A Reference Monitor for Workflow Systems with Constrained Task Execution

Jason Crampton
Information Security Group, Royal Holloway, University of London

10th ACM Symposium on Access Control Models and Technologies
A purchase order workflow system

A representation of an organizational or business process

Typically specified as a set of tasks and a set of dependencies between the tasks

Users of the system are authorized to perform certain tasks
A constrained purchase order workflow system

It may be necessary to impose constraints on who can perform a task given that a prior task has been performed by a particular individual. These constraints may exist to prevent fraud or to implement a particular feature of a business process.
The problem defined

Design an abstract machine (reference monitor) that decides whether a user request to execute a task in a workflow instance should be granted.

Necessary conditions for the request to be granted are that

- the user has appropriate authorization
- no constraint is violated

In addition, we require that the workflow instance remains satisfiable.
A trivial example

Consider the following simple situation

- Two users $a$ and $b$
- $a$ and $b$ are authorized to perform $t_1$
- $a$ is authorized to perform $t_2$
- It is required that different users perform $t_1$ and $t_2$

The reference monitor must prevent $a$ from performing $t_1$ if the workflow is to complete and all constraints are to be satisfied.
Outline of talk

• Basic concepts and definitions
• Linear extensions and execution assignments
• Simplifying constraints
• Building a reference monitor for workflow systems
• Conclusion
Basic definitions

A workflow specification is a partially ordered set of tasks $T$

- If $t < t'$ then $t$ must be performed before $t'$ in any instance of the workflow

A workflow authorization schema is a pair $(T, A)$, where $A \subseteq T \times U$

- If $(t, u) \in A$ then $u$ is authorized to perform $t$ in any instance of the workflow

- Typically $A$ will not be given explicitly and will be derived from other information (role assignments for example)
Entailment constraints

An entailment constraint has the form \((D, (t, t'), \rho)\), where
\(\rho \subseteq U \times U\) and \(t \not> t'\)

- \(D \subseteq U\) is called the domain of the constraint
- \(t\) must not follow \(t'\) in the workflow specification
- \(\rho\) is used to determine which users can perform \(t'\) once \(t\) has been performed by a user in \(D\)
  - If \(u \in D\) performs \(t\) and \(u' \in U\) performs \(t'\), then \((D, (t, t'), \rho)\)
    is satisfied iff \((u, u') \in \rho\)
Examples

Separation of duty

Binding of duty

Seniority constraint

“Restricted” separation of duty
**Constrained workflow authorization schema**

A *constrained workflow authorization schema* (CWAS) is a triple \((T, A, C)\), where \(C\) is a set of entailment constraints.

A CWAS is *well-formed* if all constraints on non-comparable tasks are symmetric.
**Linear extensions**

A *linear extension* of a workflow specification $T$ represents a possible sequence of execution of the tasks in $T$

$$[\text{createPO}, \text{apprPO}, \text{createPay}, \text{signGRN}, \text{ctrsignGRN}, \text{apprPay}]$$

There may be many different linear extensions for a workflow specification
**Execution assignments**

A *valid execution assignment* for a CWAS \((T, A, C)\) is a pair \((L, \alpha)\), where

- \(L\) is a linear extension of \(T\)
- \(\alpha : T \rightarrow U\) assigns users to tasks

such that

- each user is appropriately authorized
- all constraints are satisfied

A CWAS is *satisfiable* if there exists a valid execution assignment.
An important result

Let \([t_1, \ldots, t_n]\) be a linear extension and let \([(t_1, u_1), \ldots, (t_n, u_n)]\) be a valid execution assignment.

If \(\pi\) is a permutation of 1, \ldots, \(n\) such that \([t_{\pi(1)}, \ldots, t_{\pi(n)}]\) is a linear extension then \([(t_{\pi(1)}, u_{\pi(1)}), \ldots, t_{\pi(n)}, u_{\pi(n)})]\) is a valid execution assignment.

