**Sample Answers for Homework 5 (selected from submissions)**

Course IS-2150/TEL-2810 Introduction to Computer Security

**Question a** (10 points)

<table>
<thead>
<tr>
<th>Role (r)</th>
<th>assigned_user(r)</th>
<th>authorized_users(r)</th>
<th>Assigned_permissions(r)</th>
<th>authorized_permissions(r)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manager</td>
<td>{e_{10}}</td>
<td>{e_{10}}</td>
<td>{p_p}</td>
<td>{p_p, p_a, p_b, p_o, p_x, p_y, p_1, p_2, p_m, p_n}</td>
</tr>
<tr>
<td>Senior administrator</td>
<td>{e_5}</td>
<td>{e_5, e_{10}}</td>
<td>{p_a, p_0}</td>
<td>{p_a, p_b, p_x, p_y, p_1, p_2}</td>
</tr>
<tr>
<td>Administrator</td>
<td>{e_3, e_4}</td>
<td>{e_3, e_4, e_5, e_{10}}</td>
<td>{p_x, p_y}</td>
<td>{p_x, p_y, p_1, p_2}</td>
</tr>
<tr>
<td>Senior engineer</td>
<td>{e_8, e_9}</td>
<td>{e_8, e_9, e_{10}}</td>
<td>{p_o}</td>
<td>{p_o, p_m, p_n, p_1, p_2}</td>
</tr>
<tr>
<td>Engineer</td>
<td>{e_6, e_7}</td>
<td>{e_6, e_7, e_8, e_9, e_{10}}</td>
<td>{p_m, p_n}</td>
<td>{p_m, p_n, p_1, p_2}</td>
</tr>
<tr>
<td>Employee</td>
<td>{e_1, e_2}</td>
<td>{e_1, e_2, e_3, e_5, e_6, e_7, e_8, e_9, e_{10}}</td>
<td>{p_1, p_2}</td>
<td>{p_1, p_2}</td>
</tr>
</tbody>
</table>

(Kroeksak Tangmankhong)

**Question b**

(i) (10 points)

UA_1 is invalid, because there exists an intersection \( \cap r \in \{r_1, r_2, r_3\} \)
\( \text{assigned_users}(r) = \{u_1\} \). Every set \( \cap r \in t \text{assigned_users}(r) \) for \(|t| \geq 3\) should be empty, but this one is not. This implies that there exists a user u_1 who has 3 roles, which is not allowed according to SSD.

UA_2 is valid, because there exists no intersection \( \cap r \in \{r_1, r_2, r_3, r_4\} \)
\( \text{assigned_users}(r) \) that is not empty.

(Marco Trautmann)

(ii) (10 points)

SSD – The static separation of duty is concerned with all roles, which a user might be authorized to use. It is not limited to the currently active roles within a session but goes beyond that.

DSD – The dynamic separation of duty is only concerned with the currently active roles of a user within a session.

Having both of these separation of duty constraints present in a system at the same time could lead to conflicts. Suppose a user is authorized to roles \( r_1, r_2 \) and \( r_3 \). According to \( \{r_1, r_2, r_3, r_4\}, 3 \in SSD \) the static separation of duty constraint would be violated, while \( \{r_1, r_2, r_3, r_4\}, 3 \in DSD \) the dynamic separation of duty constraint could still be valid given that the user does not activate all three roles simultaneously.

(Dominik Pozny)
(iii) (10 points)
Conceptually, any two roles should be able to be in separation of duty. However, in the given hierarchy, the senior role *Manager* inherits all permissions from the junior roles (*Senior Administrator* and *Senior Engineer*) as a result of which conflicts may arise in the *Manager* role when there is separation of duty constraints between the junior roles. Therefore, separation of duty should not be allowed between *Senior Administrator* and *Senior Engineer* roles.

(iv) (10 points)

**Transformation rules:**

1. Create for each integrity level $L_1, L_2, \ldots, L_n$ two roles $L_iR$ and $L_iW$ such that the set of roles $R = \{L_1R, L_2R, \ldots, L_nR, L_1W, L_2W, \ldots, L_nW\}$.
2. Create to separate hierarchies for the roles $\{L_1R, L_2R, \ldots, L_nR\}$ and $\{L_1W, L_2W, \ldots, L_nW\}$ such that the hierarchy of the set $\{L_1W, L_2W, \ldots, L_nW\}$ corresponds to the lattice of the integrity levels and the hierarchy of the set $\{L_1R, L_2R, \ldots, L_nR\}$ is exactly the inverse lattice of the integrity levels.
3. Each user is assigned to exactly two roles $L_iR$ and $L_iW$.
4. Each session has exactly two roles $L_iR$ and $L_iW$.
5. Permission $(o, r)$ is assigned to $L_iR$ if and only if $(o, w)$ is assigned to $L_iW$.

Here the object $o$ is of the corresponding integrity level $L_i$.

**Arguments for the correctness of the transformation:**

Suppose we have a simple integrity model with three levels of integrity as depicted in the picture above. According to the transformation procedure one has to create two separate hierarchies of roles as depicted in the picture above. Suppose a user with integrity level $M_1$ is assigned to the roles $M_1R$ and $M_1W$. By activating both roles in a session this user can read and write objects of the integrity level $M_1$. Furthermore, given both role hierarchies the user is authorized to the roles $HR$ and $LW$, i.e. he inherits the permissions of the roles $HR$ and $LW$. This allows the user to read objects of a higher integrity level, which corresponds to Biba’s first rule, and it allows the user to write objects of a lower integrity level, which corresponds to Biba’s second rule.

*(Dominik Pozny)*