Designing Security Requirements Through Planning

Volha Bryl, Fabio Massacci, John Mylopoulos, Nicola Zannone

University of Trento, Italy
Outline

- Introduction
- Modeling with Secure Tropos
- Extending Secure Tropos Framework
  - Designing secure systems through planning
- Conclusions and Future Work
Introduction: What we are doing

- **Problem**: design of secure and trusted systems.
- **Existing solutions**: refined SE methodologies, incl. tools for **automation** support.
- **Which kind of automation the designer needs?**

> “Exploring **alternative** options is at the heart of the requirements and design processes”

- **Objective**: provide support for exploring the space of design alternatives.
Introduction: Motivation

- Design alternatives are *potential choices* designer may adopt.
- Current Software Engineering methodologies support reporting and verifying *final choices*, but not exploring potential ones.

- Other proposals for automation
  - Deductive program synthesis
    - Theorem proving: a system goal from the axioms
  - Model-Driven Architecture
    - (Automatic) model transformation
  - Design patterns
Secure Tropos

- Requirements Engineering methodology
  - which extends the Tropos methodology,
  - allows to model and analyze functional and security requirements.

- Secure Tropos concepts
  - Actor, Goal, Softgoal, Task, and Resource
  - Social relationships
    - Requesting, Ownership, Provisioning
    - Trust for Permission/Execution
    - Delegation of Permission/Execution
Example: Medical Care System

A fragment of Secure Tropos model

Tp – trust for permission
Te – trust for delegation
R – requesting
O – ownership
P – provisioning
Example: Design Alternatives

(a) Potential choice: trust

(b) Potential choice: or-decomposition

(a) Final choice: delegation

(b) Final choice: subgoal chosen
In Secure Tropos requirements are conceived as networks of delegations (of permission/execution) among actors for goals, tasks and resources.

The task of designing such networks can be framed as a planning problem

- selecting a suitable design corresponds to selecting a plan that satisfies the goals of human or system actors.
AI Planning

- Planning approach
  - automatically determine the course of actions (a plan) to achieve a desired state (a goal), where an action is a transition rule from one system state to another.

- Planning problem is defined by
  - domain description: predicates, actions, axioms
  - problem description: initial and desired state
## Predicates

<table>
<thead>
<tr>
<th>Goal Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>and_decomposition_n(g : goal; g1 : goal, … , gn : goal)</td>
</tr>
<tr>
<td>or_decomposition_n(g : goal; g1 : goal, … , gn : goal)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Actor Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>provides(a : actor, g : goal)</td>
</tr>
<tr>
<td>requests(a : actor, g : goal)</td>
</tr>
<tr>
<td>owns(a : actor, g : goal)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Actor Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>trustexe(a : actor, b : actor, g : goal)</td>
</tr>
<tr>
<td>trustper(a : actor, b : actor, g : goal)</td>
</tr>
</tbody>
</table>
Actions

Defined in terms of **preconditions** and **effects** using above defined **predicates**

- **Satisfy** ($a : actor, g : goal$)
- **DelegateExecution** ($a : actor, b : actor, g : goal$)
- **DelegatePermission** ($a : actor, b : actor, g : goal$)
- **AND_Refine_n** ($a : actor, g : goal, g1 : goal, \ldots, gn : goal$)
- **OR_Refine_n** ($a : actor, g : goal, g1 : goal, \ldots, gn : goal$)
Example: Absence of Trust

Fully trusted domains are unrealistic.
Absence of Trust

How to delegate execution with no trust?
- Establish trust (create a contract) or
- Control the execution.

Our solution combines contract & control
- Additional predicates and actions:
  \[\text{DelegateExecution} \rightarrow \text{Satisfy}\]
  is replaced with
  \[\text{Negotiate} \rightarrow \text{Contract} \rightarrow \text{DelegateExecution}\_\text{under}\_\text{suspicion} \rightarrow \text{Fulfill} \rightarrow \text{Evaluate}\]
Implementation

- Looking for off-the-shelf tool, requirements
  - should produce non-redundant plans
  - Need-to-Know property: no alternative delegation paths
  - should support PDDL 2.2
  - should be platform-independent

- Chosen planner: LPG-td
  - Does not produce optimal plans

- Experiments
  - Preliminary testing and the case study
  - Scalability tests
Example: Problem Definition

: objects
   Pat HCA HP INPDAP INPDAI - actor
   ProvideMC ProvisioningMC PaymentMC - goal
   PaymentTicket PaymentHCA - goal
   PaymentTicketHP PaymentTicketINPDAP - goal
   PaymentFullCost Reimbursement - goal
   CollectionINPDAI CollectionHP - goal
   ReimbursementINPDAI ReimbursementHP - goal

: goal
   (done ProvideMC)

: init
   (owns HCA ProvideMC)
   (requests Pat ProvideMC)
   (provides HP ProvisioningMC)
   (provides HCA PaymentHCA)
   (provides INPDAP PaymentTicketINPDAP)
   ...
   (trustexe Pat HP ProvideMC)
   (trustper HCA HP ProvisioningMC)
   (trustexe HP HCA PaymentMC)
   (trustexe HCA INPDAP PaymentMC)
   (trustper HCA INPDAI ReimbursementINPDAI)
   (trustexe INPDAI HP ReimbursementHP)
   ...
   (AND decomposition2 ProvideMC ProvisioningMC PaymentMC)
   (AND decomposition2 PaymentMC PaymentTicket PaymentHCA)
   (OR decomposition2 PaymentFullCost CollectionINPDAI CollectionHP)
   (OR decomposition2 Reimbursement ReimbursementINPDAI ReimbursementHP)
   ...
Example: Solution

DelegateExecution Pat HP ProvideMC
AND_Refine HP ProvideMC ProvisioningMC PaymentMC
DelegatePermission HCA HP ProvisioningMC
Satisfy HP ProvisioningMC
DelegateExecution HCA INPDAP PaymentMC
AND_Refine INPDAP PaymentMC PaymentTicket PaymentHCA
DelegateExecution HCA INPDAP PaymentHCA
Satisfy HCA PaymentHCA
OR_Refine INPDAP PaymentTicket PaymentTicketINPDAP PaymentTicketHP
DelegatePermission HCA INPDAP PaymentTicketINPDAP
Satisfy INPDAP PaymentTicketINPDAP
Example: Solution
Conclusions

- **Objective**: provide support for exploring the space of design alternatives
  - Important design activity
  - Not addressed in current SE methodologies

- **Result**: we have proposed the way to automate the selection of design alternatives
  - In context of Secure Tropos
  - Through viewing the design problem as a planning one

- **Future work**
  - Elaborate on the planning part (e.g. consider costs of actions)
  - Try larger industrial case studies
  - Perform thorough scalability testing
Acknowledgements

We thank Alfonso Gerevini and Alessandro Saetti for the support on the use of LPG-td planner.

This work has been partially funded by SERENITY and SENSORIA EU projects, FIRB program of MIUR under ASTRO project, and by the Provincial Authority of Trentino, through the MOSTRO project.
Thank you. Questions?