Access Control Policy Tool (ACPT)

**ACPT:**

*Access Control Policy Tool*
Presently Policy authoring are hand crafted by administrators, and difficult to check for correctness, we need tool for:

- Composing policy by structure framework
- Detecting conflicts in policy rules
- Efficient testing of implementation
- Policy code generation
Outline

Access Control Policy Tool (ACPT) Overview

Approaches
- Model specification and composition
- Property verification
- Policy testing
- XACML generation

Related work

Future work
Composition
Allows specification of policy combinations, rules and properties through model and rule templates.

Verification
Allows testing and verification of policies against specified properties and reports problems that may lead to security holes.

Testing
Generates efficient test suites (by applying NIST’s combinatorial testing technology) for testing of access control implementation, test suites can be applied to any access control implementation.

Policy
XACML policy generation.
ACPT Overview - Architecture

GUI allows specification of users, user attributes, actions, resources, resources attributes, and properties

Templates include: ABAC, Workflow, and Multi-Level

Validates models against properties

Generates encoded Model

Generates test suites

Generates combinatorial test array

Access Control Policy Tool

AC policy author

GUI

AC Model Templates

Model Checker

Model Generator

Combinatorial Array Generator

Test Suite Generator

XACML policy instance

Test suite

Generates encoded .xml

XACML policy instance

Test suite
Allow to conveniently specify mandatory AC models (as well as AC rules) through pre-defined model templates

- Allow to create various models by specifying attribute values e.g., role subjects, resources, and actions for RBAC, user and resources ranks for MLS.

- Combine different AC models or rules into a composed one e.g., combine RBAC with multi-level models.

- allow to configure model priority for combining models or rules.
Approaches: AC Model Specification and Composition - Example
Conflicts among policy entities and their complexity may leak unauthorized or prohibit authorized access privileges.

- Convert composed models and user-specified properties to input models and properties for a model checker (e.g., NuSMV).

- Verify models against specified properties, and report detected property violations.
ACPT uses the NuSMV model checker, a well-structured, flexible, and efficient tool (supporting CTL and LTL model checking)
Approaches: Property Verification - Example

Property specification in ACPT
Test the property against Policy A **combined** with Policy B. Combined policies has the priorities of the combined rules. This slide shows the combination of policies, where Policy B has higher priority than policy A.
Test the property against Policy B, the result return *true*.
Test the property against Policy A, the result return *false* with counterexample.
Assure correct policy implementations by

- Test Generation: Generate *test requests*.
- Test Execution: Evaluate test requests (against policy implementations) and produce their decisions.
- Test-Result Evaluation: Check if the decisions are consistent with expected decisions (from properties or manual inspection, etc.).
  - If inconsistent, implementation faults are revealed.
Exhaustive testing is impractical (esp. for large number of AC entities).

Generating efficient and effective test suites (from AC models) using Combinatorial Array Generation Technology.

Generated test suites can be applied to any access control implementations in practice to find implementation faults.
Collect domain variables in AC models and generate **efficient** test suite automatically to detect faults using NIST combinatorial testing tool (ACTS)

- inputs: a domain of variables
- outputs: t-way covering arrays as tests

For example, domain of variables:

- 2 subjects: Faculty and Student
- 2 actions: write and view
- 2 resources: grades and records

Given the domain, 4 and 8 tests are generated for 2-way and 3-way interactions, respectively

- `<Faculty, grades, write>`, `<Faculty, records, view>`, …
Approaches: Policy Testing - Combinatorial Test cont.

• Combinatorial tests based on 2-way interactions

<table>
<thead>
<tr>
<th>SUBJECTS</th>
<th>RESOURCES</th>
<th>ACTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Faculty</td>
<td>grades</td>
</tr>
<tr>
<td>2</td>
<td>Faculty</td>
<td>records</td>
</tr>
<tr>
<td>3</td>
<td>Student</td>
<td>grades</td>
</tr>
<tr>
<td>4</td>
<td>Student</td>
<td>records</td>
</tr>
</tbody>
</table>

• Combinatorial tests based on 3-way interactions (being exhaustive tests)

<table>
<thead>
<tr>
<th>SUBJECTS</th>
<th>RESOURCES</th>
<th>ACTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Faculty</td>
<td>grades</td>
</tr>
<tr>
<td>2</td>
<td>Faculty</td>
<td>grades</td>
</tr>
<tr>
<td>3</td>
<td>Faculty</td>
<td>records</td>
</tr>
<tr>
<td>4</td>
<td>Faculty</td>
<td>records</td>
</tr>
<tr>
<td>5</td>
<td>Student</td>
<td>grades</td>
</tr>
<tr>
<td>6</td>
<td>Student</td>
<td>grades</td>
</tr>
<tr>
<td>7</td>
<td>Student</td>
<td>records</td>
</tr>
<tr>
<td>8</td>
<td>Student</td>
<td>records</td>
</tr>
</tbody>
</table>
Approaches: Policy Testing – Example

