CSI 5342 Assignment 14.1

By: Matthew Tuan and Robbie Dutton

The first step in specifying a location-aware role-based access control (LRBAC) model is to update our RBAC model from Assignment 13.1 to incorporate the Location class:

A screenshot of a computer

AI-generated content may be incorrect.

The class and associations are based on the model developed in “LRBAC: A Location-Aware Role-Based Access Control Model” which incorporates locations with Users, Roles, Permissions, and Objects.

**Impact of Location on Role Hierarchies and SSD/DSD constraints**

Role Hierarchies contain a Senior-Junior relationship where the Senior roles can inherit the permissions of Junior roles or activate a Junior role. In the paper “A Spatio-Temporal Role-Based Access Control Model”, they outline 8 different hierarchical constraints based on unrestricted and restricted permission inheritance. Focusing only on the location-based constraints in definitions 11 and 12, we will add location restricted permission inheritance and location restricted activation constraints as follows:

A white background with black text

AI-generated content may be incorrect.A computer screen shot of a computer program

AI-generated content may be incorrect.

Next looking at SSD constraints, we need to ensure that a user cannot be assigned two roles that have SSD constraints while also taking into account the location of the user.

A close-up of black text

AI-generated content may be incorrect.

Testing out the new constraint we have 2 script tests: InvalidSSDLocationTest.x and ValidSSDLocationTest.x. These basic files test out a basic session with one user, two roles, and two locations. When a user is assigned two roles with SSD constraints and when the locations overlap the invariant fails. We can see a successful state when the user only has one role assigned and does not have an SSD conflict.

A computer screen shot of a diagram

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

DSD constraints also need to be updated to ensure conflicting roles with the same location cannot be activated in the same session.

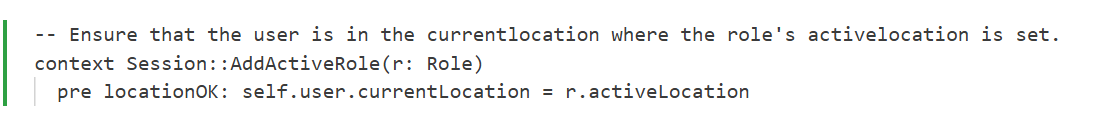
A black text on a white background

AI-generated content may be incorrect.

Moving forward, we next started to update the CheckAccess operation in Session class. Before that however, we needed to update and add a few constraints to our model. The first one below is quite similar to the previous model, however with LRBAC, we had to add the check that the user’s current location was the same as the role’s assigned location. The rest stayed the same for that.  
A computer code with black text

AI-generated content may be incorrect.

Next, we added another pre condition for a role to a session.



Quite similar to previously, we also needed to check the location was correct for the user’s session. So this was added as well. Next we also created a new association for SessionLocation between one location and many sessions at that said location.

A screenshot of a computer

AI-generated content may be incorrect.

To further this, we updated the User::CreateSession() post conditions, to now link a user session   
A screen shot of a computer code

AI-generated content may be incorrect.

Lastly, we can now update the Session::CheckAccess() operation.

A screenshot of a computer program

AI-generated content may be incorrect.

To start off, we get all the roles unioned into a set. We first check the location of the role is valid for the user’s current location. Next, for each role, we then make sure some of the original attributes are valid, like that the object and operation are correct. Lastly, we should ensure that the user’s location is allowed by the roles permission and same goes for the object’s current location.  
  
To test this all out, we created a test file called `lrbac-valid.x`, this results in the following object diagram.

A computer screen shot of a network

AI-generated content may be incorrect.

In the example above, we basically just created one of each type of object and created their associations. Now, to test this out, we just run the following command by calling the session’s CheckAccess operation.  
  
A screenshot of a computer

AI-generated content may be incorrect.  
  
For this next test, we wanted to make sure a user with a different current location, than the object’s current location, would result in a false when returning from the Session::CheckAccess(). Below was the model generated via the ‘lrbac-invalid.x’ file.

A screenshot of a computer

AI-generated content may be incorrect.

Running a test on this produce the correct result of false!

A screenshot of a computer

AI-generated content may be incorrect.

**Summary of each member’s contribution**