RBACvisual User Guide

Contents

List of Figures

1 Overview

1.1 Introduction

RBACvisual is a visualization tool designed to facilitate the study and teaching of the role-based access control (RBAC) model. This model is widely used to restrict system access to authorized users. It can be utilized not only for the implementation of mandatory access control but also for discretionary access control. RBACvisual focuses on the interpretation of the user-to-role and role-to-object relationship combined with the role inheritance hierarchy. The visualization allows the analysis of an access control specification file with two different views and the modification of the specification via textual input and graphical operations. Additionally, a Practice and Test section is provided for evaluation of the understanding of the model.

1.2 User Interface

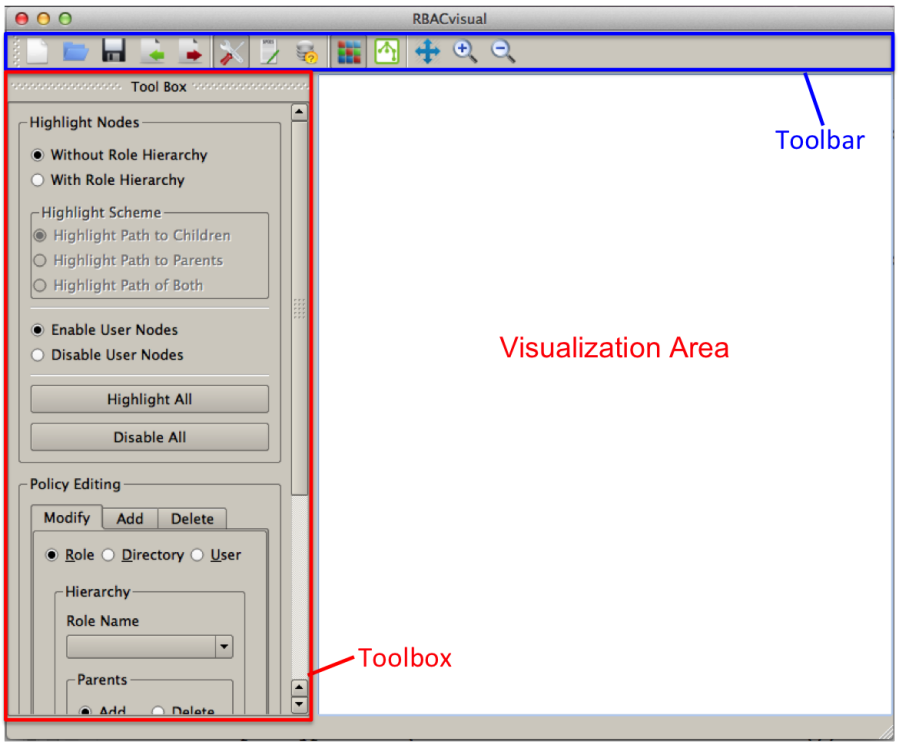


Figure 1: User Interface

Figure 1 shows the user interface of RBACvisual. There are three main sections. The toolbar is the section in the blue frame and provides shortcuts to commonly-used functionality. The left hand side within the red frame is the toolbox. It has two sections. The Highlight Nodes section controls characteristics of the policy visualizations. The Policy Editing section is used to modify the specification under analysis. The blank part on the right hand side is the visualization area where graphical depictions of the policy under analysis will appear.

The toolbar as in Figure 2 provides shortcuts to the most commonly-used functionalities in the software. It supports functionalities such as New session, Open/Save visfile and Import/Export spec for input and output. Users may start from scratch and build a new policy step by step or take either visfile or spec file as

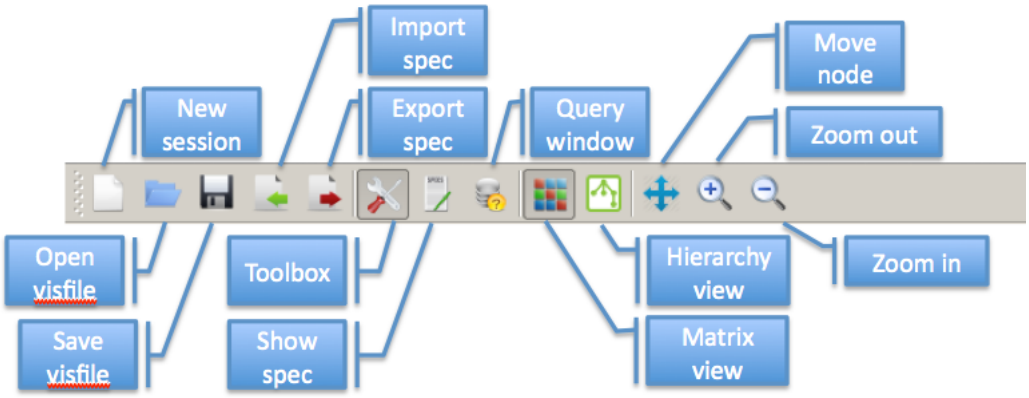


Figure 2: Toolbar

basic policy and apply further changes. It also contains the switches for showing the specification and for controlling the visualization of the policy under analysis. These functions are described in more detail below.

2 Input and Output

The section describes the input and output of the tool. The types of files supported and the methods to load and generate specific file types will be explained.

2.1 File Types

RBACvisual supports two types of files: specification file (.rbac) and visualization file (.rbacvis). The specification file contains text, in the format given below, that describes role inheritance, user-to-role assignments and role-to-object permissions. It allows the policy to be saved in the specification file format. The visualization file stores the graphical information for the visualization, which provides a way to save user's own graphical arrangement and configurations.

