Simple PERMIS Java API Cookbook

Authors: Chadwick D., Laborde R., Otenko O, Zhao G.
Date: 16 January 2005
Version 1.0_beta_2

Contents

Simple PERMIS Java API Cookbook....1
Contents....1
1. Objective of this document....1
2. Overview of PERMIS & Simple PERMIS....1
    a. PERMIS Decision Engine Architecture....2
    b. SimplePERMIS....3
3. Installation of SimplePERMIS....4
    a. Requirements....4
    b. Installation....4
4. Play with SimplePERMIS....4
6. How to use SimplePERMIS in applications of your own?....6
References....7

1. Objective of this document

This cookbook provides a step-by-step example for using the Simple PERMIS Java API to the PERMIS decision engine. On completion of this tutorial, you will be able to:

- Install/Setup all necessary components for using the Simple PERMIS API.
- Run example code with the Simple PERMIS API.
- Understand how to use the Simple PERMIS API for authorisation decision making in applications of your own.

2. Overview of PERMIS & Simple PERMIS

PERMIS is a policy based authorisation system, a Privilege Management Infrastructure. It can work with any and every authentication system (Shibboleth, PAPI, Kerberos, PKI, username/PW, etc.). Given a username, a target and an action, the PERMIS decision engine says whether the user is granted or denied access based on the policy for the target. The policy is role/attribute based i.e. users are given roles/attributes and roles/attributes are given permissions to access targets. The policy is written in XML, and it is similar to XACML but simpler, and it can be produced more easily using the PERMIS Policy Editor graphical user interface.

The PERMIS decision engine can work in push mode (attributes are sent to PERMIS by the application) or pull mode (PERMIS fetches them itself given the distinguished name of the user by the application).

a. PERMIS Decision Engine Architecture

The PERMIS decision engine is built using a modular approach in order to be generic (like when you want an ice cream and you ask the ice cream man for a double chocolate banana in his special chocolate cone) so that you can customise the PERMIS decision engine to get your own privilege management infrastructure specific to your own requirements (see Figure 1).

