Role Mining Based on Weights

Xiaopu Ma Ruixuan Li Zhengding Lu

School of Computer Science and Technology
Huazhong University of Science and Technology

SACMAT 10, Pittsburgh
June 9, 2010
Outline

1 Motivation
   - Role engineering
   - Motivating example
   - Contributions

2 Role Mining Based on Weights
   - How to compute weights
   - How to mine roles based on weights

3 Experimental Results
   - Performance of the algorithm
   - Accuracy and coverage of the algorithm

4 Summary

Ruixuan Li, Huazhong University of Science and Technology
Role Mining Based on Weights
1 Motivation
   - Role engineering
   - Motivating example
   - Contributions

2 Role Mining Based on Weights
   - How to compute weights
   - How to mine roles based on weights

3 Experimental Results
   - Performance of the algorithm
   - Accuracy and coverage of the algorithm

4 Summary
Outline

1. Motivation
   - Role engineering
   - Motivating example
   - Contributions

2. Role Mining Based on Weights
   - How to compute weights
   - How to mine roles based on weights

3. Experimental Results
   - Performance of the algorithm
   - Accuracy and coverage of the algorithm

4. Summary
Outline

1. Motivation
   - Role engineering
   - Motivating example
   - Contributions

2. Role Mining Based on Weights
   - How to compute weights
   - How to mine roles based on weights

3. Experimental Results
   - Performance of the algorithm
   - Accuracy and coverage of the algorithm

4. Summary
Outline

1 Motivation
   - Role engineering
     - Motivating example
     - Contributions

2 Role Mining Based on Weights
   - How to compute weights
   - How to mine roles based on weights

3 Experimental Results
   - Performance of the algorithm
   - Accuracy and coverage of the algorithm

4 Summary
Role engineering is introduced as a solution to create a comprehensive framework of defining the structure of RBAC.

- **Top-down**
  - Fernandez: Determine role rights from use cases [RBAC97]
  - Neumann: Determine functional roles through scenarios [SACMAT02]
  - Time consuming, well reflect the functional requirements

- **Bottom-up**
  - ORCA: Build a hierarchy using role mining [SACMAT05]
  - RoleMiner: Use a subset enumeration to mine roles [CCS06]
  - Automatically, ignore the functional requirements
Outline

1 Motivation
   • Role engineering
   • Motivating example
   • Contributions

2 Role Mining Based on Weights
   • How to compute weights
   • How to mine roles based on weights

3 Experimental Results
   • Performance of the algorithm
   • Accuracy and coverage of the algorithm

4 Summary
Motivating Example

Most of the existing role mining approaches didn’t consider the different nature and importance of each permission, or treated the permissions evenly.

- **Problems**
  - The important roles may not be identified by the traditional role mining techniques.

- **Example of importance**
  - “read” and “write” of patient’s personal information
  - “read” and “write” of student’s achievement
What is weight

The weight of permissions can be viewed as a function of selected attributes attached to permissions.

- different properties of operations
- different sensitive degrees of objects
- different attributes of users associated with permissions
Outline

1 Motivation
   - Role engineering
   - Motivating example
   - Contributions

2 Role Mining Based on Weights
   - How to compute weights
   - How to mine roles based on weights

3 Experimental Results
   - Performance of the algorithm
   - Accuracy and coverage of the algorithm

4 Summary
Contributions

- Give a way to compute the weights of permissions
- Design an algorithm to mine roles based on weights
- Evaluate the performance of the algorithm
Outline

1. Motivation
   - Role engineering
   - Motivating example
   - Contributions

2. Role Mining Based on Weights
   - How to compute weights
   - How to mine roles based on weights

3. Experimental Results
   - Performance of the algorithm
   - Accuracy and coverage of the algorithm

4. Summary
Intra-relationship: relationship between permissions (or between users)

Inter-relationship: relationship between user and permission
The original similarity between both users and permissions

- The original similarity between users
  \[
  \text{sim}(u_i, u_j)^{\text{origsim}} = \frac{|\text{UserPerms}(u_i) \cap \text{UserPerms}(u_j)|}{|\text{UserPerms}(u_i) \cup \text{UserPerms}(u_j)|}
  \]

  Identify whether roles exist

- The original similarity between permissions
  \[
  \text{sim}(p_i, p_j)^{\text{origsim}} = \frac{|\text{PermUsers}(p_i) \cap \text{PermUsers}(p_j)|}{|\text{PermUsers}(p_i) \cup \text{PermUsers}(p_j)|}
  \]

  Identify whether user groups exist
The reinforced similarity between permissions

- Factors affecting the similarity between permissions
  - Inter-relationship
  - Intra-relationship

- The reinforced similarity between permissions

\[
\text{sim}(p_i, p_j)^{resim} = \alpha \times \text{sim}(p_i, p_j)^{origsim} + \frac{\beta \times \sum_{\forall x, y \in P \cup U} \text{sim}(p_i, x)^{origsim} \text{sim}(p_j, y)^{origsim} \text{sim}(x, y)^{origsim}}{(|P| + |U|)^2}
\]
The weight of each permission

\[
\omega_{p_i} = \gamma \times \frac{\sum_{j=1, j \neq i}^{n} \text{sim}(p_i, p_j)^{\text{resim}}}{(n - 1)} + \delta \times \omega_0
\]