The syntax of the specification file is shown below:

 # Comments after "#"
 # Role inheritance

"inheritance:" `<ROLE>(> RoleList)*`

 # users

"user:" `RoleList UserList`

 # objects

"object:" `RoleList PermList Res`

where

$$\texttt{Res} \rightarrow (\text{“-r”}\ \texttt{<PATH>}) \mid \texttt{<PATH>}$$
$$\texttt{RoleList} \rightarrow \texttt{<ROLE>}\ (\text{“,”}\texttt{<ROLE>})^*$$
$$\texttt{UserList} \rightarrow \texttt{<USER>}\ (\text{“,”}\texttt{<USER>})^*$$
$$\texttt{PermList} \rightarrow \texttt{<PERM>}\ (\text{“,”}\texttt{<PERM>})^*$$

It results in policy specifications that look like the following:

```
inheritance: role1 > role2, role3
inheritance: role2 > role3
inheritance: role3

user: role1 user1               # one user one role
user: role1,role2,role3 user2   # one user, multiple roles
user: role2 user3,user4         # multiple users, one role
user: role1,role2 user5,user6   # multiple users, multiple roles

object: role1 r -r /usr
object: role2 r,x -r /lib
object: role2,role3 r,w,x /usr/somefile
```

2.2 File Operation

Following are the supported operations for file input and output, which are available from the toolbar and application File menu.

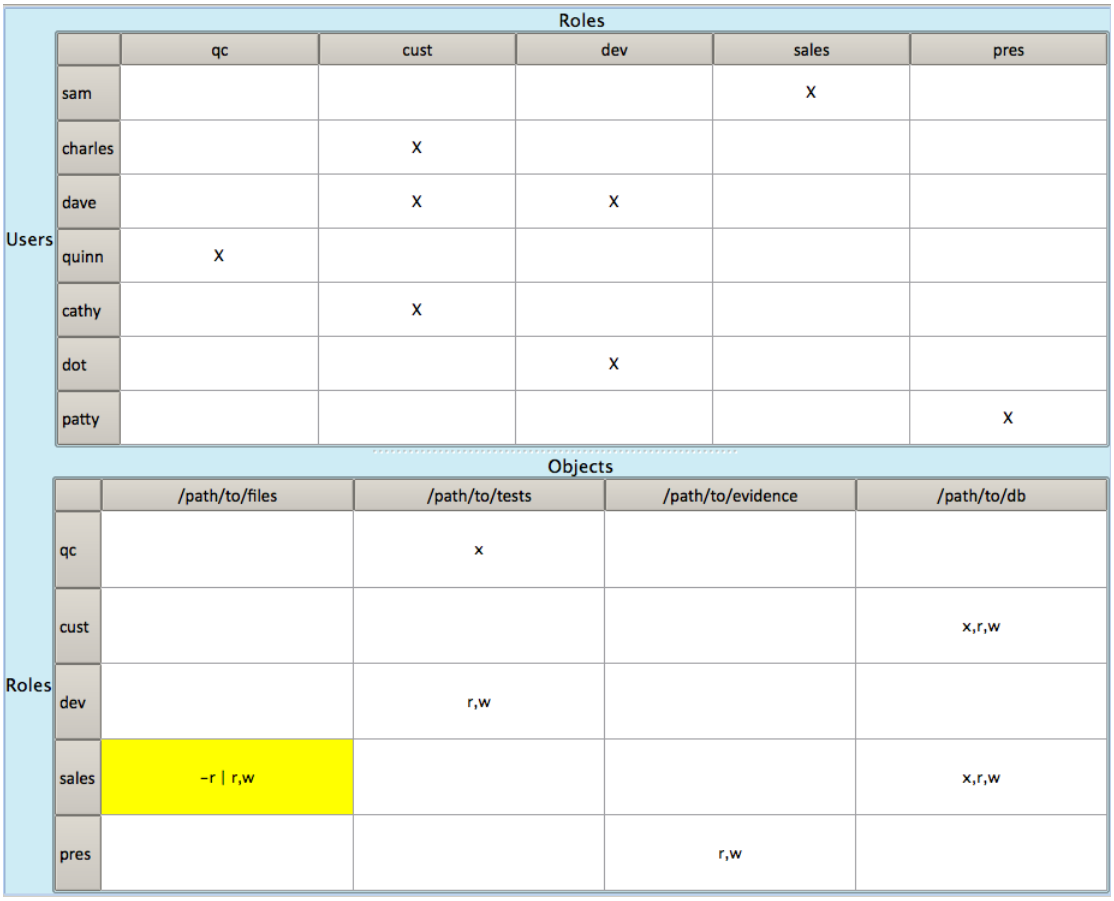
- **New:** Create a blank policy.
- **Import:** Read in rbac specification file.
- **Export:** Save present policy as rbac specification file. No visualization information will be saved.
- **Open:** Read in rbacvis visualization file.
- **Save:** Save present visualization result to rbacvis visualization file.

3 Visualization

This section will walk through the functionality for analyzing a policy and the ways to modify an existing policy both via graphical operations and toolbox widgets.

3.1 Views

Two different views – Matrix View and Hierarchy View are available to examine a policy. The matrix view shows the user-to-role and role-to-object permission as matrices. The top matrix is for the user-to-role



Roles

Users	qc	cust	dev	sales	pres
sam				X	
charles		X			
dave		X	X		
quinn	X				
cathy		X			
dot			X		
patty					X

Objects

Roles	/path/to/files	/path/to/tests	/path/to/evidence	/path/to/db
qc		x		
cust				x,r,w
dev		r,w		
sales	-r | r,w			x,r,w
pres			r,w	

Figure 3: Matrix View

assignment and the bottom matrix shows the role-to-object permissions. Figure 3 has an example of the Matrix View.

