Technical Guide to domRBAC Simulator

Installation and operation instructions

Document version: 0.1

Antonios Gouglidis

{agougl@uom.gr}

Basic Research Project

Title: "Development of a new improved role-based model for controlling access to Web services in Grid/Cloud computing environments".

Scientific coordinator: Associate Professor Ioannis Mavridis, Department of Informatics, University of Macedonia, Greece.

Research collaborator: Antonios Gouglidis

Acknowledgements

The author, Antonios Gouglidis, wish to acknowledge the Research Committee of the University of Macedonia in Greece for funding this research.

Disclaimer

Table of Contents

1. Introduction

This technical document provides information regarding the installation and operation of the domRBAC simulation application version 0.0.1, which is an implementation of the domRBAC access control model proposed in [1] and the model checking technique proposed in [2]. Therefore, this technical guide of the simulator is an integral part of the manuscripts in [1] and [2].

2. Prerequisites

The domRBAC simulation has been implemented and tested only in the Linux operating system (i.e., Debian Squeeze). Nevertheless, it should compile and operate in most major UNIX and Linux distributions.

The following software is required for the compilation and correct operation of the domRBAC simulator:

- g++ 4.x
- Qt 4.x
- QMake version 2.x
- Boost C++ Libraries 1.42.x
- Graphviz
- python
- python-networkx

3. Installation

To install the domRBAC simulator extract the source code and compile it from a terminal, as follows:

```
1. user@system:~/directory$ tar –xzvf domRBAC-src-0.0.1.tar.gz
2. user@system:~/directory$ cd domRBAC-src-0.0.1
3. user@system:~/directory/domRBAC-src-0.0.1$ make
```

After successfully compiling the application, execute the application from a terminal, as follows:

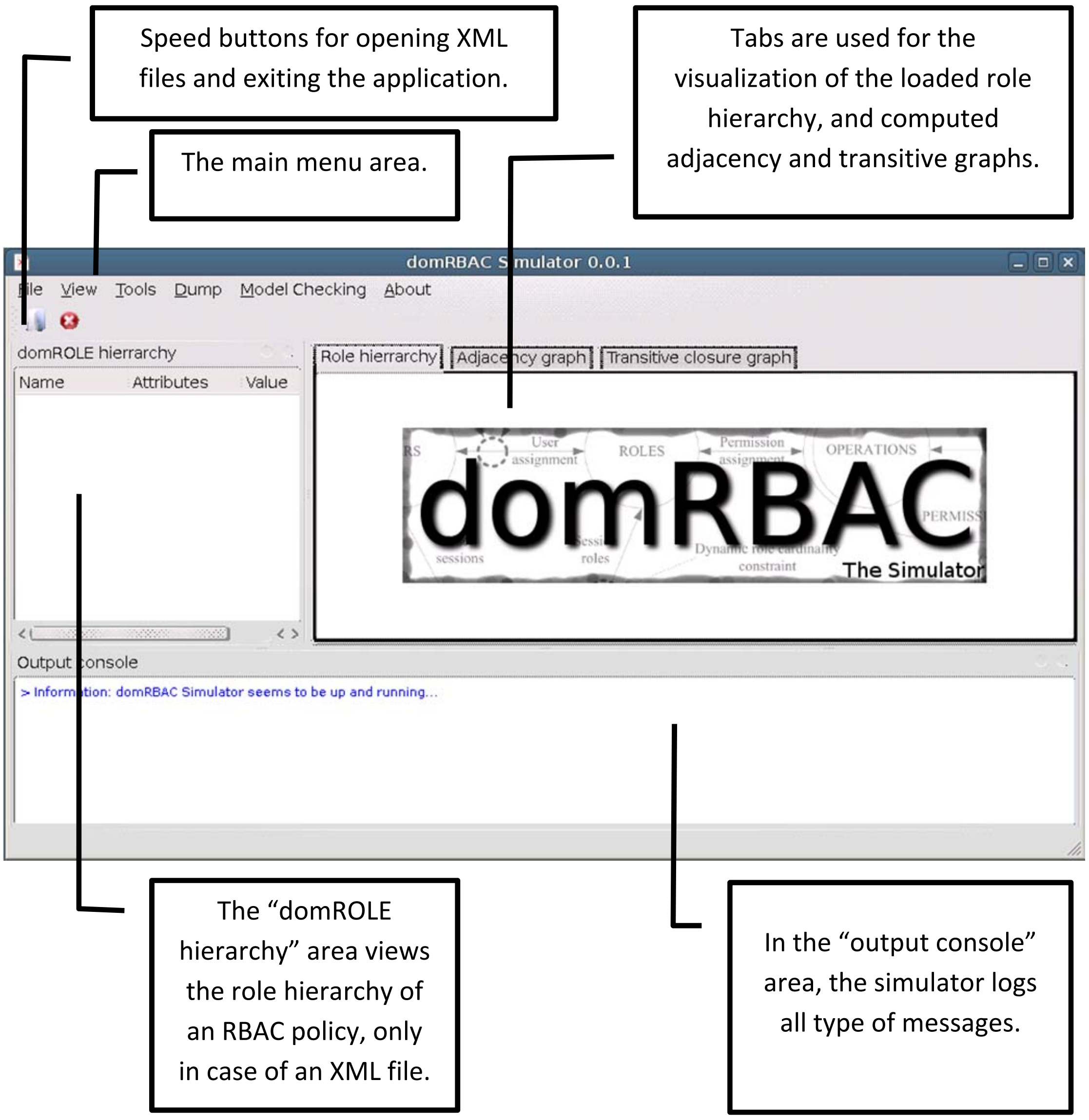
```
4. user@system:~/directory/domRBAC-src-0.0.1$ ./domRBACsim
```

