

Klorida Miraj

Constraint Languages in RBAC

CS590U- Access Control: Theory and Practice
Dr. Ninghui Li

Department of Computer Sciences
Purdue University

The plan

- Problem Sketch
- Introduction to Constraints in RBAC
- RCL 2000
- Graphical Access Control Model const. expr.
- Enforcing Constraints
- Human Factors Comparison of query lang.
- Project Goal
- Conclusions

Problem Sketch

- Human factor is the Achilles heel of information security.
GONZALES, J., AND SAWICKA, A. A framework for human factors in Information security. WSEAS International Conference on Information Security (2002).
- To provide efficient security policies:
 - Technology
 - Information Science
 - Psychology
 - Management
- **The problem:** True assertions on the involvement of the human-oriented aspects of computer systems.

Problem Sketch cont.

- **The problem:** True assertions on the involvement of the human-oriented aspects of computer systems.
- **Work done before:** Welty and Stemple, 1981 used human factors testing in the field of computer languages.
- **The solution:** Develop a usability testing which would involve testing human users.
- **How?** By doing a human factors comparison of the different constraint languages of RBAC.

Introduction to constraint lang.

- Most important feature of an Access Control Model: the ability to verify the safety of its configuration.
- But the safety of an Access Control configuration cannot be decided for a general access control model.

HARRISON, M. A., RUZZO, W. L., and ULLMAN, J. D. 1976.
Protection in operating Systems. *Commun. ACM* 19, 8 (Aug.).

- Solution?

Introduction cont.

- Two new approaches:

1. Use of limited Access Control Models, such as Bell LaPadula, Domain and Type Enforcement (DTE), etc.

Complete trust to assign subjects/objects to types.

2. Use of Constraints (which describe safety requirements).

AHN, G., AND SANDHU, R. 1999. The RSL99 language for role-based separation of duty constraints. In *Proceedings of the 4th Workshop on Role-Based Access Control*.

Too Complex? Hard to implement? Can we do better?

Introduction cont.

- We will concentrate on the second approach.
- There have been many different constraint languages for RBAC. We will concentrate on three:
 - RCL 2000
 - Graphical Access Control Model
 - Enforcing Constraints
- We will examine each of these languages carefully.

RCL 2000

AHN, G., AND SANDHU, R. Role-based authorization constraints specifications. *ACM Transactions on Information and System Security* 3, 4 (2000), 207 – 226.

- To specify RBAC constraints, this paper uses a formal language called RCL 2000, which is defined in context of RBAC96.
- Why formal language?
 - formal way to reason about constraints
 - framework for identifying new types of constraints
 - classification scheme for types of constraints
 - basis for supporting optimization and specification techniques

RCL 2000 cont.

- RCL 2000 is a substantial generalization of RSL99, and it adds obligation constraints to the usual separation of duty and prohibition constraints.
- Obligation constraints:
 - constraints that require that certain roles should be simultaneously active in the same session
 - constraints that require a user to have certain combinations of roles in user-role assignment.

RCL 2000: basic components

- RCL 2000 is defined in context of RBAC96.
- Six entity sets: users (U), roles (R), **objects (OBJ)**, **operations (OP)**, permissions (P) and sessions (S).
- The functions **user** and **roles**, no changes on these functions during the life of a session (RBAC roles).
- Two nondeterministic functions: **oneelement** (OE), get one element x from set X, and **allother** (AO), get a set by taking out one element.

RCL 2000: syntax

- The syntax is defined:
 1. **Syntax diagram**
 2. **Grammar**
- Syntax Diagram
 - rules taking the form of flow diagrams
 - possible paths representing the possible sequence of symbols
- Grammar is described by Backus Normal Form (BNF)

RCL 2000: formal semantics

- To discuss the formal semantics for RCL 2000, RFOPL is used.
- RFOPL is equivalent to RCL 2000; any property written in RCL 2000 can be translated to an equivalent RFOPL.
- The soundness and completeness of these two languages proven in the paper.