Figure 4 shows the Hierarchy View, which consists of two parts. The Role Hierarchy Section with green background constructs a graph based on the role hierarchy. The Object Hierarchy Section with red background shows a hierarchy of objects (directories or files) in the system. Green nodes correspond to roles. Yellow nodes represent users and the red nodes are for objects. For roles, an edge is drawn from node X to node Y when the specification indicates that node X inherits node Y or when analysis of the specification indicates that there is an inheritance relationship.[1] If a policy specifies two roles satisfying the inheritance relationship, then we will connect these two role nodes via dashed line suggesting the possible hierarchical relationship. The users assigned to a role present as yellow user nodes circling around the role node.

3.2 Analysis Mode

The Matrix View presents a straightforward and natural way to view the policy. Users may click on the items in matrices to highlight or change the content of the cell of interest. In the bottom matrix, the permissions

[1] The permissions of role Y are a subset of the permissions of role X and the users of role Y are a subset of the users of role X.

of roles towards objects are listed. The content follows the format "-r| permissionset"[2] when the permission applies to the objects underneath and the cell will be highlighted in yellow. When the permissions do not apply to objects underneath, the format will simply be "permissionset".

As for the Hierarchy View, clicking on a node of interest will highlight the user and object nodes the role has access to. If the role has recursive access to an object, then that object node will be drawn in purple. Additionally, explicit read, write and execute permissions can be found in the top left corner of Object Hierarchy Section. When an user node is clicked, role it assigned to and the objects that are can be access through those roles are highlighted. Clicking on an object node will highlight the roles and users that have access to the object.

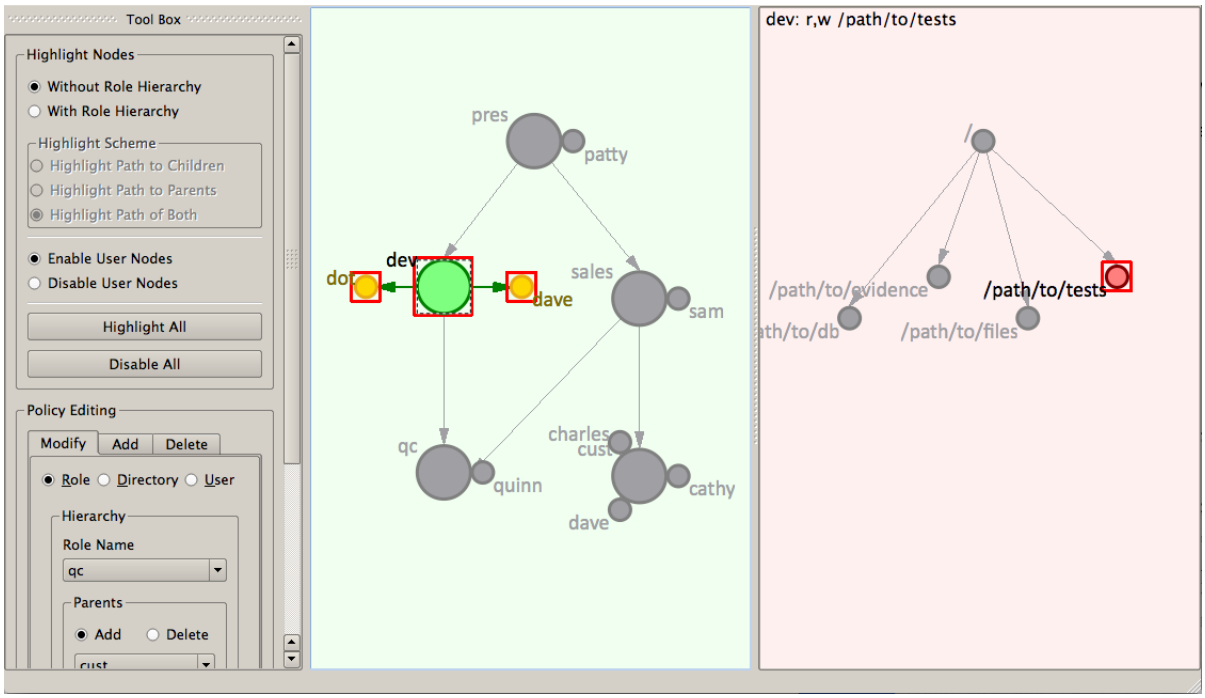


Figure 4: Role Node Highlight without Inheritance

Figure 12 shows the analysis section in the toolbox. These functions allows users to configure the role nodes being highlighted. Choices of highlighting parent nodes, children nodes and both parent and children role nodes are available. Clicking on a role node will highlight the role with some other role nodes based on the choice selected in toolbox, therefore, the highlight shows the user and objects the role has access to from itself and the roles it has inheritance relationship. Besides, user nodes are visible by default but can also be turned off to reduce the clutter in the graph. The operation detail is explained below through examples.

Figures 4 and 5 show an example of clicking on a role node. When the Without role hierarchy option is selected, each of the role nodes can be turned on and off by clicking on it. In Figure 4, role dev is turned on, therefore, its user nodes dot and dave and object node /path/to/tests are highlighted by red frames while all the other nodes are still turned off as shown in gray color.

This view also allows turning on multiple role nodes, i.e., pres and dev in Figure 5. The role node will be turned on with a red frame around the first time it is clicked, as are all the corresponding user and object nodes. When a second role node is clicked, the first role node remains on while the red frame switches to the latest clicked node. All users and objects accessible by highlighted nodes will be shown. In the example,

[2]The permissionset can be any subset of r,w,x where r stands for read, w for write, x for execute. Therefore, if granting read and write permissions, the permissionset should be "r, w".

