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:

EasyChair conference system

Google docs

Create and share your work online

- **Upload** from and save to your desktop
- **Edit** anytime, from anywhere
- Pick who can **access** your documents
- **Share** changes in real time
- Files are stored **securely** online
- It's **free**!

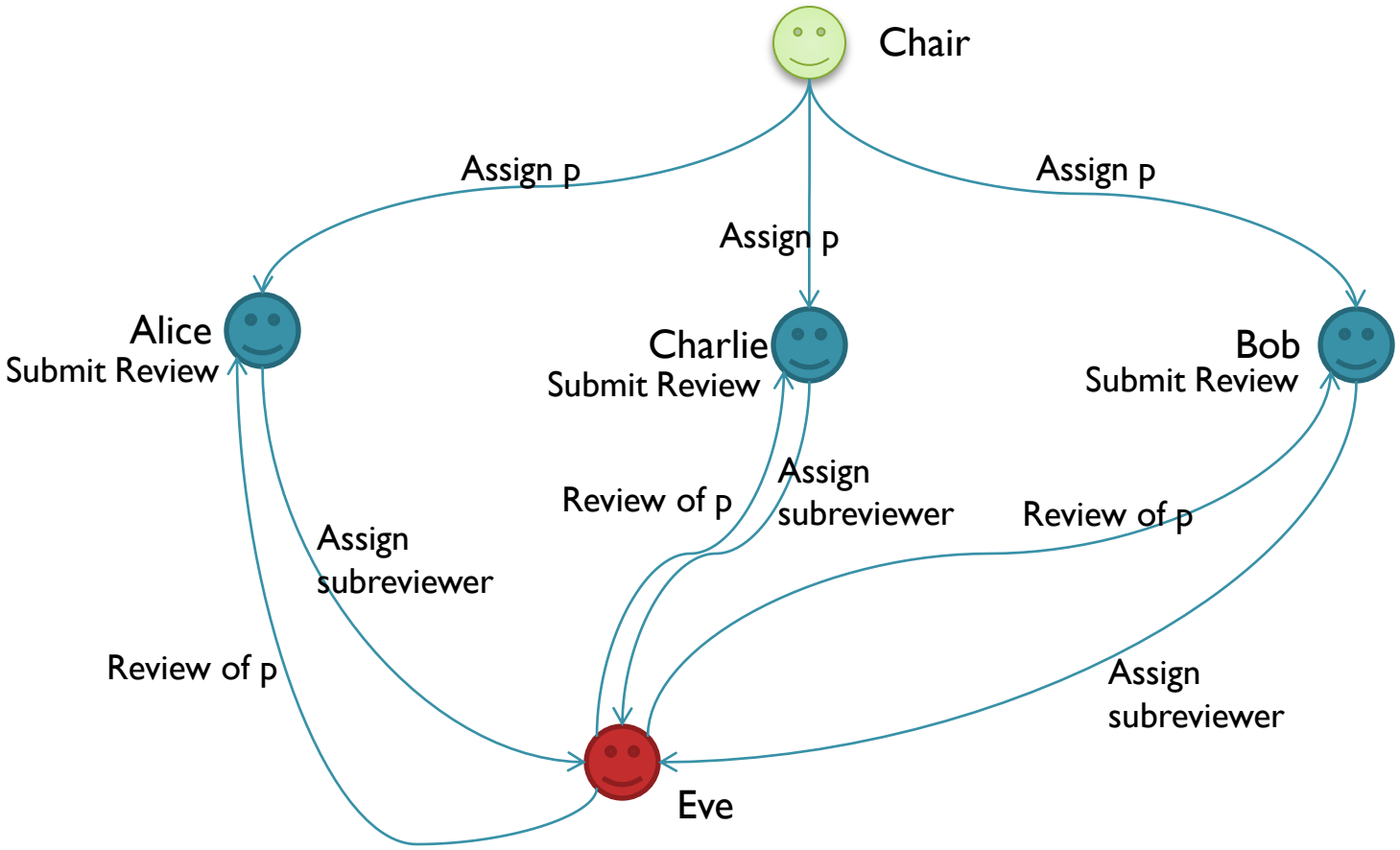
Modern access control systems complexity makes reasoning about them by hand infeasible.

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 Φ?

