\}\}$
- $\mathit{write}()$: $\{\{A, B, C, D, E, F, G, H\}, \{A, B, F, H, I\}\}$
- $\mathit{read} - \mathit{write}()$: $\{\{A, F, H, I\}, \{B\}, \{C, D, G\}, \{C, E, G\}\}$

- So *B* cannot obtain read privileges from any other process.
- Some process can (transitively) provide every process but *B* with read permissions.

Some Analyses
(1/4)

Let's take max confidentiality closures first:

- *read*(): $\{\{A, C, D, E, F, G, H, I\}, \{B\}\}$
- *write*(): $\{\{A, B, C, D, E, F, G, H\}, \{A, B, F, H, I\}\}$
- *read* – *write*(): $\{\{A, F, H, I\}, \{B\}, \{C, D, G\}, \{C, E, G\}\}$

- So *B* cannot obtain read privileges from any other process.
- Some process can (transitively) provide every process but *B* with read permissions.
- The 2nd *write*() closure is *I*'s confidentiality closure.

Some Analyses

(2/4)

Let's look specifically at D now:

- The 4-TC of D w.r.t. reading is $\{A, C, D, E, F, G, H\}$.

  So D can provide read privileges to every process but I within four steps.
- But the 1-TC of D w.r.t. reading is $\{D, G\}$.
  - So D can only reach G in one step.
  - So, if our threat model is that only one process gets compromised, and it's D, then at most D can maliciously provide read privileges to G.

Some Analyses

(3/4)

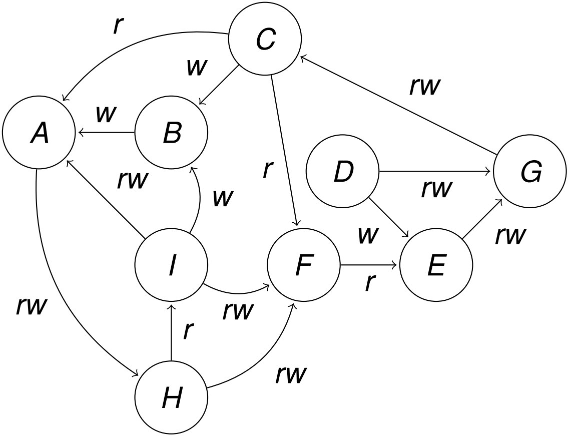
Let's stipulate the following separation policy:

- If a process x can provide another process y read permissions on a file, then y cannot provide x with any write permissions:
- For edge $x \xrightarrow{r} y$, there is a corresponding $x \overset{w}{\nrightarrow} y$, where $x \neq y$.

Let's apply the separation algorithms:

Some Analyses

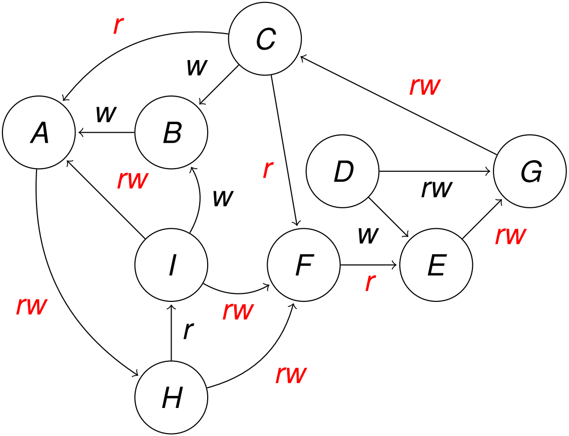
(4/4)



- Sink-separation and source-separation: $A \xrightarrow{rw} H$, $E \xrightarrow{rw} G$, $G \xrightarrow{rw} C$, $H \xrightarrow{rw} F$, $I \xrightarrow{rw} A$, $I \xrightarrow{rw} F$, $C \xrightarrow{r} A$, $C \xrightarrow{r} F$, $F \xrightarrow{r} E$, and $H \xrightarrow{r} I$.
- Greedy hitting set: $A \xrightarrow{rw} H$ and $E \xrightarrow{rw} G$.

Some Analyses

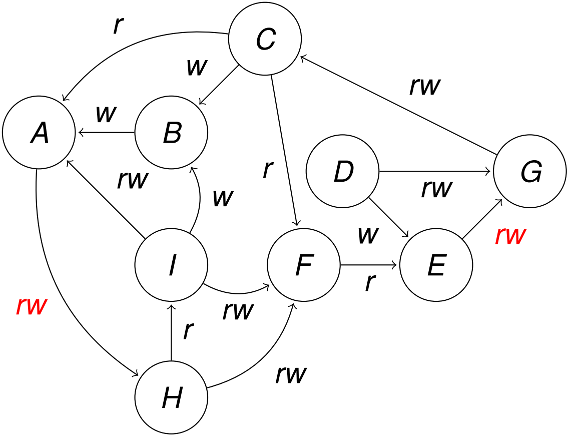
(4/4)



- Sink-separation and source-separation: $A \xrightarrow{rw} H$, $E \xrightarrow{rw} G$, $G \xrightarrow{rw} C$, $H \xrightarrow{rw} F$, $I \xrightarrow{rw} A$, $I \xrightarrow{rw} F$, $C \xrightarrow{r} A$, $C \xrightarrow{r} F$, $F \xrightarrow{r} E$, and $H \xrightarrow{r} I$.
- Greedy hitting set: $A \xrightarrow{rw} H$ and $E \xrightarrow{rw} G$.

Some Analyses

(4/4)



- Sink-separation and source-separation: $A \xrightarrow{rw} H$, $E \xrightarrow{rw} G$, $G \xrightarrow{rw} C$, $H \xrightarrow{rw} F$, $I \xrightarrow{rw} A$, $I \xrightarrow{rw} F$, $C \xrightarrow{r} A$, $C \xrightarrow{r} F$, $F \xrightarrow{r} E$, and $H \xrightarrow{r} I$.
- Greedy hitting set: $A \xrightarrow{rw} H$ and $E \xrightarrow{rw} G$.

Conclusions

- We've presented a simple model of *allowed* information flow to help in system analysis.
  - Useful, sometimes with surprising analyses.
  - Many ideas from the folklore but I haven't seen them collected.
- We've described simple confidentiality and integrity analyses.
- We've described algorithms for analyzing separation policies in our model.
- Future work:
  - Automatic translation from access control policies, linkers, etc., and visualization of abstractions.
  - Heuristics/advice for system designers: "Perhaps you can separate components A and B?"
- Tool: `http://www.cs.indiana.edu/~lepike/pubpages/infoflow.html` or google $\boxed{\text{post-hoc lee pike}}$