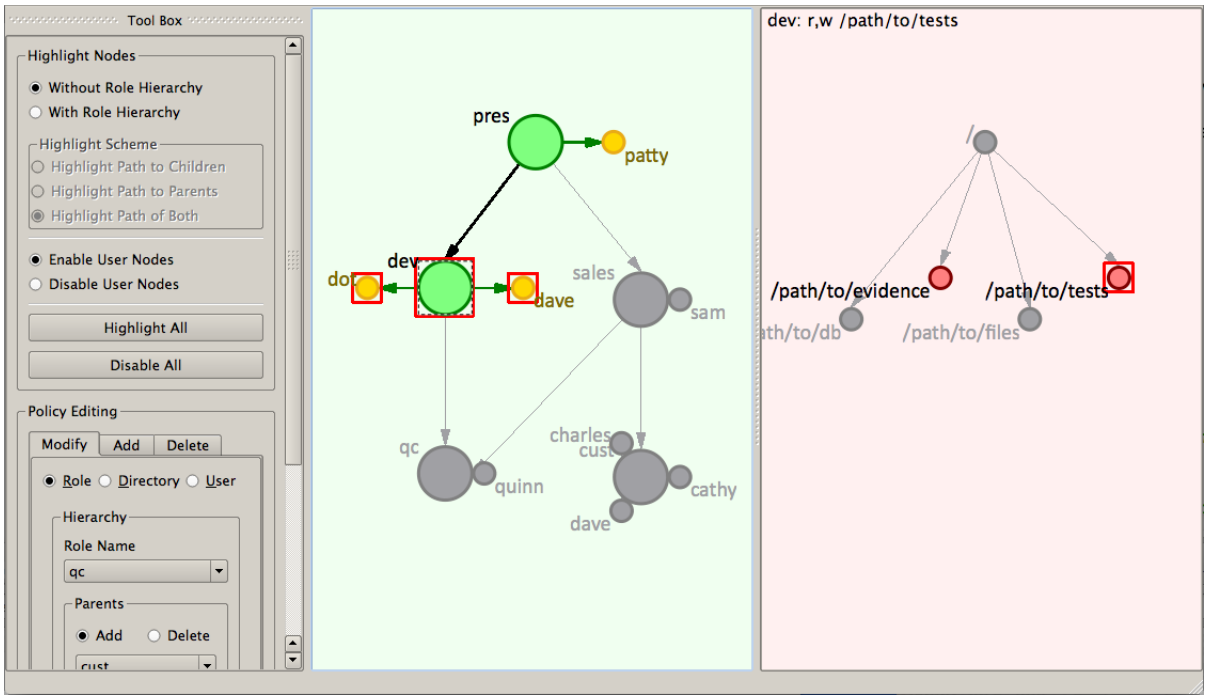


Figure 5: Multiple Role Nodes without Inheritance

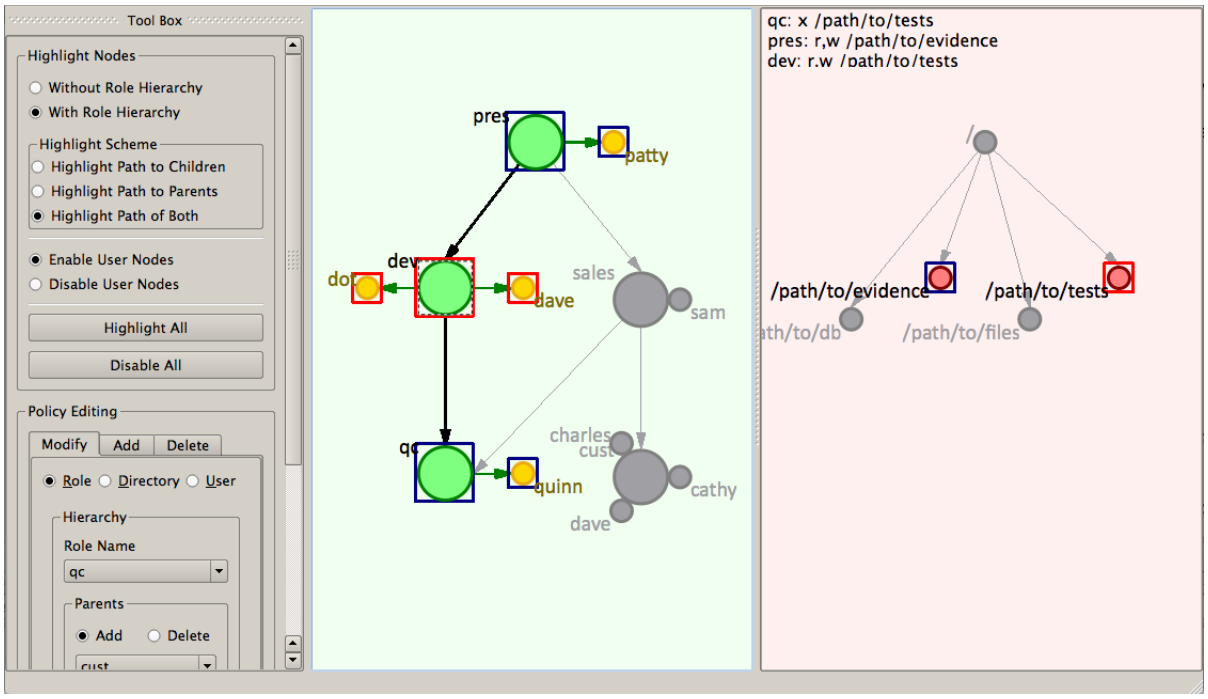


Figure 6: Role Node Highlight with Parents and Children

when pres and dev are both highlighted, objects accessible to either of these roles are highlighted. In this case, /path/to/tests (accessible from dev) and /path/to/evidence (accessible from pres) are shown and the object nodes that can be recursively accessed are drawn in purple.

