

ERBAC: Event Driven Role Based Access Control

Prototype Quick User Guide

Piero Bonatti, Clemente Galdi and Davide Torres

January 05th, 2014

Contents

1	Introduction	2
2	Quick start: Running the examples	2
3	Prototype basics	2
3.1	Input Description	2
3.2	Spatial Hierarchy	3
3.3	List of Commands	3
3.4	Access Policy	3
4	Output Specification	5

1 Introduction

This document describes the execution environment of the prototype implementing the ERBAC model described in [1].

In the following we assume ERBACDIR to be the directory in which the files in the package have been extracted. Such directory should include the following:

- `erbac.jar`: A jar file containing the prototype implementation
- `policy_translator.jar`: A utility needed to translate policies written using a “user-friendly” language into xml-coded policies
- `log4j.properties`: A configuration file
- Subdirectories `EXAMPLES1/`: A directory containing a set of predefined examples.

2 Quick start: Running the examples

The prototype is provided with a set of pre-defined examples. The prototype takes as input the name of the directory containing the required files. In order to run the example provided in the directory `Example1`, just type the following:

```
~> cd ERBACDIR
~/ERBACDIR> java -jar erbac.jar EXAMPLES/Example1
```

3 Prototype basics

The prototype is an extension of the SUN XACML implementation. It basically loads a definition of the space hierarchy, the access policy and a sequence of ‘commands’ that are meant to simulate a sequence of environmental conditions and/or user movements. In order to ease the generation of XML-style files needed by the prototype, we have defined an *ad-hoc* specification language that is then translated, by means of the `policy_translator` utility, in the appropriate XML format.

3.1 Input Description

The prototype takes as inputs the following information:

- **Spatial Hierarchy**: This is stored in a file that **must be named** `locations`. The syntax is specified in the next sections.
- **Access Policy**: The access policy is split into two different files. Such files can be automatically generated using the utility `policy_translator` described in Section 3.4.
 - `policy`: This file contains the definitions of role templates and privileges
 - `enablingPolicy`: This file contains the definition of enabling/disabling rules (role enabling base).
- **List of commands**: This file, named `commands` contains the sequence of operations to be executed.
- **Logging parameters**: Stored in the file named `log4j.properties`, these include the file name that will contain logs and logging level.

3.2 Spatial Hierarchy

The description and organization of the spatial hierarchy has to be specific in a file named **locations** using the syntax reported in Table 1. It is possible to define location types, along with relations between different types, and logical locations that describe the space.

Table 1: Location syntax

Syntax:	createType::typename
Description:	Creates a new location type named 'typename'
Example:	createType::department createType::operatingRoom

Syntax:	createTypeRelation::t1>t2
Description:	Creates a new relation between types t1 and t2, where t2 is more specific than t1.
Example:	createTypeRelation::department>operatingRoom

Syntax:	createLocation::name::type::coordinates coordinates= coord {';' coord} coord=num,num
Description:	The field coordinated contains a list of points that identify a polygon. This command creates a new logical location named ' name ' of type ' type ', that is associated to all the physical locations contained in the polygon described by the list ' coordinates '. Important: The first and the last element in coordinates have to be the same.
Example:	createLocation::cardiology::department::0,0;20,0;20,20;0,20;0,0

IMPORTANT: The file **locations** has to define location types and relations among locations types before the definition of logical locations.

3.3 List of Commands

The prototype will execute, one by one, all the commands contained in a text file named **commands**. Each line in the file must contain exactly one command, written using the syntax in Tables 2 and 3.

3.4 Access Policy

The access policy should be specified by means of two XML files:

- **policy**: This file contains role template and privileges definitions
- **enablingPolicy**: described the role enabling base.

In order to ease the write-up of access policies, we provide a utility named **policy_translator.jar**. This utility takes as input the pathname of the file containing the description of the policy (including role/permissions definitions and role enabling base) written using a user-friendly language (defined using the specification language described below) and outputs the XML files.