RCL 2000: examples

- RCL 2000 used to express the properties of SSOD

- No user should be assigned to two roles which conflict

$$|\text{roles}^*(\text{OE}(\text{U})) \cap \text{OE}(\text{CR})| \leq 1$$

- Two conflicting users cannot be assigned to roles in the same conflicting role set

$$|\text{user}(\text{OE}(\text{CR})) \cap \text{OE}(\text{CU})| \leq 1$$

RCL 2000: expressive power

- RCL 2000 specifies the various separation of duty properties.
- RCL 2000 clearly expresses the properties of SSOD, seen in the previous slide's examples.
- RCL 2000 can also be used to express DSOD similarly to SSOD.

Graphical AC Model

JAEGER, T., AND TIDSWELL, J. Practical safety in flexible access control models. *ACM Transactions on Information and System Security* 4, 3 (2001), 158 – 190.

- The access control policy is expressed using a graphical model:
 - nodes represent sets (subjects, objects, etc)
 - edges represent binary relationships on those sets
- Constraints are expressed using a few, simple operators on graph nodes.

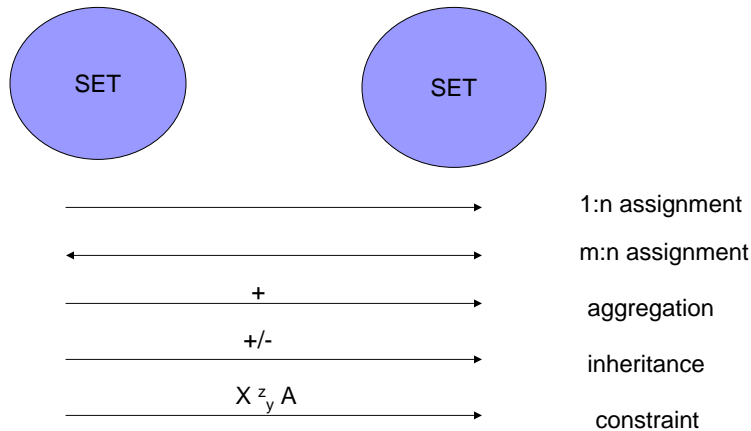
Graphical AC Model cont.

- Constraint expression in this model is simplified:
 - Constraints are defined in terms of binary relations
 - Constraint expressions is split into three steps: set identification, input selection, and input comparison
 - Set identification is done using graphical model
 - Input comparison is done using a small number of set-based ops.

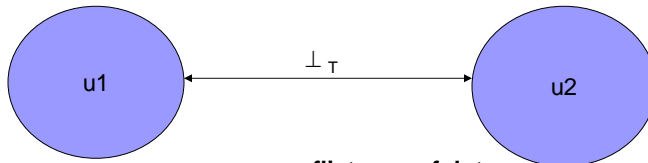
GACM: basic components

- A graph $G = (Y, Z)$ where Y is the set of nodes (each node \rightarrow a set), and Z is the set of edges (each edge \rightarrow relationship).
- **Set types:** S (subject types), P (permission types), T (authorization types), X (sessions), O (object types), Op (operations on object types), *aggregate* (elements of same type), etc.
- **Relationships:** SA (subject to authorization types), PA (permissions to authorization types), SXA (subject to a session), etc

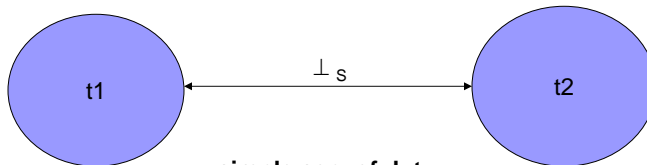
GACM: relationship types



GACM: examples



user-user conflict sep. of duty

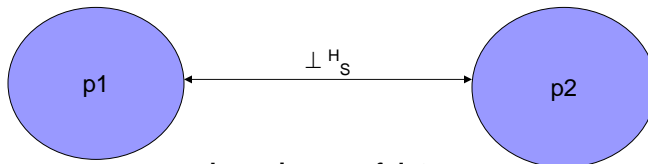


simple sep. of duty

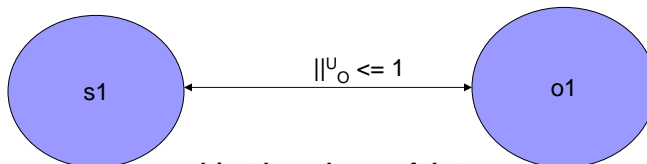
Purdue University

19

GACM: more examples



dynamic sep. of duty



object based sep. of duty

Purdue University

20

GACM: expressive power

- The model is restricted along two dimensions:
 1. all constraints must be expressed as binary relationships
 2. only a limited number of constraint relationship are available

- However these restrictions
 - are useful in reducing the complexity of expressions
 - have not prevented us from expressing the constraints as seen in the examples.