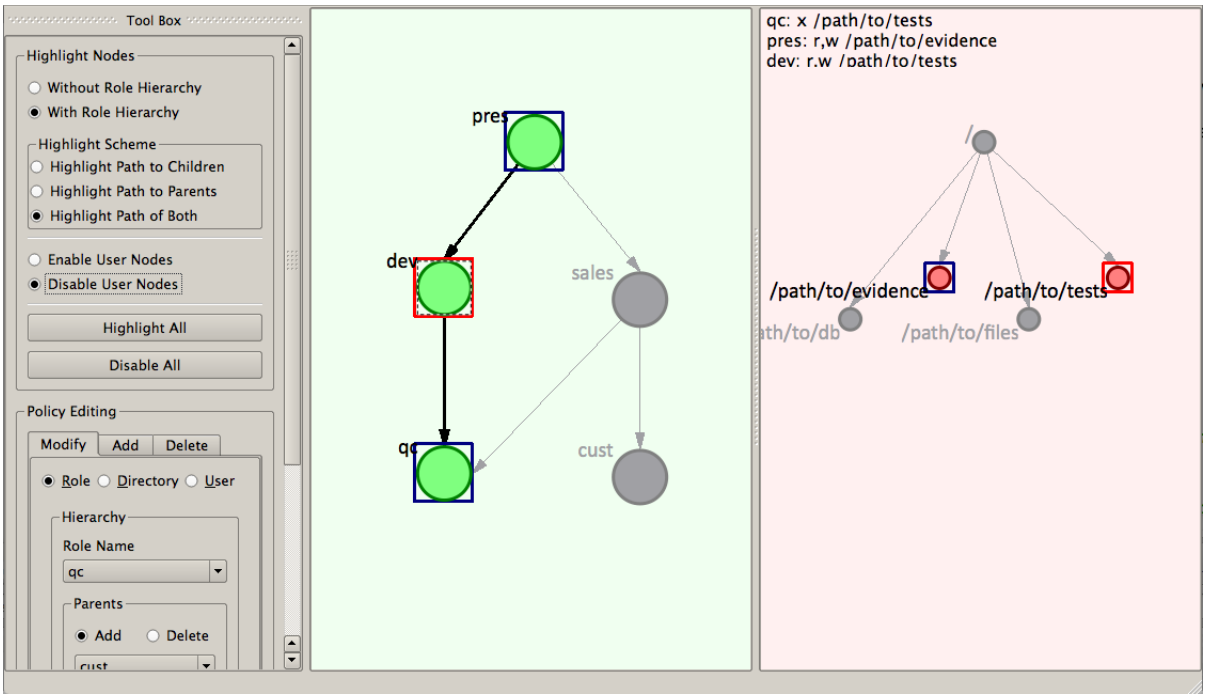


Figure 7: Role Node Highlight with Parents and Children but no User

When role inheritance is taken into consideration, it is possible to configure the Highlight Nodes section in the toolbox so that the children, parents, or both children and parent role nodes of the clicked role node are shown. Different from the mode without hierarchy, this mode only allows one role node to be selected at a time. Along with the selected node, all the desired children and/or parents role nodes will be highlighted in blue frames. User nodes and object nodes will be highlighted in red frames if directly accessible from the clicked role or in blue frames if accessible from blue framed roles. In Figure 6, the parent role prev and child role qc of role dev are within blue frames. The object /path/to/tests can be directly accessed from role dev while /path/to/evidence can be accessed through roles that have a hierarchical relationship with dev. For scenarios where permission to objects is the only interest, user nodes can be made invisible to reduce the visual clutter by choosing Disable User Nodes in toolbox.

Likewise, when an user node is clicked, the roles it assigned to and the objects accessible from those roles will be highlighted in red frames. When an object node is clicked, the roles and users that have access to the object are highlighted. Figures 10 and 11 have the examples of clicking on user node and object node.

3.3 Edit Mode

The edit section consists of three parts: Modify, Add and Delete. Figure 13 shows the edit section of the toolbox for modifying a policy. Role, User or Directory can be chosen to modify their properties. Modifying a role corresponds to changing its inheritance relationships. If Role is chosen, an existing role in the policy can be selected via the combo box under Role Name. Then its Parents and Children role nodes will be available in the combo boxes beneath for deletion. Unrelated role nodes can be added to its hierarchy from combo box selection. If Directory or User is selected in the radio button group within the Modify tab, role assignments

