Coverage-Based Testing of Obligations in NGAC Systems

Erzhuo Chen, Vladislav Dubrovenski, Dianxiang Xu
University of Missouri – Kansas City
{vadpb7, lcrnd, dxu} @umsystem.edu
Background

- **NGAC** (Next Generation Access Control), a new *access control* standard
- Proposed by NIST (National Institute of Standards and Technology)
- Designed to **address limitations**: limited flexibility, difficulty in managing policies, and limited interoperability
Configuration $C = \langle U, UA, O, OA, AR, PC, ASSIGN, ASSOCIATION, PROHIBITION \rangle$:

- $U$: set of users
- $UA$: set of user attributes
- $O$: set of objects
- $OA$: set of object attributes
- $AR$: set of all access rights
- $PC$: set of policy classes
- $ASSIGN$ & $ASSOCIATION$ & $PROHIBITION$: three sets of relations defined on policy elements
Obligation

\[
\text{<event pattern>} ::= \text{[<user spec>] [<pc sepc>] <op spec> [<pe spec>]}
\]
\[
\text{<response>} ::= \text{<response condition>} <\text{conditional action}>\{,<\text{conditional action}>\}
\]
\[
\text{<response condition>} ::= \text{[if} \text{<condition>} \text{then]}
\]
\[
\text{<conditional action>} ::= \text{[if} \text{<condition>} \text{then]} \text{<action>}\{,<\text{action}>\}
\]
\[
\text{<condition>} ::= \text{<factor>}\{\text{and} <\text{factor}>\}
\]

SAMPLE

Obligation $\phi$

Event: \langle supervisor, delete, alex\rangle

Response:
If alex exists then

Actions:
If $\neg$alex.Loan $\land$ supervisor.OfficeHour then
delete \langle alex, accounts\rangle
If $\neg$\langle alex, accounts\rangle then delete object alex
Problem Statement

• NGAC is highly expressive and flexible, enabling creating complex access control policies. Additionally, it allows for dynamic changes to policies.

• However, there is a lack of work on quality assurance of NGAC policies. Meanwhile, the dynamic privilege changes through obligations come with potential concerns about errors and harm to the authorization state, leading to unauthorized access, privilege escalation, and denial of service.

• My research aims to investigate methods for ensuring the quality of obligations.
Approach

Coverage-based test generation method:

1. Define a family of coverage criteria

2. Generate constraints for satisfying coverage criterion

3. Solve constraints by a SMT-based solver

4. Translate the solution into tests
Obligation Test

Format of an obligation test:

- Test input:
  A sequence of access requests, \( \{q_1, q_2, \ldots, q_n\} \).

- Test oracle:
  Expected configuration changes, \( \{O_1, O_2, \ldots, O_n\} \).

\[ t = \{\{q_1, O_1\}, \{q_2, O_2\}, \ldots, \{q_n, O_n\}\} \], where \( q_i \) represents access request and \( O_i \) represents the expected configuration changes after the permitted access \( q_n \) occurs.
Coverage Criteria

- **Obligation Coverage (OC):**
  - each obligation is triggered once
- **Action Coverage (AC):**
  - each action applies once
- **Decision Coverage (DC):**
  - each outcome (true/false) of decision is covered
- **Factor/Decision Coverage (FDC):**
  - each outcome (true/false) of factor combinations is covered
  - each factor independently affects the outcome

---

**Obligation $\phi$**

**Event:** $\langle \text{supervisor, delete, alex} \rangle$

**Response:**

If $\text{alex}$ exists then

**Actions:**

If $\neg \text{alex Loan} \land \text{supervisor.OfficeHour}$ then delete $\langle \text{alex, accounts} \rangle$

If $\neg \langle \text{alex, accounts} \rangle$ then delete object $\text{alex}$
Algorithm for generating OC tests

**Obligation \( \phi \)**

**Event:** \( \langle \text{supervisor, delete, alex} \rangle \)

**Response:**

If \( \text{alex} \) exists then

**Actions:**

If \( \neg\text{alex.Loan} \land \text{supervisor.OfficeHour} \) then delete \( \langle \text{alex, accounts} \rangle \)