Spec. & Enforc. Constraints

CRAMPTON, J. Specifying and enforcing constraints in role-based access control. *SACMAT '03*. (2003), 43 – 50.

- Two important issues to constraints:
 1. Their specification
 2. Their enforcement

- Informally, a constraint defines a family of “bad” sets.

- Enforcement of a constraint requires that for all bad sets, it is never the case that all elements in a bad set can occur.

SEC: formal description

- Specification scheme:
A constraint is a triple (s, c, x) , where s is the constraint scope, c is the constraint set and x is the context (static, dynamic, historical).
- A static separation of duty constraint which requires that no user can be assigned to both roles $r1$ & $r2$:

$(U, \{r1, r2\}, s)$

SEC: enforcing constraints

- Constrained role-based system = role-based system for which a set of constraints is defined.
- The constraint monitor, an additional component of the reference monitor does the enforcement of the constraints.
- Three types of constraints: static (by the configuration), dynamic (by the state), and historical (by the history).

SEC: blacklist

- Enforcing historical constraints? $(U, \{p1, p2\}, h)$
- Blacklist -> a dynamic access control structure that contains constrained requests.
- In case of a constraint request, go to the blacklist and see what to do. If the request is part of the blacklist, ignore. If not, we refer the request to the role-based reference monitor.

SEC: expressive power

- The model is simpler than the previous ones
- It has defined specifications and enforcement for a class of constraints that includes separation of duty.
- First time an implementation model was discussed.

Testing Human Users

WETLY, C., AND STEMPLE, D. Human factors comparison of a procedural and a nonprocedural query language. *ACM Transactions on Information and System Security* 6, 4 (1981), 626 – 649.

- Efficiency in the use of system resources can be ineffective if a system is not designed to match the needs and abilities of its users.
- Human factors testing:
 1. Test whether a language is learnable
 2. Eliminate minor difficulties in a language

THU: outline

- The languages used: SQL (non-procedural) and TABLET (procedural).
- The effect a difference in procedurality might have on the learning of two very similar query languages.

THU: the experiment

- The experiment run two times, spring 1978 and 1979:
 1. 72 undergraduate split into two groups: one group (35) learning SQL and the other (37) learning TABLET. Half had taken a programming course before; split evenly.
 2. Same number of students, TABLET slightly changed. None had taken a programming course before.
- Same professor taught both languages in both cases.
- Identical manuals with the same examples and problems were used in both languages.
- One final decisive for final grade. Retention test, identical, three weeks after. Hard/easy questions. Grading same in both cases.

THU: the results

- Subjects using TABLET attained higher average scores on the hard queries than those using SQL.
- Clear difference in the ability of students having little or no experience with computers to learn the two lang.
- TABLET was shown to be easier to remember, as retention test average much higher than SQL.

THU: conclusion

- It answered the questions:
 1. Which language was easier to learn
 2. Which language was more efficient to use
 3. Which language was easier to remember
- This makes an example of usability testing.

Project Goal

- Develop a usability testing which would involve testing human users.
- Usability testing, according to Nielsen:
 - Learnability (the system should be easy to learn)
 - Efficiency (the system should be efficient to use)
 - Memorability (the system should be easy to remember)
 - Errors (the system should have a low error rate)
 - Satisfaction (the system should be pleasant to use)

Project Goal cont.

- In other words:
 - I will be using human factors testing to test the different constraint languages for RBAC on how learnable, how efficient, how easy to remember, how error-prone, and how pleasant to use they are.
- By the end of my work, I should be able to give answers to questions such as which language was easier to learn, more efficient to use etc.

Conclusion

- There is no work done before on human factors comparison of the different languages for RBAC.
- A new approach is needed.
- Help from various Software Engineering books on usability testing and from previous work on constraint languages. While working on the human factors comparison will also use the approach of the previous work on query languages.