SMARTIE – Secure and Smarter Cities Data Management

BASIC CONCEPT / PLAN

Vedran Semenski

I. INTRODUCTION

The main goal is to build an access control framework using ABAC and the OASIS XACML/JSON standard. Along with this one NoSQL storage system will be setup for storing sensor data and another one for storing policies. This will be built in the context of the SMARTIE project but as a separate and independent component that will allow the enforcement of access control in any place needed inside the SmartData platform (that is being developed for the SMARTIE project). The best places to enforce access control at the moment would be when requesting access to sensor data directly.

II. BASIC CONCEPT

The NoSQL databases that will be used are Cassandra for storing sensor data and Redis for storing policies. Cassandra was chosen because of it's scaling characteristics and Redis for it's speed. A DataManager module will be built for each of these allowing easier use of these for other parts of the solution. The access control framework will have the OASIS architecture containing the: PEP, PDP, PRP, PIP and PAP but will be implemented with a subset of simpler functionalities. The limitations that will be taken into account come directly from the current state of the SMARTIE project which means that all or most use-case scenarios will be fulfilled but the overall solution can be simplified making the delivery more realistic (taking in account the time constraints).

III. SOME DETAILS REGARDING THE SOLUTION

The NoSQL database used for storing sensor data in the first stages will be setup for simple and generic sensor readings. The database will contain the following data: sensorID, sensorType, timestamp, valueOfReading. The structure will most likely mirror that list of values. In later stages the database will either be expanded to support the real sensor data currently used in the SMARTIE project or the policy list will be expanded and a connection to the current database used in SMARTIE (which is also a Cassandra DB).

The policies that are going to be generated and used are going to be simple and access to data will be denied by default. Options for policies in the first stages will include allowing access according to sensorID, sensorType, userGroup, userType (role). In later stages they will support basic environment conditions like day of the week and time of the day and other resource and user attributes (not yet decided which ones they will be).

The main focus most of the time spent in the development will be for the access control part. For the PDP which is one of the more complicated parts, an open source solution (Balana) will be used as a starting point. It supports XACML and should simplify the implementation of a PDP significantly. For the XACML/JSON policy building, a framework (https://github.com/att/XACML) with a REST service will be utilized to help with the integration. The PRP and PIP will connect to the policy database and will retrieve the attributes of the users, resource and environment. Implementation of these should be simple and straightforward. In the first stages User and Resource attributes will be stored in a separate database (not decided which one yet) and a number of test users and test resources and data will be generated for testing and development purposes. Later stages will include integration with SmartData regarding fetching user attributes and generating and storing resource attributes. The PAP will be the last component made in the Access Control framework and will support the creation of simple policies described near the beginning of this section.

This solution doesn't rely on other parts of the SMARTIE project and SmartData platform. This is good because independence will allow for good testing and easies development. However it is going to be built knowing that it has to be integrated or that it should be able to be integrated without much effort and/or modification.

IV. LIMITATIONS TAKEN IN CONSIDERATION

In its current stage, SmartData doesn't need support for complicated policies. Policies should mainly give access according to the users id, group and role and the resources id and type. There are no conditions regarding environment conditions, range of sensor values or other more complex conditions. This greatly simplifies the PAP and PDP needed in the solution.

V. POTENTIAL IMPROVEMENTS/ACHIEVEMENTS

In the current state, SmartData has access control implemented by encrypting data and allowing decryption to users that have access to it. The policies that contain the information on which users can access the data are stored together with the data itself ("sticky policies"). This means

that the all data needs to be fetched in order to evaluate the policy on it. This has a drawback in case of requests that result in denying access. If a user or set of users send a large number of requests for a lot of data, for which they don't have access to, an opening for a DoS type of attack could happen. This also means that in case of changing access policies for past data means fetching all the data that we want the change to affect and updating of the policy attached to it. This solution therefore isn't flexible if policies that are meant to be updated are to be integrated.

This Access Control framework should solve this issues offering a safer and more flexible solution and could potentially be integrated as access control "before" (regarding the point in communication) the current access control point.

VI. CONCLUSION

Using the OASIS ABAC implementation proposal architecture will allow the solution to be easily be built upon and allow the adding of other essential functionalities. This solution will therefore retain the structure of the OASIS ABAC implementation but will not have the full set of features/functionalities implemented. Just a subset determined to be enough taking in account the limitations set. The integration in SMARTIE or SmartData is not yet certain but will helpfully be done and prove to have significant benefits.