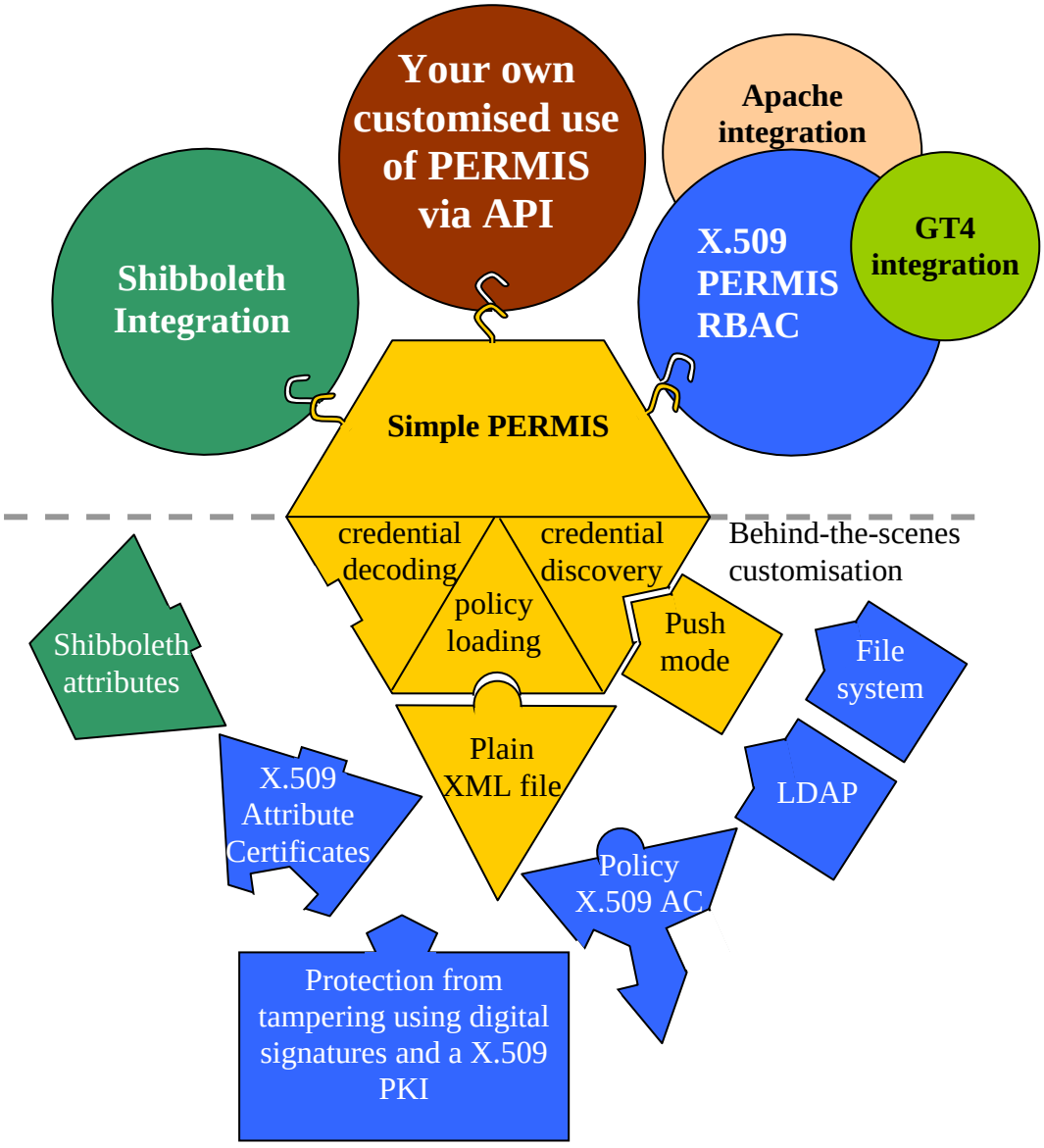

Figure 1. PERMIS Decision Engine architecture

The two main components that can be customised are the retrieval and processing of credentials and access control policies.

Credentials are the roles/attributes assigned to users by Attribute Authorities. The value of the roles/attributes can be presented in different formats depending on the application or the security technologies. The Simple PERMIS decision engine accepts plain vanilla attributes without any security protection. Currently, the modular PERMIS decision engine includes two extensions: one to decode SAML Attributes Assertions and another to decode X.509 Attributes Certificates. These modules check the validity, retrieve the role/attribute value, and return the credential in a PERMIS standardized format. They provide the role value and type, the role validity period and also the sub-roles in the role hierarchy, which are specified in the RBAC policy. (Sub-roles allow a user to benefit from inherited permissions).

In addition, the PERMIS decision engine can obtain credentials according to two modes:

- The *push mode* where credentials are given to PERMIS by the application,
- The *pull mode* where PERMIS retrieves the credentials itself, given the distinguished name of the user by the application..

In pull mode, the credentials need to be stored in repositories. The PERMIS decision engine provides interfaces for retrieving the credentials from any type of repository, and comes built-in with implementations to access LDAP repositories and local file systems repositories.

Policies state who is trusted to allocate which roles to whom, and what permissions are assigned to the roles/attributes. Policies can be specified in a plain XML file or in an XML formatted attribute embedded in an X.509 Attribute Certificate to secure it from being tampered with. Policies can be written and stored in an LDAP repository or the local file system, or passed dynamically via the PERMIS API.

More information about the PERMIS architecture can be found in http://sec.cs.kent.ac.uk/permis/.

b. SimplePERMIS

SimplePERMIS represents the core of the PERMIS decision engine (the yellow pieces in Figure 1). It provides the core access control service (i.e. authorisation decision-making). It works in push mode only and the policy is stored in a plain XML file. It can be considered as a lightweight PERMIS decision engine with the decoupling of credential and policy protection implementations.

In this case any security features – like the protection of the policy or the user credentials - should be provided by external technologies (e.g. the file access control mechanism or SSL). SimplePERMIS is not released with the credential and policy protection implementations, but SimplePERMIS keeps the interfaces for verifying credentials and policies. Users can implement their own protection mechanisms as plug-ins for SimplePERMIS, which is a major feature for SimplePERMIS to be agile and generic. In this way, SimplePERMIS can be customized to fit in most of the application scenarios and be aligned with the chosen technologies by uses.

3. Installation of SimplePERMIS

a. Requirements

To run the example PERMIS gatekeeper application it is only necessary to download the Cookbook example:

- Cookbook Example – the example software used in this cookbook. [http://sec.cs.kent.ac.uk/permis/]
- Java Runtime Environment (JRE) – needed to run the SimplePERMIS example. Please note, if you wish to modify the software, you will need the Java SDK available from http://java.sun.com.

To develop your own application using SimplePERMIS it is necessary to download and install the following packages:

- Java SDK (version 1.5.0 or later) – needed to run and modify the software. Note that Java SDK also includes JRE. This is available from http://java.sun.com

b. Installation

1. Download the SimplePERMIS cookbook example.
2. Download the example XML Policy .

4. Play with SimplePERMIS

The example policy (file test1.xml) is:

- users within the 'O=PERMIS,C=GB' domain who are assigned to the role 'role0' can do 'Action0' and 'Action3' on any target from the 'O=PERMIS,C=GB' domain,
- users within the 'O=PERMIS,C=GB' domain who are assigned to the role 'role1' can do 'Action1' and 'Action3' on any target from the 'O=PERMIS,C=GB' domain.

