On the Definition of Role Mining

Mario Frank, Joachim M. Buhmann, David Basin

15th ACM Symposium on Access Control Models and Technologies

link to the paper
What is role mining?
How should this problem be defined?

Also:
• How should it be solved?
• How should solutions be evaluated?

Strategy:
• Start with clear parts such as input/output.
• Look at basic requirements for RBAC.
• Define problem such that solutions meet requirements.
RC=(ROLES,UA,PA)

Input / Output

Role Mining

UPA

USERS PRMS
Input / Output

Role Mining

RC=(ROLES, UA, PA)

Role Hierarchy (RH)

Deployed Roles

Top-Down Information (TDI)

UPA

USERS PRMS
RC = (ROLES, UA, PA)

Input / Output

Role Mining

Top-Down Information (TDI)

Deployed Roles

Role Hierarchy (RH)
What is required from an RBAC configuration?

Candidates:

- **Perfect** match with original assignment UPA (``0-consistency``)
- **Best possible** match with UPA.
- The ``smaller`` the configuration the better (**best compression**).
  - Number of roles
  - Number of assignments
  - Number of exceptions
  - Linear combination of size measures
- **No** transfer of **errors** from UPA to RBAC (violates perfect match)
- ...

Hard to decide which ones to take since all very technical. Our understanding of the requirements are more high-level.
What is required from an RBAC configuration?

Most important requirements from an enterprises perspective:

• **Provisioning**
  Users are enabled to carry out their tasks.

• **Security**
  Configuration conforms to the enterprise security policies.

• **Maintainability**
  Administration of the system is as easy as possible:
  - understandable roles
  - easy to add users (roles generalize well)
What do we actually get as an input?

**Black boxes**: observed entities

**Gray boxes**: hidden entities

**Legend**
- RC*: hidden role configuration/structure underlying UPA
- UPA’: direct assignment generated from RC*
- UPA: UPA’ perturbed by exceptions/errors (”noise“)
- TDI: any information that possibly influenced UPA

**Assumptions**
1. A hidden structure **RC* underlies UPA**
2. **RC* reflects top-down information** (parts of which are possibly given as additional role mining input).
3. **Exceptions** (errors) might exist.
What do we actually get as an input?

Black boxes: observed entities
Gray boxes: hidden entities

Legend
RC*: hidden role configuration/structure underlying UPA
UPA’: direct assignment generated from RC*
UPA: UPA’ perturbed by exceptions/errors (``noise“)
TDI: any information that possibly influenced UPA

Assumptions
1. A hidden structure RC* underlies UPA
2. **RC* reflects top-down information** (parts of which are possibly given as additional role mining input).
3. Exceptions (errors) might exist.
Definition **INFERENCE RMP:**
Let a set of users USERS, a set of permissions PRMS, a user-permission relation UPA, and, optionally, part of the top-down information TDI be given. Under Assumption 1-3, infer the unknown RBAC configuration RC*=(ROLES*, UA*, PA*).

**Assumptions** (from last slide):
1. Structure **R* is hidden in UPA**
2. **R* reflects top-down information** (TDI).
3. **Exceptions** (errors) might exist.
Why is this a good definition?

Rationale:
The solution fulfills the real-world requirements.

• Input data UPA is generated from underlying RC* (modulo exceptions)
• RC* reflects security policies and business properties of the enterprise

⇒ RC* is configuration that
  • fulfills provisioning requirement
  • conforms to the enterprises security policies
  • is intuitive
Solving the problem and assessing solutions

Pointer to some ways of solving and evaluating that problem.

Solving:
• Difficult!

• Use your own method of choice to attack this problem.

• E.g., we used a probabilistic approach [1,2,3]:
  RC* is the most probable configuration under an appropriate model $\Rightarrow$ RM as a modeling problem

Solving the problem and assessing solutions

Assessing:

- easy when RC* is known (artificially created data UPA)
- avoid repeated comparison! Can give very good scores to trivial solutions.

\[
\begin{align*}
\text{inferred roles} & \quad \text{true roles} \\
{r_1 = \{p_1, p_4, p_8\}} & \quad {r^*_1 = \{p_1, p_4, p_8\}} \\
{r_2 = \{p_4, p_7\}} & \quad {r^*_2 = \{p_1, p_2\}}
\end{align*}
\]

- find the **global** permutation of roles that minimizes the deviation (can be found via Hungarian method).

\[
\begin{align*}
\text{inferred roles} & \quad \text{true roles} \\
{r_1 = \{p_1, p_4, p_8\}} & \quad {r^*_1 = \{p_1, p_4, p_8\}} \\
{r_2 = \{p_4, p_7\}} & \quad {r^*_2 = \{p_1, p_2\}}
\end{align*}
\]

method is demonstrated in [1]

Unknown RC*: Generalization Test

It is still possible to evaluate solutions!
Exploit that underlying structure RC* reproduces over the users, whereas the noise does not.

Generalization test:
1. randomly split UPA in UPA(1) and UPA(2)
2. train roles R on UPA(1)
3. assign users from UPA(2) to roles based on x% of their permissions
4. predict remaining (100-x)% of permissions
5. compute prediction error

The closer solution is to RC* the better is the prediction error.

See [1] for such an evaluation.

Unknown RC*: Generalization Test with TDI

When top-down information is available it should be included in the assessment of the found RBAC states.

Generalization test (when TDI is given):
1. randomly split UPA in UPA(1) and UPA(2) and split TDI in TDI(1) and TDI(2)
2. train roles R on UPA(1) and TDI(1)
3. assign users from UPA(2) to roles based on x\% of their permissions and their top-down properties
4. predict remaining (100-x)\% of permissions
5. compute prediction error

See [2] for such an evaluation.

Summary

We have presented:

• Novel definition of the role mining problem
  • motivated from basic requirements on RBAC and
  • relying on realistic assumptions on the input data
• Pointer to high-level solution strategy
• Evaluation techniques exist

Appeal to the community:

• Papers on role mining methods should contain problem definition and evaluation criteria.
• Definition, algorithm and evaluation should agree.
• Let’s try to agree on one definition of the problem (discuss!).
Thank You