Test cases generation:

1: (Government_Category = Federal) & (CFR_Part_23_Training = Current) & (Assurance_Level = 2) & (Remote_Access = True) & (Privacy_Category = ISE) & (read = True) -> decision = Permit

2: (Government_Category = Federal) & (CFR_Part_23_Training = Current) & (Assurance_Level = 2) & (Remote_Access = False) & (Privacy_Category = SL7) & (read = False) -> decision = Deny

3: (Government_Category = Federal) & (CFR_Part_23_Training = Expired_None) & (Assurance_Level = 2) & (Remote_Access = True) & (Privacy_Category = SL7) & (read = True) -> decision = Deny

4: (Government_Category = Federal) & (CFR_Part_23_Training = Expired_None) & (Assurance_Level = 2) & (Remote_Access = False) & (Privacy_Category = ISE) & (read = False) -> decision = Deny

5: (Government_Category = State) & (CFR_Part_23_Training =...
Generate XACML policy based on the verified (combined or individual) models and rules.
XACML generation:

```
<Subject>
  <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchemaString">urn:oa
    <SubjectAttributeDesignator SubjectCategory="urn:oa"
    </SubjectMatch>
  </Subject>
</Subject>

<Subject>
  <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchemaString">urn:oa
    <SubjectAttributeDesignator SubjectCategory="urn:oa"
    </SubjectMatch>
  </Subject>
</Subject>

<Resources>
  <Resource>
    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchemaString">Privoc
      <ResourceAttributeDesignator AttributeId="Privoc"
      </ResourceMatch>
  </Resource>
</Resources>
```

run NuSMV verification....
It would takes time for generating Test oracles....
Test oracle creation is finished....

Approaches: XACML Generation – Example cont.

<PolicySet PolicySetId="n" PolicyCombiningAlgId="First-Applicable">
  <Target/>
  <Policy PolicyId="RBAC_school" RuleCombinationAlgId="First-Applicable">
    <Target/>
    <Rule RuleId="1" Effect="Deny">
      <Target>
        <Subjects><Subject>Student</Subject><Subject>Secretary</Subject></Subjects>
        <Resources><Resource>Grades</Resource></Resources>
        <Actions><Action>Change</Action></Actions>
      </Target>
    </Rule>
    <Rule RuleId="2" Effect="Permit">
      <Target>
        <Subjects><Subject>Professor</Subject><Subject>Lecturer</Subject><Subject>Secretary</Subject></Subjects>
        <Resources><Resource>Grades</Resource><Resource>Records</Resource></Resources>
        <Actions><Action>Change</Action><Action>Read</Action></Actions>
      </Target>
    </Rule>
  </Policy>
  <Policy PolicyId="ABAC_school" RuleCombinationAlgId="First-Applicable">
    <Target/>
    <Rule RuleId="3" Effect="Permit">
      <Target>
        <Subjects><Subject>Jim</Subject></Subjects>
        <Resources><Resource>Records</Resource><Resource>Grades</Resource></Resources>
        <Actions><Action>Change</Action></Actions>
      </Target>
    </Rule>
  </Policy>
</PolicySet>

Rule 1:
A student or secretary can not change grades.

Rule 2:
A professor, lecturer, or secretary can change grades or records.

Rule 3:
Jim can change grades or records.
Related Work: Compare with Commercial AC Tools

A commercial AC policy management tool does not have all the following capabilities that NIST ACPT has:

• **AC model templates** for specifying models/policies: ABAC, Multi-Level, and Workflow.

• **Composition of multiple AC models** into a composed one, e.g., combine RBAC with MLS models.

• **AC property verification** to detect faults in models/policies. Some have only limited SOD (Separation of Duty) check.

• **Test-suite generation** for testing AC implementations in real operation environment to detect faults in implementations.
Future Work

• Available soon after final Alpha test.
• Enhance capabilities:
  -- White-box model/properties verification to verify coverage and confinement of AC rules.

  -- Additional AC policy templates including dynamic and historical access control models.

  -- API or mechanism for acquiring or consuming information about users, attributes, resources, etc.
Questions?

vhu@nist.gov