In addition, there is no role hierarchy. Documents about the PERMIS XML policy can be found in [1].

To launch the SimplePERMIS example, double click on *simplePermis.bat*.

The program will ask you if you want to run the non-interactive example (typing "automatic"), which is a pre-defined request using the TEST1.XML policy file, or to run the interactive example, where you can specify the policy file and build your own requests.

In interactive mode, the program will ask you to input the policy filename???, the user's DN, the user's role and the target DN (target name you want to access) and the name of the action you want to perform. After you have input appropriate names from Table 1 below, PERMIS will tell you if the action is allowed or denied, or will report any errors encountered. Then, the program will ask you if you want to build another request. Using the entries in Table 1 below, you can test different combinations of users, targets they are trying to access and the actions they are attempting to perform according to the TEST1.XML policy file.

User	Role	Target	Action	Result
Any DN from the o=PERMIS,c=GB domain	Role0	Any DN from the o=PERMIS,c=GB domain	Action0 or Action3	Action succeeded
Any DN from the o=PERMIS,c=GB domain	Role1	Any DN from the o=PERMIS,c=GB domain	Action1 or Action3	Action succeeded
Others combinations of User*Role*Target*Action				the action is not allowed

Table 1. Policy for test program

For example, if you input the first entry from the policy (in table 1):

uerDN: cn=user0,o=permis,c=gb
roleType: permisRole
roleValue: Role0
action : Action0
target DN: cn=object0, o=permis,c=gb

You should see the result '0: Action succeeded', as shown in the screenshot below (Figure 2).

Figure 2. Screenshot of simple PERMIS example

However if you input:

Screen Promote	**Input**
userDN	cn=User0,o=permis,c=gb
roleType	PermisRole
roleValue	Role1
action	Action0

target DN	o=permis,c=gb

Notice that after each input, you will need to press the “Enter” key on your keyboard.

You should now see the result ‘1: the action is not allowed’.

You can quit the SimplePERMIS example program by using the command ***exit.***

6. How to use SimplePERMIS in applications of your own?

SimplePERMIS works in push mode only, i.e. when a user requests to do an action on an object, the PEP give the credentials to the PDP using the method *PermisRBAC.getCreds*. This method assigns the standardised credentials to a principal, which is a *Subject* entity in PERMIS.

The user’s identifier and his role/attribute are given in your specific application format. The first step is to recognize the user’s identifier and his different roles/attributes. The second step is to convert the user’s identifier into a distinguished name. The class *SimplePERMISPrincipal* is a simple example. You can use it to create a java object for your own application.

The third step is to convert the different roles/attributes into the PERMIS authorisation token format (see the class *DefaultParsedToken*) in order to be understandable by the SimplePERMIS decision engine. Because the PERMIS decision engine is an RBAC-Based Authorisation infrastructure, the PERMIS decision engine uses roles/attributes to make access control decisions. Roles can be hierarchically organised in the PERMIS policy. In this context, a role is assigned to the permissions that are explicitly specified in the policy and also to the permissions assigned the sub-roles in the roles hierarchy. Then, the sub-roles should also be given to the PERMIS decision engine for its decision making process. The class *SimplePERMISTokenParser* shows how credentials can be standardised into PERMIS authorisation tokens.

You can use this class. However, if you want to modify it you should respect the following rules:

1. Use the same name or modify in the method *_init_* the class *CustomisePERMIS* the java code line “setAuthTokenParser("issrg.simplePERMIS.SimplePERMISTokenParser");” by “setAuthTokenParser("**your_class_name**");”. The class

CustomisePERMIS can be considered like a configuration file where the name of some classes used by PERMIS are defined. The AuthTokenParser class is dynamically loaded by PERMIS using the method getAuthTokenParser().

2. Your AuthTokenParser class should have the same methods as *SimplePERMISTokenParser* with the same signatures (i.e. number of parameters, type of the parameters, and order of the parameters …).

Finally, you have to write a PEP. The PEP captures the user's request and formats the request into calls to the PERMIS PDP for decisions. The classes *SimpleAEF_A* and *simpleAEF_I* are two examples of PEPs.

All SimplePERMIS Java classes can be downloaded in http://sec.cs.kent.ac.uk/permis/

References

[1] D.W.Chadwick, A. Otenko. "RBAC Policies in XML for X.509 Based Privilege Management" in Security in the Information Society: Visions and Perspectives: IFIP TC11 17th Int. Conf. On Information Security (SEC2002), May 7-9, 2002, Cairo, Egypt. Ed. by M. A. Ghonaimy, M. T. El-Hadidi, H.K.Aslan, Kluwer Academic Publishers, pp 39-53.