Security Analysis of Role-based Access Control

Anna Lisa Ferrara, Truc Nguyen Lam, and Gennaro Parlato
Cyber Security Centre
University of Southampton

Aims

Maintain desirable security properties of access control policies while delegating administrative privileges.

Access Control Policies are designed to support authorized access to protect resources. When administrative permissions are delegated, security properties could be violated.

Examples of security properties:
- Availability properties
  - A doctor must always be able to access patients’ record
  - Escalation of privileges
  - A receptionist cannot access patients’ records
  - Conflict of interests
  - A doctor cannot be also a receptionist

Tool: Verifier of Access Control

http://users.ecs.soton.ac.uk/gp4/VAC.html

VAC is a tool to verify security of ARBAC policies. VAC uses a combination of abstraction and reduction to program verification in order to perform the security analysis.

Policy

Security Query

- pruning
- ad hoc-abstraction
- encode in a program
- interval-abstractions using INTERPROC
- model-checking using GetaFix

NO: policy correct
YES: security breach
NO: policy correct
YES: may be a false security breach

VAC translates ARBAC policies to imperative programs that simulate the policy abstractly, and then utilizes further abstract-interpretation techniques from program analysis to analyze the programs in order to prove the policies secure.

Experimental Results

VAC has been used to verify real policies and case studies
- hospital policies
- university policies
- bank policies
- three suites of complex policies

After Pruning

<table>
<thead>
<tr>
<th>Roles</th>
<th>Fractions</th>
<th>#Users</th>
<th>Pruned</th>
<th>New Pruned</th>
<th>Time</th>
<th>Violation</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>25</td>
<td>1054</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>0.3s No</td>
</tr>
<tr>
<td>32</td>
<td>135</td>
<td>24</td>
<td>16</td>
<td>23</td>
<td>0.7s</td>
<td>No</td>
</tr>
<tr>
<td>108</td>
<td>2372</td>
<td>3000</td>
<td>0</td>
<td>89</td>
<td>1.7s</td>
<td>No</td>
</tr>
<tr>
<td>136</td>
<td>2372</td>
<td>3000</td>
<td>1</td>
<td>104</td>
<td>1.7s</td>
<td>Yes</td>
</tr>
<tr>
<td>4900</td>
<td>26000</td>
<td>75900</td>
<td>1</td>
<td>370</td>
<td>5.3s</td>
<td>Yes</td>
</tr>
<tr>
<td>20000</td>
<td>80000</td>
<td>200000</td>
<td>1</td>
<td>815</td>
<td>3.9ms</td>
<td>Yes</td>
</tr>
<tr>
<td>30000</td>
<td>120000</td>
<td>400000</td>
<td>1</td>
<td>820</td>
<td>8ms</td>
<td>Yes</td>
</tr>
<tr>
<td>49000</td>
<td>200000</td>
<td>800000</td>
<td>1</td>
<td>1500</td>
<td>14ms</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Key results / deployment / future work

Program verification techniques are effective on complex instances
- pruning implements and uses a blend of those techniques
- abstract interpretation
- BDD based model-checkers

We plan to enrich VAC with
- counter-example-guided-abstraction-refinement (CEGAR)
- under-approximation techniques to detect security breaches (based on SAT/SMT solvers)
- new techniques to analyse temporal ARBAC policies

References


Role-based Access Control

Role-based access control (RBAC) has emerged as a simple and effective access control mechanism for large organizations
- standardized by the NIST
- implement a variety of MAC and DAC policies
- is supported in several systems including Microsoft SQL Servers, Microsoft Active Directory, SELinux, Oracle DBMS, NHS, ...

RBAC simplifies policy specification and the management of user rights using a two-tier management. It groups users into roles and assigns permissions to each role. Administrative role-based access control (ARBAC) is a policy mechanism for controlling how changes can be made to the RBAC policy by various administrators.

Automated Analysis

Access-control systems can be seen as state-transition systems.
In an ARBAC system, state changes occur via administrative operations. Security analysis techniques answer questions such as whether an undesirable state is reachable and whether every reachable state satisfies some safety or availability properties.

- Policies are difficult to inspect by hand (state space explosion)
- Monitoring is not acceptable (denial-of-service)
- Static policy analysis is essential

OUR APPROACH

ARBAC systems can be encoded in programs. Efficient program verification techniques/tools can be adapted/used to analyze ARBAC systems:
- Abstract interpretation (to prove absence of security breaches)
- SAT/SMT solvers (not complete; useful to detect security breaches)
- BDD based model checkers (complete; drawback: state-space-explosion on real policies)

Off-the-shelf program verification tools fail on real ARBAC policies. We propose a scheme that uses a combination of abstraction and reduction to program verification to perform security analysis:
Pruning [1]
- shrink policies, achieving orders of magnitude in reduction
- effective on several instances
  (analyzed via off-the-shelf model-checking technology)
Abstract interpretation
- useful to prove absence of security breaches
- effective on complex instances

We have implemented these techniques in a tool called VAC.

Conflict of interests
- A receptionist cannot access patients' records
- Administrative role-based access control (RBAC) is a policy mechanism for controlling how changes can be made to the RBAC policy by various administrators.