An alternative way to compile the source code is by loading the "`domRBACsim.pro`" Qt project file using the Qt Creator (bundled with Qt) [5]. To compile the source code select from the menu **Build → Build All**, and then to execute the application select from the menu **Build → Run**.

4. The domRBAC simulator

In this section, the reader can be informed about the capabilities and operations of the domRBAC simulator.

4.1. Main Screen



```
> Information: domRBAC Simulator seems to be up and running...
```

Speed buttons for opening XML files and exiting the application.

The main menu area.

Tabs are used for the visualization of the loaded role hierarchy, and computed adjacency and transitive graphs.

The “domROLE hierarchy” area views the role hierarchy of an RBAC policy, only in case of an XML file.

In the “output console” area, the simulator logs all type of messages.

4.2. Loading RBAC policies

The domRBAC simulator supports loading of RBAC policies using:

- XML files
- Dot files

4.2.1. XML files

To load an RBAC policy from an XML file, select from the menu **File → Open** or click the open file speed button. Use the open dialog to browse and select the XML file to open, and click on the **Open** button.

To create valid XML files, use and follow the “`domRBAC.xsd`” file included in the source code directory. A description of the file is provided in Appendix A.

4.2.2. Dot files

To load an RBAC policy from a dot file, select on the menu **File → Load external hierarchy**. Use the open dialog to browse and select the dot file to open, and click on the **Open** button. Repeat the procedure to load multiple dot files, were each one of them is interpreted as an RBAC policy of a single domain.

To randomly create valid dot files use NetworkX, which is a python language software package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks [3]. An example of a python script for the creation of valid dot file for the domRBAC simulator is provided in Appendix B.

4.3. The View menu

The View menu includes the following operations:

- **domROLE hierarchy**: toggles the domROLE hierarchy area.
- **Output console**: toggles the output console area.
- **Reload images**: reloads the graph images in all tabs, if any.
- **Zoom out image**: Zooms out the graph image in the currently selected tab.
- **Zoom in image**: Zooms in the graph image in the currently selected tab.

4.4. The Tools menu

The Tools menu operates on a step-by-step basis for the creation of the adjacency and transitive closure graphs, and the checking of a series of violations.