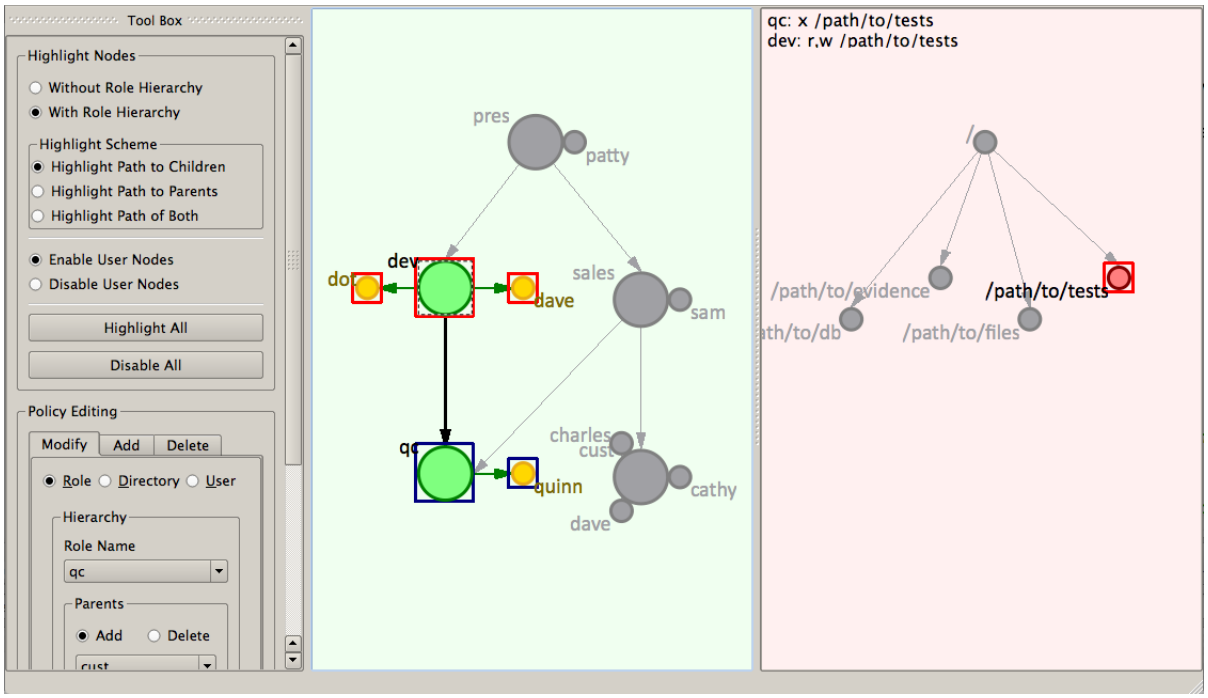


Figure 8: Role Node Highlight with Child Nodes

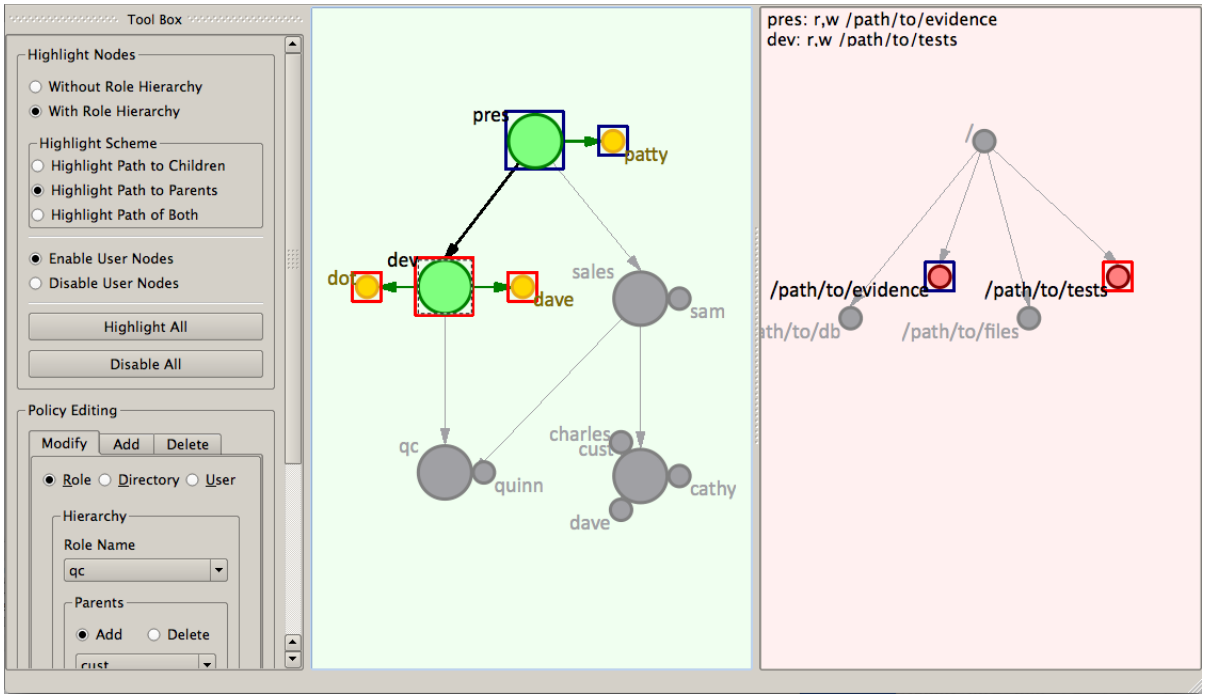


Figure 9: Role Node Highlight with Parent Nodes

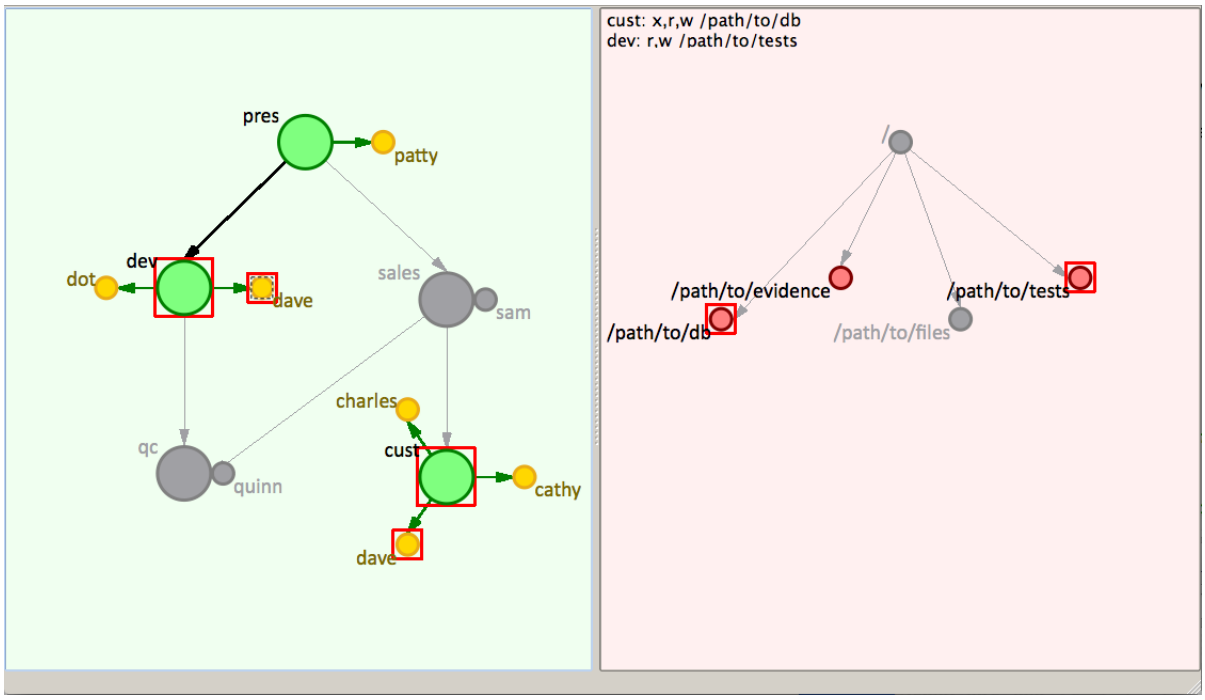


Figure 10: User Node Highlight

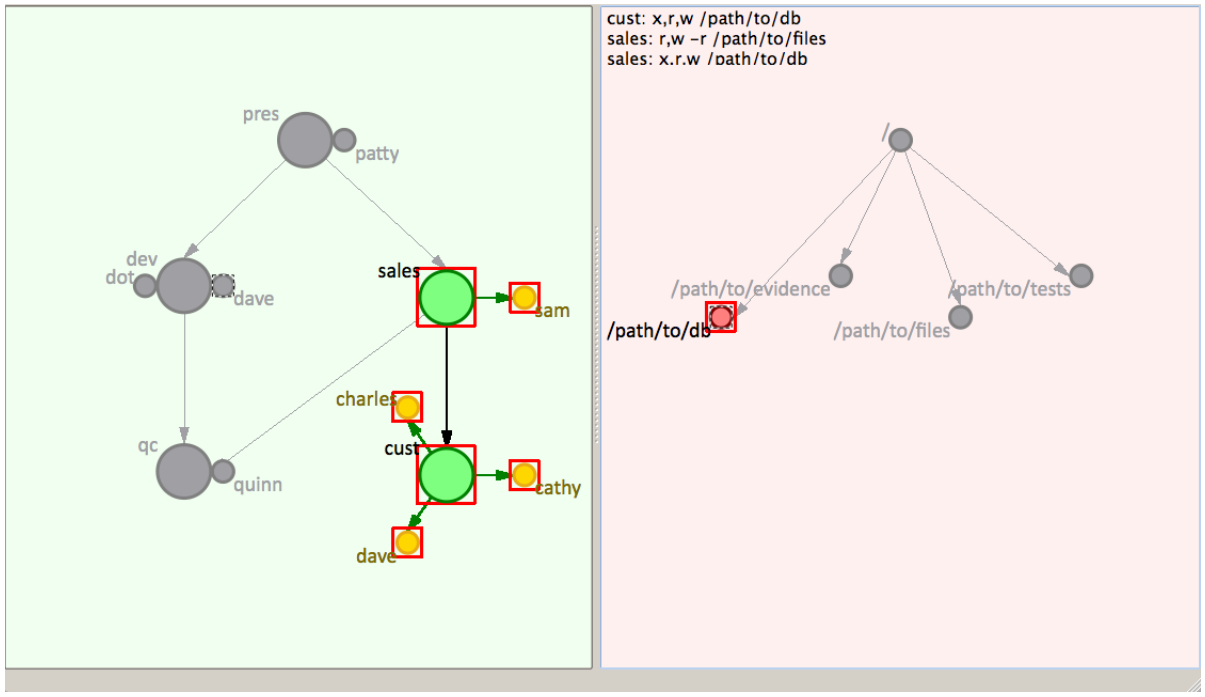


Figure 11: Object Node Highlight

(user to role, role to object) can be deleted and new role assignments can be added. Permissions can be set up via the check box group during role to object permission setup.

When an item (role, user or object) needs to be added/removed, the toolbox also have tabs as shown in Figures 14 and 15. For adding new item, the name of the item should be typed into the line edit and further association (user to