If \( \neg \langle \text{alex, accounts} \rangle \) then delete object \( \text{alex} \)

```plaintext
Function name: GenerateTestForOC
Input: Policy \( P = (C_0, \Phi) \), \( C_0 \) is initial configuration, \( \Phi \) is initial obligation
Output: KnownSequences is a set of distinct event sequences

foreach \( \phi \) in \( \Phi \) do
  foreach sequence in KnownSequences do
    if event(\( \phi \)) in sequence then
      continue;
    newSeq \( \leftarrow \) seqFinder(\( P, \phi \));
    if newSeq = null then
      continue;
    update \( \leftarrow \) true;
    foreach sequence in KnownSequences do
      if isSequenceCovered(sequence, newSeq) then
        update \( \leftarrow \) false;
        break;
      else
        if isSequenceCovered(newSeq, sequence) then
          remove sequence from KnownSequences;
      if update then
        add newSeq to KnownSequences;
  return KnownSequences;
```
Algorithm for generating OC tests

Obligation $\phi$

Event: $\langle$supervisor, delete, alex$\rangle$

Response:
If alex exists then
   Actions:
   If $\neg$alex.Loan $\land$ supervisor.OfficeHour then
      delete $\langle$alex, accounts$\rangle$
   If $\neg$$\langle$alex, accounts$\rangle$ then delete object alex

newSeq = {q1,q2,q3,q4} and
an old seq = {q1,q3}
=> only keep {q1,q2,q3,q4}
Algorithm for generating AC tests (part)

```
1 foreach action in obligation.response do
2   if action.covered then
3     continue;
4   if currentConstraints = null then
5     action.covered ← true;
6     coveredCount++;
7     currentConstraints ← obC ∧ reC ∧ action.conC;
8     solution ← action.selfSolution;
9     solution.involvedActions ← solution.involvedActions ∪ action.index;
10   else
11     tmpConstraints ← currentConstraints ∧ action.conC;
12     tmpSolution ← solver(P,tmpConstraints);
13     if tmpSolution = null then
14       continue;
15     else
16       action.covered ← true;
17       coveredCount++;
18       currentConstraints ← tmpConstraints;
19       solution ← tmpSolution;
20       solution.involvedActions ← solution.involvedActions ∪ action.index;
```
### Evaluation: subject policies

<table>
<thead>
<tr>
<th></th>
<th>#PC</th>
<th>#UA</th>
<th>#OA</th>
<th>#ASM</th>
<th>#ASC</th>
<th>#PRO</th>
<th>#OBL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank</td>
<td>2</td>
<td>6</td>
<td>10</td>
<td>33</td>
<td>6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GPMS</td>
<td>4</td>
<td>34</td>
<td>27</td>
<td>91</td>
<td>8</td>
<td>-</td>
<td>19</td>
</tr>
</tbody>
</table>

**Bank:**

An access control system of the management structure of a bank system.

**GPMS:**

A web-based application that aims to automate the grant proposal approval workflow at an academic institution.
### Evaluation: obligation mutation operators

<table>
<thead>
<tr>
<th>No</th>
<th>Fault Type</th>
<th>Mutation Operator</th>
<th>No</th>
<th>Fault Type</th>
<th>Mutation Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Extra obligation</td>
<td>ROB Remove one OBligation</td>
<td>19</td>
<td>Wrong assignment descendant</td>
<td>CDA Change Descendant in Assign</td>
</tr>
<tr>
<td>2</td>
<td>Wrong subject</td>
<td>CES Change Event Subject</td>
<td>20</td>
<td>Wrong assignment direction</td>
<td>RDA Reverse Direction of Assign</td>
</tr>
<tr>
<td>3</td>
<td>Extra subject</td>
<td>RES Remove Event Subject</td>
<td>21</td>
<td>Wrong grant subject</td>
<td>CSG Change Subject in Grant</td>
</tr>
<tr>
<td>4</td>
<td>Wrong operation</td>
<td>CEO Change Event Operation</td>
<td>22</td>
<td>Wrong grant target</td>
<td>CTG Change Target in Grant</td>
</tr>
<tr>
<td>5</td>
<td>Missing operation</td>
<td>AEO Add Event Operation</td>
<td>23</td>
<td>Wrong access right in grant</td>
<td>CARG Change Access Right in Grant</td>
</tr>
<tr>
<td>6</td>
<td>Extra operation</td>
<td>REO Remove Event Operation</td>
<td>24</td>
<td>Missing access right in grant</td>
<td>AARG Add Access Right in Grant</td>
</tr>
<tr>
<td>7</td>
<td>Wrong target</td>
<td>CET Change Event Target</td>
<td>25</td>
<td>Extra access right in grant</td>
<td>RARG Remove Access Right in Grant</td>
</tr>
<tr>
<td>8</td>
<td>Extra target</td>
<td>RET Remove Event Target</td>
<td>26</td>
<td>Wrong subject in deny</td>
<td>CSD Change Subject in Deny</td>
</tr>
<tr>
<td>9</td>
<td>Extra condition</td>
<td>ROC Remove One Condition</td>
<td>27</td>
<td>Wrong target in deny</td>
<td>CTD Change Target in Deny</td>
</tr>
<tr>
<td>10</td>
<td>Negated condition</td>
<td>NOC Negate One Condition</td>
<td>28</td>
<td>Wrong access right in deny</td>
<td>CARD Change Access Right in Deny</td>
</tr>
<tr>
<td>11</td>
<td>Extra condition</td>
<td>ROF Remove One Factor</td>
<td>29</td>
<td>Missing access right in deny</td>
<td>AARD Add Access Right in Deny</td>
</tr>
<tr>
<td>12</td>
<td>Negated condition</td>
<td>NOF Negate One Factor</td>
<td>30</td>
<td>Add access right in deny</td>
<td>RARD Remove Access Right in Deny</td>
</tr>
<tr>
<td>13</td>
<td>Extra action</td>
<td>ROA Remove One Action</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Wrong action</td>
<td>COA Change One Action</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Wrong ascendant in create</td>
<td>CAC Change Ascendant in Create</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Wrong descendant in create</td>
<td>CDC Change Descendant in Create</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Wrong direction in create</td>
<td>RDC Reverse Direction of Create</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Wrong assignment ascendant</td>
<td>CASA Change ASCendant in Assign</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Evaluation