The output files will be written in the same directory in which the source file is stored. Following is an example:

Table 2: Commands syntax

Syntax:	createDevice::username::coordinates::roleinstances roleinstance=roledef[;roleinstance] roledef=rolename([parameterdef]) parameterdef={vaname:vartype:value[,parameterdef]}
Description:	Creates a new device for user username , at location coordinates assigned to role instances specified by the list of instances roleinstances . Each role instance might have a (possibly empty) list of parameters.
Example:	<code>createDevice::Dr.Doe::10,5::doctor(department:string:cardiology);gendocor()</code>

Syntax:	createResource::resourcename::coordinates::categories categories=category_name[,categories]
Description:	Creates a new resource named resourcename , located at coordinates , belonging to the categories defined in the list categories .
Example:	<code>createResource::patientMonitor::30,10::medicalDevices</code>

Syntax:	moveUser::username::newcoordinates newcoordinates= num,num
Description:	Updates the position of user username to newcoordinates
Example:	<code>moveUser::Dr.Doe::15,5</code>

Syntax:	createEvent::priority::name::genBy::genFor::coordinates priority={0 1 2 3 4 5 6 7 8 9} genby=string 'null' genfor=string 'null' coordinates=num,num
Description:	Creates an event named name , with priority priority . The other parameters are interpreted as follows: <ul style="list-style-type: none"> • genBy: The user name of the user who generated it. Use 'null' for system generated events. • genFor: The user name to which the event is addressed. Use 'null' for general events. • coordinates: The physical location in which the event is generated. Use 'null' for global events. .
Example:	<code>createEvent::0::medicalEmergency::patientFoo::10,10</code>

Syntax:	removeEvent::priority::name::genBy::genFor::coordinates
Description:	Removes the event identified by the parameters
Example:	<code>removeEvent::0::medicalEmergency::patientFoo::10,10</code>

Table 3: Syntax

Syntax:	requestAccess::user::role::oper::resource::datetime[:property] property=property_def[:property] property_def=property_name:{string=value point=num,num}
Description:	Simulates an access request to the PDP with the following parameter: <ul style="list-style-type: none"> • user: username requesting the access • role: role name used to gain access. It is necessary to specify the role name and omit the parameters, e.g., <code>doctor<department></code> becomes <code>doctor</code>. • oper: identifies the operation for which the user requires the access. • resource: name of the resource on which the operation has to be executed. • date time: Date and time in which the operation access request should be executed. If 'null' the system assumes the current date and time, otherwise a date in the format <code>dd/mm/yyyy hh:mm</code> is expected. • property (optional): List of additional properties to provide to the PDP.
Examples:	<pre>requestAccess::Dr.Doe::doctor::read::patientMonitor:: 3/5/2011 11:00::device-of:string=patientFoo</pre> <pre>requestAccess::Dr.Nick::assistant::read::medicalRecord::null::patient.position:point=3,3</pre>

```
~> java -jar policy_translator.jar EXAMPLE1/human-readable-policy1
Policy files succesfully created:
->EXAMPLE1/enablingPolicy
->EXAMPLE1/policy
```

IMPORTANT: Temporal expressions and role templates must be defined before enabling/disabling rules.

4 Output Specification

The prototype uses log4j for logging activities. The results of a single execution is written in the file `global.log`. It is possible to set two different levels of logging, by modifying the attribute value in of the XML element `tt` value in the file `log4j.properties`.

```
<root>
<level value="info"/>
<appender-ref ref="APPENDER_FILE" />
</root>
```