- 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

X-Policy modelling language and verification

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

Executes when *a* clicks on "RequestReviewing" button to assign *p* to *b*

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

Executes when an agent clicks on "ShowReview" button to read the review of paper *p* submitted by *a*

```
Program ShowReview (p:Paper, a:Agent):-
{
        return Submitted-review(p, a);
}
```

Program execution permission in X-Policy

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

$$\text{exec}(\text{ShowReview}(p, a, b), u) \rightleftharpoons \left(\begin{array}{c} \left(\begin{array}{c} \text{Chair}(u) \\ \wedge \text{Chair-review-menu-enabled}() \\ \wedge \exists d : \text{Agent}.\text{Author}(p, d) \end{array} \right) \\ \vee \ldots \\ \vee \ldots \end{array} \right)$$

Access control model verification

We now can specify different properties like:

- $\Phi = \exists p : Paper, \exists a : Agent \ (author(p, a) \wedge reviewer(p, a))$
- $\Phi = \exists p : Paper, \exists a, b, c : Agent \ (subreviewer(p, a, c) \wedge subreviewer(p, b, c))$

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:

$$M' \models \phi \quad \text{implies} \quad M \models \phi$$

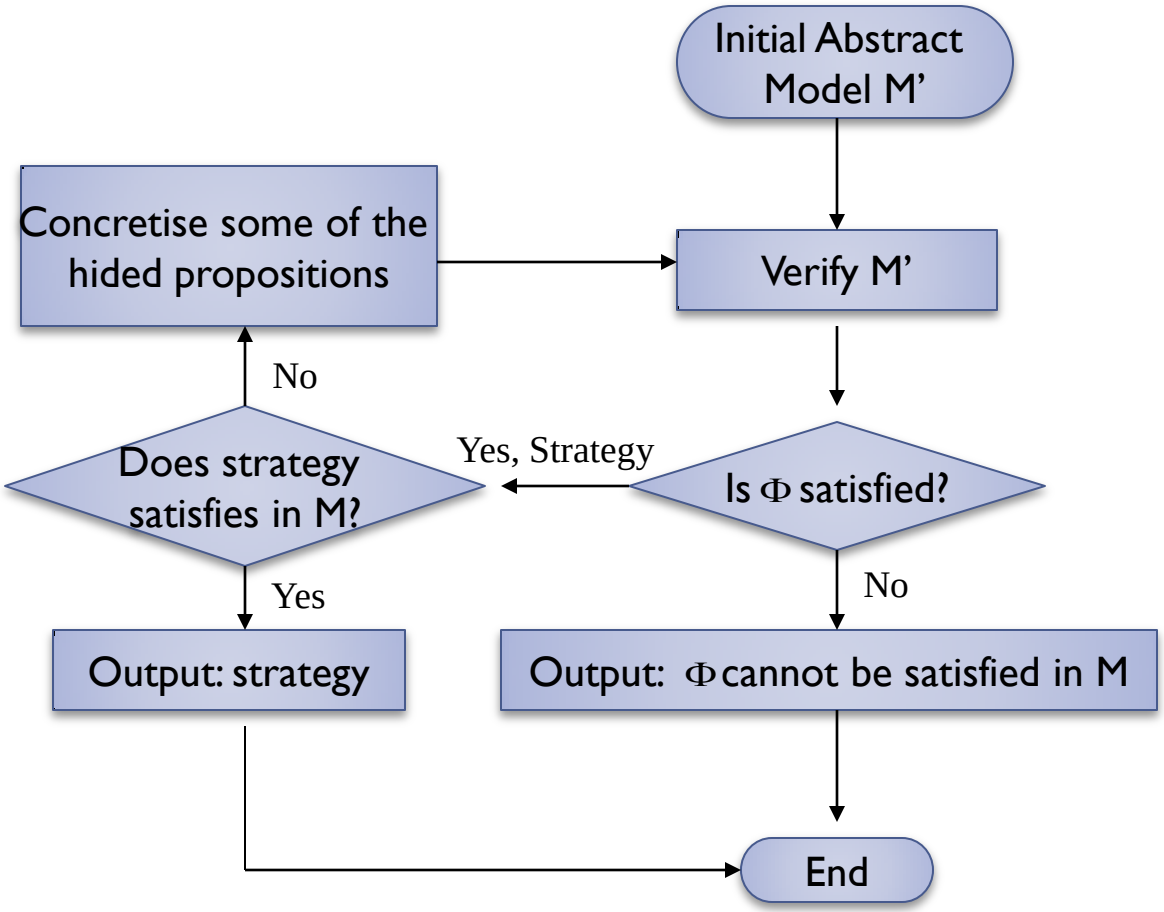
where ϕ is an ACTL* specification formula.

[1]-Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, Helmut Veith, "Counterexample-guided abstraction refinement for symbolic model checking", Journal of the ACM, Vol. 50, No. 5. (01 September 2003), pp. 752-794.

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



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