Where \(\omega_0\) is the initial weight of permission \(p_i\) preset by the system based on knowledge of comprehensive effect of all factors on permission \(p_i\); \(\gamma\) and \(\delta\) are parameters used to adjust the relative importance.
The weight and weighted support of permission set

- The weight of permission set

\[ \omega_{PS} = \sum_{i=1}^{k} \omega_{pi} \]

- The weighted support of permission set

\[ wsf(PS) = \omega_{PS} \times \frac{numUsers(PS)}{numUsers(All)} \]

where \( numUsers(PS) \) is the number of users which possess permission set \( PS \), and \( numUsers(All) \) is the total number of users in user-permission assignments.
The process of role mining based on weights

- Compute the original similarity between users and between permissions
- Compute the reinforced similarity between permissions
- Compute the weight of each permission
- Use the alternative *Weighted Aprior* algorithm to generate the frequent permission set as roles
Running example

<table>
<thead>
<tr>
<th></th>
<th>(p_1)</th>
<th>(p_2)</th>
<th>(p_3)</th>
<th>(p_4)</th>
<th>(p_5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(u_1)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(u_2)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(u_3)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(u_4)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Original similarity matrix

\[
\begin{pmatrix}
\begin{array}{ccccc}
  u_1 & u_2 & u_3 & u_4 \\
  p_1 & 1 & 0.5 & 0.5 & 0.25 \\
  p_2 & 0.5 & 1 & 0.6 & 0.75 \\
  p_3 & 0.5 & 0.6 & 1 & 0.4 \\
  p_4 & 0.25 & 0.75 & 0.4 & 1 \\
\end{array}
\end{pmatrix}
\begin{pmatrix}
  p_1 \\
p_2 \\
p_3 \\
p_4 \\
p_5 \\
\end{pmatrix}
= 
\begin{pmatrix}
  0.91 & 0.68 & 0.61 & 0.30 & 0.46 \\
  0.68 & 0.91 & 0.45 & 0.23 & 0.68 \\
  0.61 & 0.45 & 0.90 & 0.00 & 0.23 \\
  0.30 & 0.23 & 0.00 & 0.90 & 0.30 \\
  0.46 & 0.68 & 0.23 & 0.30 & 0.91 \\
\end{pmatrix}
\]
Running example

- Reinforced similarity between permissions

\[
\begin{pmatrix}
 p_1 & p_2 & p_3 & p_4 & p_5 \\
 p_1 & 0.91 & 0.68 & 0.61 & 0.30 & 0.46 \\
 p_2 & 0.68 & 0.91 & 0.45 & 0.23 & 0.68 \\
 p_3 & 0.61 & 0.45 & 0.90 & 0.00 & 0.23 \\
 p_4 & 0.30 & 0.23 & 0.00 & 0.90 & 0.30 \\
 p_5 & 0.46 & 0.68 & 0.23 & 0.30 & 0.91 \\
\end{pmatrix}
\]

- The weight of each permission

\[
\omega_{p_1} = 0.51, \omega_{p_2} = 0.51, \omega_{p_3} = 0.32, \omega_{p_4} = 0.21, \omega_{p_5} = 0.42
\]
Running example

- The frequent permission sets ($w\text{minsup} = 0.4$)
  
  $F_1 = \{(p_2, u_1 u_2 u_3 u_4, 0.51)\}$
  $C_1 = \{(p_1, u_2 u_3 u_4, 0.38) (p_2, u_1 u_2 u_3 u_4, 0.51) (p_3, u_2 u_4, 0.16) (p_5, u_1 u_2 u_3, 0.31)\}$
  
  $F_2 = C_2 = \{(p_1 p_2, u_2 u_3 u_4, 0.77) (p_1 p_3, u_2 u_3, 0.42)(p_1 p_5, u_2 u_3, 0.465) (p_2 p_3, u_2 u_4, 0.42)(p_2 p_5, u_2 u_3, 0.465)\}$
  
  $F_3 = \{(p_1 p_2 p_3, u_2 u_4, 0.67) (p_1 p_2 p_5, u_2 u_3, 0.72)\} C_3 = F_3 \cup \{(p_1 p_3 p_5, u_2, 0.31) (p_2 p_3 p_5, u_2, 0.31)\}$
  
  $F_4 = C_4 = \{(p_1 p_2 p_3 p_5, u_2, 0.44)\}$

- Hence the resulting role is $r_1$
  (where RolePerms($r_1$) = $\{p_1, p_2, p_3, p_5\}$)
Outline