Temporal Expressions	
Syntax:	'TimeExp' Symbolic_name = (Interval , Periodic_Expression) 'TimeExp' Symbolic_name = (Interval ',') 'TimeExp' Symbolic_name =(',' Periodic_Expression) Interval=(begin, end) (begin,) (,end) begin, end= dd/mm/yyyy hh:mm Periodic_Expression = { Oi } * Calendar [+ { Oi } * Calendar...] > {Od} * Calendar Oi = index[; index] index = num num-num Od=num Calendar = Years Months Weeks Days Hours
Examples:	TimeExp Ex1 = ((12/1/2010 00:00; 20/1/2010 02:20), {1}*Months+{2;5}*Days+{8-12;15-18}*Hours>{2}*Hours) TimeExp Ex2 = ((12/1/2010 00:00; 20/1/2010 02:20),) TimeExp Ex3 = ((; 20/1/2010 02:20), {1}*Months+{2;5}*Days+{8-12;15- 18}*Hours>{2}*Hours) TimeExp Ex4 = ((12/1/2010 00:00;), {1}*Months+{2;5}*Days+{8-12;15- 18}*Hours>{2}*Hours) TimeExp Ex5 = (, {1}*Months+{2;5}*Days+{8-12;15-18Hours>{2}*Hours)

Table 4: Specification of Temporal Expressions

The possible values for the attribute value are `info` and `debug`.

References

- [1] P. Bonatti, C. Galdi, and D. Torres. ERBAC: event-driven RBAC. In *Proceedings of the 18th ACM symposium on Access control models and technologies, SACMAT '13*, pages 125–136, New York, NY, USA, 2013. ACM.

Specification of Conditions over Permission	
true	true expression
ValueMatch(property:name1, parameter:name2)	True if property:name1=parameter:name2 name1: name of the attribute being accessed name2: role template parameter
EventVisible(event:evname)	True if the event evname is visible
EventVisibleIn(parameter:locname, event:evname)	True if the event evname is visible in location locname
EventGeneratedBy(event:evname, property:username)	True if the event evname is visible and generated by user username
PositionIn(property:pos,parameter:parname)	True if pos belongs to the location identified by template parameter parname. pos should be a property of type 'geometry'.
AreNear()	True if subject and resource are close
AreNear(property:pos)	True if the position of the subject is close to the one defined by pos

Table 5: Permission Expression Elements

Specification of Role Templates	
Syntax:	<pre> Role_Name = role (Perm [; Perm]). Role_Name = Symbolic_Name < [Parameters] > Parameters = Param_Name [, Parameters] Perm = (Operation_Name, Category_Name, Expression) Role_Name Expression = Element [and Expression] Categories=Category_name[,Categories] </pre>
Examples:	
	<pre> Doctor<dept,hospital> = role((read,PatientRecs,ValueMatch(property:recdept,parameter:dept)); (write,PatientRecs,EventVisible(event:medEmergency)); (read,PatientRecs,EventVisibleIn(parameter:hospital,event:medEmergency)); (read,PatientRecs,EventGeneratedBy(event:medEmergency,property:MrFoo)); (read,PatientRecs,PositionIn(property:patient_pos,parameter:operRoom)); (read,PatientRecs,AreNear(property:pos)); (read,PatientRecs,AreNear())). </pre>
	<pre> GeneralDoc<> = role((read,Category1,true);(write,Category1,true)). Doctor<dept> = role(GeneralDoc<>; (read,Category2,true)). </pre>

Table 6: Policy Specification: Roles

Enabling/Disabling rules	
Syntax:	<pre> (Tcond , Scond , Econd), Action, Role_Instance Tcond = 'AnyTime' Temporal_Expression_Name Scond = 'AnyPlace' Logical_Location_Name Location_Type_Name:lt Econd = 'AnyEvent' Event_Name Action = 'enable' 'disable' Role_Instance = Role_name<Parameters> Parameters = Param [' ',' Parameters] Param = Param_Value 'Any' </pre>
Examples:	<pre> (AnyTime,loc1,criticalPatient), enable Doctor<cardiology> (WorkingHours,loc1,AnyEvent), enable Doctor<cardiology> (AnyTime,loctype:lt,AnyEvent), enable Doctor<cardiology> (AnyTime,department:lt,criticalPatient), enable Doctor<Any> (AnyTime,department:lt,criticalPatient), enable Doctor<Any,Any> </pre>

Table 7: Policy Specification