It is sufficient to consider a fixed linear extension to establish whether \(W\) is satisfiable.
Simplifying constraints (1)

Before

\[(D, \rho)\]

\[t \rightarrow t'\]

\[\rho\]

\[t \rightarrow t' \quad \rho'\]

\[\rho \rightarrow \rho'\]

\[t \rightarrow t' \rightarrow t''\]

After

\[(U, \rho \cup \{(u,v) : u \notin D, v \in U\})\]

\[t \rightarrow t'\]

\[\rho \cap \rho'\]

\[t \rightarrow t'\]

\[{(u, u'') : (u, u') \in \rho, (u', u'') \in \rho'}\]

\[t \rightarrow t''\]
Simplifying constraints (2)

Each task $t$ is associated with a set of authorized users $V(t)$

For each pair of tasks $(t_i, t_j)$ we can derive an “authorization constraint” $(U, (t_i, t_j), \rho)$ where $\rho_{ij} = V(t_i) \times V(t_j)$
The satisfiability algorithm

Inputs: canonical linear extension \([t_1, \ldots, t_n]\)  
user authorization information  
set of constraints \(C\)

```
01 for i = 1 to n
02 let V(i) = set of users authorized to perform task i
03 for i = 1 to n
04 for j = i to n
05 if ((t_i, t_j), rel) \in C
06 let V(i,j) = (V(i) \times V(j)) \cap rel
07 if V(i,j) = \emptyset then return false
08 let V(i) = set of users in first position of V(i,j)
09 let V(j) = set of users in second position of V(i,j)
10 return true
```
An important observation

A partially completed instance of a CWAS is itself a CWAS in which completed tasks have a single “authorized” user

- Let $I$ be an instance of $(T, A, C)$, where $I$ is a prefix of some linear extension of $T$
- Let $I(t)$ denote the user who performed task $t$ in $I$
- Then $(T, A|I, C)$ is a CWAS, where

$$A|I = \{(t, I(t)) : t \in I\} \cup \{(t, u) : t \notin I\}$$
The main results

The same decision procedure can be used to answer the questions

- Is a CWAS satisfiable?
- Is an instance of a CWAS satisfiable?

A reference monitor that guarantees every workflow instance completes can simply implement the satisfiability algorithm to check that

- If the request were to be granted, the resulting CWAS is satisfiable

The reference monitor runs in time

- $O(|U|^2|T|^2)$ if we only have separation of duty and binding of duty constraints
- $O(|U|^4|T|^2)$ otherwise
Extensions to the basic model

In the paper we consider workflow specifications in which

- A task may be repeated (arbitrarily often or a fixed number of times)
- Constraints exist on repeated instances of tasks
- Tasks may be omitted from certain workflow instances

Workflow specification is treated as labelled, acyclic, directed graph
Design of reference monitor is similar (has to account for multiple task execution)
Complexity results are the same
Why is this interesting? (1)

Bertino, Ferrari and Atluri described a reference monitor in their seminal paper “The specification and enforcement of authorization constraints in workflow management systems” (TISSEC 1999)

The computational complexity of their algorithm is exponential in the number of tasks and users

- It pre-computes and searches the space of all valid execution assignments to determine the validity of a request

Our algorithm is polynomial in the number of users and tasks

- It only tests for the existence of a valid execution assignment
Why is this interesting? (2)

Independent of underlying computational model and access control model

- Has generality that other approaches lack
- Can be implemented in a variety of different ways
- Uniform treatment of constraints

Simple and rigorous

- Design of reference monitor is simpler
- Overall understanding of mechanisms is improved
- Computational complexity of reference monitor and storage requirements are significantly reduced

Far wider range of constraints can be specified
Future work

Main priority is to develop model to provide more sophisticated treatment of the interaction between cardinality constraints and entailment constraints

Consider how changes to the authorization data over the lifetime of a workflow instance might affect the operation of the reference monitor

Consider developing our workflow model to include primitive operations associated with each task (begin, commit, abort, etc.)
Questions?