1. **XML to dot**: It converts the loaded XML file into a dot file. (Enabled but not operational when loading RBAC policies from dot files)
2. **dot to image**: Enabled only after completing step 1. It converts the dot file into an image file and views it in the Role hierarchy tab. (Enabled but not operational when loading RBAC policies from dot files)
3. **Create adjacency matrix**: Enabled after completing step 2. It creates the adjacency matrix of the loaded RBAC policy, and loads an image of it on the Adjacency graph tab.
4. **Create transitive closure**: Enabled after completing step 3. It creates the transitive closure of the adjacency matrix, and loads an image of it on the Transitive closure tab.
5. **Inter-domain role assignment violations**: Enabled after completing step 4. It checks for cyclic inheritance violations
6. **SSD violations**: Enabled after completing step 3. It checks for SSD violations.
7. **DSD violations**: Enabled after completing step 3. It checks for DSD violations.
8. **Performance metrics**: Prints on the output console a number of metrics regarding the running RBAC policy (i.e., roles, violations, memory consumption etc.)
9. **Start simulation**: Enabled after completing step 4. Starts the simulation process. The simulation parameters are described in the `“domRBACsim.conf”` file (see Appendix C). All simulated information is logged in file `“results.txt”`.

4.5. The Dump menu

The Dump menu includes the following operations, and is mostly used for debugging reasons and advanced analysis of the simulated RBAC policies:

- **Dump original AG and TCG**: It clears any existing information in both adjacency and transitive closure lists, and recreates them. Execute only if it is required to reset the aforementioned lists after a simulation process ends.
- **Hierarchy to dot**: Execute after step 9 described in subsection 4.4 to convert the running hierarchy into a dot file and view it in the Role hierarchy tab.
- **Adjacency to dot**: Execute after step 9 described in subsection 4.4 to convert the running adjacency matrix into a dot file and view it in the Adjacency graph tab.

- **Transitive to dot**: Execute after step 9 described in subsection 4.4 to convert the running transitive closure into a dot file and view it in the Transitive closure graph tab.
- **SSD to console**: Execute after step 9 described in subsection 4.4 to view in the Output console the running SSD relations.
- **DSD to console**: Execute after step 9 described in subsection 4.4 to view in the Output console the running DSD relations.
- **Top roles console**: Execute after step 9 described in subsection 4.4 to view in the Output console the id of the roles that are on top of the hierarchies.

4.6. The Model Checking menu

The **Model Checking → Export to ACPT** selection on the menu should be executed only after step 9 that is described in subsection 4.4. It exports the running RBAC policies and security properties generated by the simulation process in valid XML format for the ACPT tool [4] as described in [2]. The XML file is saved under the running directory. The name of the XML file is "`Exported_domRBAC.xml`".

4.7. The About menu

The About menu provides information about the version of the application.

5. References

[1] Antonios Gouglidis, Ioannis Mavridis, "domRBAC: An Access Control Model for Modern Collaborative Systems". Computers & Security, Volume 31, Issue 4, June 2012, Pages 540-556, ISSN 0167-4048, http://dx.doi.org/10.1016/j.cose.2012.01.010.

[2] Antonios Gouglidis, Ioannis Mavridis, Vincent C. Hu, "Security policy verification for multi-domains in cloud systems". International Journal of Information Security (IJIS) Springer, 2013, (DOI) 10.1007/s10207-013-0205-x

[3] NetworkX, http://networkx.github.io/

[4] NIST, Access Control Policy Tool (ACPT), http://csrc.nist.gov/groups/SNS/acpt/acpt-beta.html