role assignment or role to object permission setting) can be made in Modify tab. For deleting an item, the association it has with other items will automatically been removed.

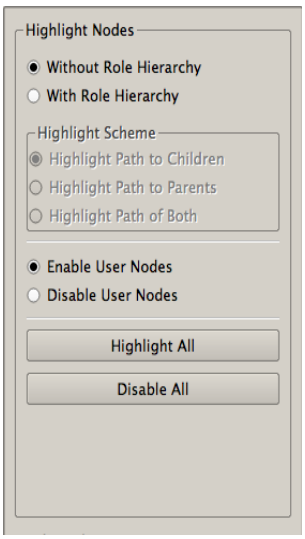


Figure 12: Toolbox Analysis Section

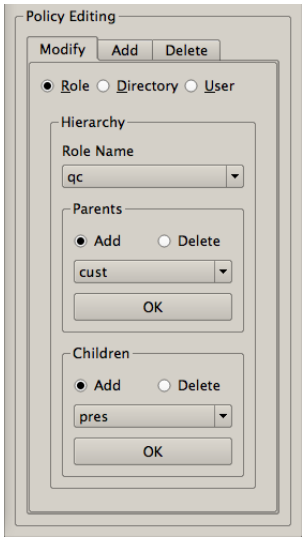


Figure 13: Toolbox Edit Section

Figure 14: Toolbox Add Element in Edit Section

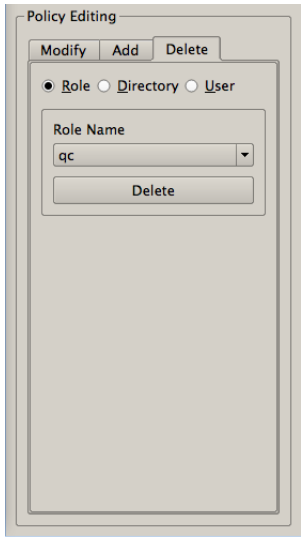


Figure 15: Toolbox Delete Element in Edit Section

4 Specification and Query

The Specification window in Figure 16 is a window that shows the text-based specification of the existing policy. It can be edited via graphical operations on views or textual edit within the window. Changes will be reflected immediately.