### Mutation Scores (%) for GPMS-NGAC

<table>
<thead>
<tr>
<th>Mutant Group</th>
<th># Mutants</th>
<th>OC</th>
<th>AC</th>
<th>DC</th>
<th>FDC</th>
</tr>
</thead>
<tbody>
<tr>
<td>event mutants</td>
<td>1548</td>
<td>87.3</td>
<td>87.5</td>
<td>87.5</td>
<td>87.5</td>
</tr>
<tr>
<td>action mutants</td>
<td>5650</td>
<td>4.7</td>
<td>73.2</td>
<td>73.2</td>
<td>73.2</td>
</tr>
<tr>
<td>condition mutants</td>
<td>168</td>
<td>0</td>
<td>38.1</td>
<td>85.1</td>
<td>91.7</td>
</tr>
<tr>
<td>overall</td>
<td>7366</td>
<td>22.0</td>
<td>75.5</td>
<td>76.6</td>
<td>76.8</td>
</tr>
</tbody>
</table>

### Mutation Scores (%) for Bank

<table>
<thead>
<tr>
<th>Mutant Group</th>
<th># Mutants</th>
<th>OC</th>
<th>AC</th>
<th>DC</th>
<th>FDC</th>
</tr>
</thead>
<tbody>
<tr>
<td>event mutants</td>
<td>188</td>
<td>70.7</td>
<td>72.9</td>
<td>73.9</td>
<td>73.9</td>
</tr>
<tr>
<td>action mutants</td>
<td>78</td>
<td>0</td>
<td>82.8</td>
<td>83.9</td>
<td>83.9</td>
</tr>
<tr>
<td>condition mutants</td>
<td>18</td>
<td>0</td>
<td>38.5</td>
<td>69.2</td>
<td>69.2</td>
</tr>
<tr>
<td>overall</td>
<td>306</td>
<td>43.5</td>
<td>73.0</td>
<td>76.5</td>
<td>76.5</td>
</tr>
</tbody>
</table>

### MKPR Scores

<table>
<thead>
<tr>
<th>Subject</th>
<th>OC</th>
<th>AC</th>
<th>DC</th>
<th>FDC</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPMS-NGAC</td>
<td>34.4</td>
<td>62.4</td>
<td>43.3</td>
<td>32.8</td>
</tr>
<tr>
<td>Bank</td>
<td>11.1</td>
<td>6.1</td>
<td>3.7</td>
<td>2.8</td>
</tr>
</tbody>
</table>

**MS (Mutation Score)** = \#KM (Killed Mutants) / \#NEM (Non-Equivalent Mutants)

**MKPR (Mutants Killed Per Request)** = \#KM / \#Test
Conclusions

• Presented the test coverage criteria for NGAC obligations

• Presented efficient methods for generating tests to satisfy each coverage criterion

• Conducted empirical studies to evaluate the fault detection capabilities and cost-effectiveness of these coverage-based test generation methods

• FDC test suites, can provide a high level of confidence in the correct enforcement of access control