[5] Qt Project, http://qt-project.org/

Appendix A. The XSD file

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2011 rel. 2 sp1 (http://www.altova.com) by John (Smith) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
	<xs:element name="DomainRole_Graph">
		<xs:complexType>
			<xs:sequence minOccurs="0" maxOccurs="unbounded">
				<xs:element name="Organization"/>
				<xs:element name="DomainRole" minOccurs="0"
maxOccurs="unbounded"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="Organization">
		<xs:complexType>
			<xs:sequence>
				<xs:element name="Org_Name" type="xs:string"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="Org_Name">
		<xs:complexType mixed="true"/>
	</xs:element>
	<xs:element name="DomainRole">
		<xs:complexType>
			<xs:sequence>
				<xs:element name="Name" type="xs:string"/>
				<xs:sequence minOccurs="0" maxOccurs="unbounded">
					<xs:element name="Inter_Parent_Role"
type="xs:string" minOccurs="0" maxOccurs="1"/>
				</xs:sequence>
				<xs:sequence minOccurs="0" maxOccurs="unbounded">
					<xs:element name="Inter_Child_Role"
type="xs:string" minOccurs="0" maxOccurs="1"/>
				</xs:sequence>
				<xs:sequence minOccurs="0" maxOccurs="unbounded">
					<xs:element name="Intra_Parent_Role"
type="xs:string" minOccurs="0" maxOccurs="1"/>
				</xs:sequence>
				<xs:sequence minOccurs="0" maxOccurs="unbounded">
					<xs:element name="Intra_Child_Role"
type="xs:string" minOccurs="0" maxOccurs="1"/>
				</xs:sequence>
				<xs:sequence minOccurs="0" maxOccurs="unbounded">
					<xs:element name="SSD_Role" type="xs:string"
minOccurs="0" maxOccurs="1"/>
				</xs:sequence>
				<xs:sequence minOccurs="0" maxOccurs="unbounded">
					<xs:element name="DSD_Role" type="xs:string"
minOccurs="0" maxOccurs="1"/>
				</xs:sequence>
				<xs:element name="SR_Cardinality" type="xs:unsignedLong"
minOccurs="0" maxOccurs="1"/>
				<xs:element name="DR_Cardinality" type="xs:unsignedLong"
minOccurs="0" maxOccurs="1"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="Name">
		<xs:complexType mixed="true"/>
	</xs:element>
	<xs:element name="Inter_Parent_Role">
		<xs:complexType mixed="true"/>
	</xs:element>
	<xs:element name="Inter_Child_Role">
		<xs:complexType mixed="true"/>
	</xs:element>
	<xs:element name="Intra_Parent_Role">
		<xs:complexType mixed="true"/>
	</xs:element>
	<xs:element name="Intra_Child_Role">
```

```
                <xs:complexType mixed="true"/>
        </xs:element>
        <xs:element name="SSD_Role">
                <xs:complexType mixed="true"/>
        </xs:element>
        <xs:element name="DSD_Role">
                <xs:complexType mixed="true"/>
        </xs:element>
        <xs:element name="SR_Cardinality">
                <xs:complexType mixed="true"/>
        </xs:element>
        <xs:element name="DR_Cardinality">
                <xs:complexType mixed="true"/>
        </xs:element>
</xs:schema>
```

Appendix B. Python script for the creation of dot files

```
import networkx as nx

DG=nx.DiGraph()

print ('Network 1')
DG=nx.gnr_graph(40, 0.001)
DG=DG.reverse(True)
nx.draw_graphviz(DG)
nx.write_dot(DG,'ext_dot_gnr_20_1_RE.dot')

print ('Network 2')
DG=nx.gnr_graph(40, 0.2)
DG=DG.reverse(True)
nx.draw_graphviz(DG)
nx.write_dot(DG,'ext_dot_gnr_20_2_RE.dot')
```

Appendix C. Configuration file for the simulation process

```
#Number of roles in line 2
100000
#Percentage to create a new role [0..1]
0
#Percentage to create a new domain [0..1]
0
#Percentage to create a top role [0..1]
0
#Enable SSD and DSD after number of roles
1
#Percentage increment intra/inter-role assignment [0..1] in line 4
1
#Percentage increment SSD [0..1] in line 6
0.01
#Percentage increment DSD [0..1] in line 6
0.01
#Number of maximum iterations (0: infinite)
5000
```