The Query winodw in Figure 17 contains questions commonly asked about an RBAC policy. Parameters for certain questions can be configured via the combo boxes in interface and answers to questions can be found in the bottom field with the answer to the most recent question highlighted in cyan.

5 Practice and Test

A test will be initiated by clicking on "Practice->Test". The questions are configurable so that instructors can use their own questions to achieve various teaching goals. All the questions are multi-choice questions and this section has three test modes. Instructors will have to write up a question file with "qes" extension following the format described in the Instructor Manual to specify the instructor's public key, the testing mode, the question itself along with the choices and policy files if necessary. Instructors can share the question file with the students and a test can be started by importing the question file into the system through a dialog, as shown in Figure 18.

Figure 16: Specification Window

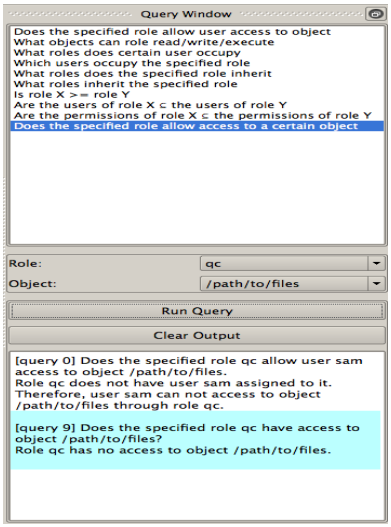


Figure 17: Query Window

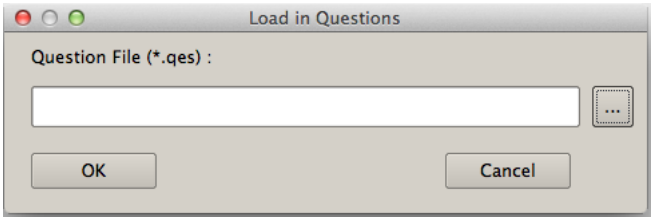


Figure 18: Quiz File Import Dialog

RBACvisual supports three testing modes. The first mode is Traditional mode where students' answers will be sent at the end of the quiz. No correct answers are given to students in this mode. The second mode is Multiple Trial Mode. In this mode, students are allowed to try multiple times until they get the correct answer to the current question. The number of attempts for each question will be stored. The last is Self-test Mode. Correct answers will be shown to the students after a choice has been confirmed for a question. The students' answers will be recorded. Figure 19 and 20 show the warnings of answering the question wrong in the last two modes.

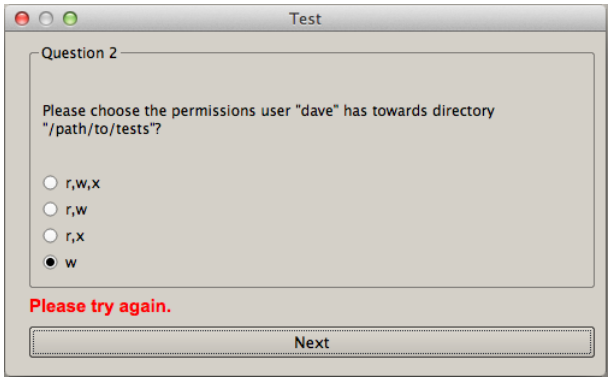


Figure 19: Multiple Trial Quiz Mode with Wrong Answer

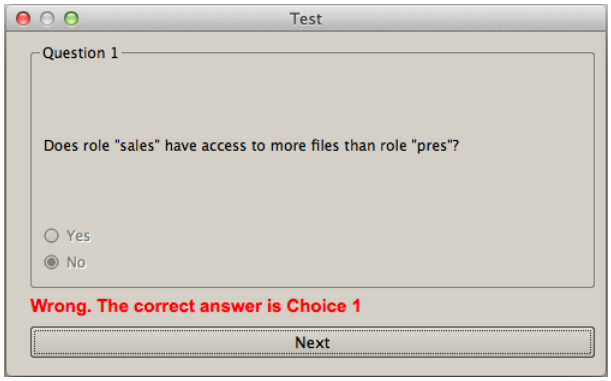


Figure 20: Self-test Quiz Mode with Wrong Answer

A dialog (Figure 21) of confirming the submission of answers will show up as the last step of the test. It will use Thunderbird to send answers out. If Thunderbird is not installed, a warning dialog, as in Figure 22, will show up indicating where the answer file is stored and the student will be able to send the email manually. In this case, the answer file will be encrypted using the instructor's public key to prevent any manual change.

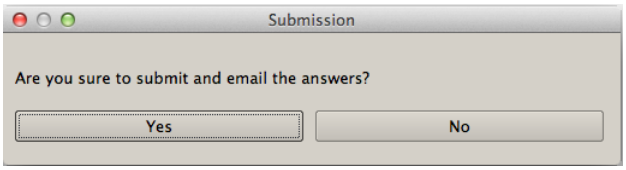


Figure 21: Submission Confirmation

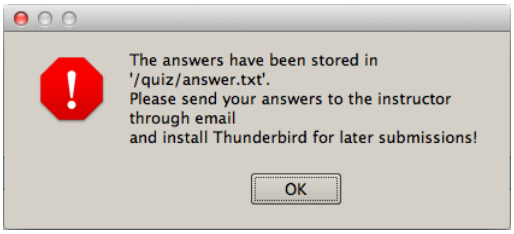


Figure 22: Submission Warning