1 Motivation
   - Role engineering
   - Motivating example
   - Contributions

2 Role Mining Based on Weights
   - How to compute weights
   - How to mine roles based on weights

3 Experimental Results
   - Performance of the algorithm
   - Accuracy and coverage of the algorithm

4 Summary
The descriptions of the experiments

- We compare our algorithm with *Apriori* algorithm and *ORCA* algorithm.
- We consider the performance and accuracy of our algorithm.
- We implement these algorithms on a Pentium (R)D 2.8G PC with 1GB memory.
- We choose the parameters $\alpha = 0.9, \beta = 0.1, \gamma = 1, \delta = 0$. 
Performance comparison with fixed number of users

<table>
<thead>
<tr>
<th></th>
<th>No. of users</th>
<th>No. of permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>data1</td>
<td>2000</td>
<td>500</td>
</tr>
<tr>
<td>data2</td>
<td>2000</td>
<td>1000</td>
</tr>
<tr>
<td>data3</td>
<td>2000</td>
<td>1500</td>
</tr>
<tr>
<td>data4</td>
<td>2000</td>
<td>2000</td>
</tr>
</tbody>
</table>

- **WRM** algorithm can find more roles than others.
- **WRM** algorithm scans the user-permission assignments only once that costs less.

Ruixuan Li, Huazhong University of Science and Technology
Performance comparison with fixed number of permissions

<table>
<thead>
<tr>
<th>No. of users</th>
<th>No. of permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>data1</td>
<td>500</td>
</tr>
<tr>
<td>data2</td>
<td>1000</td>
</tr>
<tr>
<td>data3</td>
<td>1500</td>
</tr>
<tr>
<td>data4</td>
<td>2000</td>
</tr>
<tr>
<td></td>
<td>1500</td>
</tr>
</tbody>
</table>

(a) Number of roles

(b) Search time
Outline

1 Motivation
   - Role engineering
   - Motivating example
   - Contributions

2 Role Mining Based on Weights
   - How to compute weights
   - How to mine roles based on weights

3 Experimental Results
   - Performance of the algorithm
   - Accuracy and coverage of the algorithm

4 Summary

Ruixuan Li, Huazhong University of Science and Technology
Role Mining Based on Weights
Accuracy and Coverage

The accuracy of the algorithm is defined as the ratio of the number of generated roles exactly matching the original role sets to the number of original role sets.

The coverage of the algorithm is defined as the ratio of the number of permissions covered by the generated roles to the number of total original permissions.
Accuracy and coverage with fixed numbers of users and permissions

<table>
<thead>
<tr>
<th></th>
<th>No. of users</th>
<th>No. of permissions</th>
<th>No. of roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>data1</td>
<td>1000</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>data2</td>
<td>1000</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>data3</td>
<td>1000</td>
<td>100</td>
<td>150</td>
</tr>
<tr>
<td>data4</td>
<td>1000</td>
<td>100</td>
<td>200</td>
</tr>
</tbody>
</table>

(a) Accuracy

(b) Coverage
Accuracy and coverage with fixed numbers of users and roles

<table>
<thead>
<tr>
<th></th>
<th>No. of users</th>
<th>No. of permissions</th>
<th>No. of roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>data1</td>
<td>2000</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>data2</td>
<td>2000</td>
<td>500</td>
<td>100</td>
</tr>
<tr>
<td>data3</td>
<td>2000</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>data4</td>
<td>2000</td>
<td>1500</td>
<td>100</td>
</tr>
</tbody>
</table>

(a) Accuracy

(b) Coverage
Accuracy and coverage with fixed number of permissions and roles

<table>
<thead>
<tr>
<th></th>
<th>No. of users</th>
<th>No. of permissions</th>
<th>No. of roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>data1</td>
<td>1000</td>
<td>500</td>
<td>100</td>
</tr>
<tr>
<td>data2</td>
<td>2000</td>
<td>500</td>
<td>100</td>
</tr>
<tr>
<td>data3</td>
<td>2500</td>
<td>500</td>
<td>100</td>
</tr>
<tr>
<td>data4</td>
<td>3000</td>
<td>500</td>
<td>100</td>
</tr>
</tbody>
</table>

(a) Accuracy

(b) Coverage
Accuracy and coverage with varied numbers of users, permissions and roles

<table>
<thead>
<tr>
<th>No. of users</th>
<th>No. of permissions</th>
<th>No. of roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>data1</td>
<td>500</td>
<td>100</td>
</tr>
<tr>
<td>data2</td>
<td>1000</td>
<td>500</td>
</tr>
<tr>
<td>data3</td>
<td>2000</td>
<td>1000</td>
</tr>
<tr>
<td>data4</td>
<td>2500</td>
<td>1500</td>
</tr>
</tbody>
</table>

For all cases, WRM algorithm generates more roles exactly covered by the original role set than others.
Summary

- Consider the nature and importance of permissions
- Use the similarity between both users and permissions to compute the weight of permissions
- Propose a weighted role mining algorithm to generate roles based on weights that scans the database only once

Future work:
- Some comprehensive ways to integrate weights into role mining
- Employ weights in other related issues
Questions and comments?

rxli@hust.edu.cn

http://individual.utoronto.ca/rxli/