Modelling and verifying access control policies for web-based collaborative systems

Masoud Koleini, Hasan Qunoo, Mark Ryan

School of Computer Science
University of Birmingham
18th Nov 2009
Introduction

There is an ever increasing use of web-based systems for managing collaborative work. Systems like:

Modern access control systems complexity makes reasoning about them by hand infeasible.
Motivation

Consider a conference review system with the following policy:

- **PC chair** can assign **PC members** to review a paper
- **PC members** can assign **sub-review** to a paper that is assigned to them
- **Sub-reviewers** send their reviews to the reviewer
- Once the **reviewer** receives the paper review from the **subreviewer**, the **reviewer** can submit the review to the system
A possible vulnerability in the system
What makes stateful systems vulnerable?

- Interactions between the rules
- Co-operations between agents
- Multi-step transactions
The need for an expressive access control policy language

Given an access control policy model $M$, can a set of agents $A$ achieve the goal $\Phi$?

- The modelling language must have a clear formalism that is expressive enough to model arbitrary access control policies
- Query language must be expressive enough to allow complex and nested goals
- We need appropriate verification methods and analysis techniques which are able to search for strategies that achieve the goal
We propose a modelling language and verification tool, called X-Policy. It offers us the ability to:

- Model atomic transactions that can update several variables in synchrony
- Express complex execution permissions for each transaction
- Find attack strategies using model checking
- Reason about agents knowledge of the system
X-Policy examples

Program RequestReviewing (p:Paper, a:Agent, b:Agent):
{
  Requested-subreviewing(p,a,b):= T;
  Decided-subreviewing(p,a,b):=F;
}

Program ShowReview (p:Paper, a:Agent):
{
  return Submitted-review(p, a);
}
Program execution permission in X-Policy

- We use the program permission statement \( \text{exec}(g,u) \) to define the conditions for an agent \( u \) to execute a program \( g \).

\[
\text{exec}(\text{ShowReview}(p, a, b), u) \iff \left( \begin{array}{c}
\text{Chair}(u) \\
\land \text{Chair-review-menu-enabled}() \\
\land \exists d : \text{Agent.Author}(p, d) \\
\lor \ldots \\
\lor \ldots 
\end{array} \right)
\]
Access control model verification

We now can specify different properties like:

- $\Phi = \exists p : \text{Paper}, \exists a : \text{Agent} \ (\text{author}(p, a) \land \text{reviewer}(p, a))$

- $\Phi = \exists p : \text{Paper}, \exists a, b, c : \text{Agent} \ (\text{subreviewer}(p, a, c) \land \text{subreviewer}(p, b, c))$

The model checking tool will run a backward reachability algorithm to check whether the property hold and it outputs a strategy in case the model satisfies the property.
Model abstraction

Model checkers suffer from the state explosion problem when the number of propositions grows. Abstraction helps us by reducing the state space.

- We use a CEGAR\([1]\) based variable-hiding abstraction and refinement technique and build from \(M\) an abstract model \(M'\) such that:

\[
M' \models \phi \quad \text{implies} \quad M \models \phi
\]

where \(\phi\) is an ACTL* specification formula.

Abstraction refinement

- We reduce the number of propositions by a specific variable hiding abstraction.
- If the specification cannot be satisfied in the abstract model, it will not be satisfied in the concrete model.
- If the specification get satisfied in the abstract model, the strategy found should be checked over the concrete model. If it is a spurious strategy, abstract model should get refined.
- Using a special algorithm to rank the propositions and put concretise them in the refinement process according to their rank.
Verification process

Initial Abstract Model $M'$

Verify $M'$

Does strategy satisfies in $M$?

Concretise some of the hided propositions

No

Yes

Output: strategy

Is $\Phi$ satisfied?

Yes, Strategy

Output: strategy

No

Output: $\Phi$ cannot be satisfied in $M$

End
Future work

- We are planning to implement the model checking algorithm for X-Policy model.

- We are working on developing the abstraction and refinement technique as discussed.
Questions

Your comments would be much appreciated