The State of the Access Control 2009

Principles, Requirements, Standards, Implementations & Gaps

Hal Lockhart, Oracle

Introduction

This paper briefly surveys the access control landscape from an XACML-centric point of view. It begins with some general principles which underlie successful access control systems and then discusses what access control components are available, where there are functional gaps and makes recommendations.

Access Control Principles

A number of principles of large-scale access control systems have emerged in recent years. Some of these are widely known; others are less so. All of them condition the possible solutions to access control requirements. Although not all of them were explicitly articulated at the time of its development, XACML is consistent with all of them.

Federation Principle

The Federation Principle is that data used as input to access control decisions (and for other security-related purposes) should be maintained and obtained directly from the party in the best position to be the authoritative source of that data. The rationale for this is that moving data to a central collection point increases the chance of incorrect or out of date information as well as requiring a extra entity to assume responsibility for protecting it. It also avoids the potential threat to privacy of giving a party unnecessary access to it. Federation is a relatively new concept because the necessary protocols to provide data on demand were only developed in the last decade.

Encapsulated Policy Engine

Modern access control systems encapsulate all policy evaluation into a Policy Decision Point (PDP), which is isolated from the Policy Enforcement Point (PEP) by a well-defined interface. This has a number of advantages. It greatly simplifies changing polices across many systems or identifying a single change within many systems and applications. It makes the policy much easier to analyze. It avoids the risks of incorrectly implemented policy logic scattered throughout application code.

In order for this principle to be effective, it must be followed throughout the access control model. Many current access control models do not follow this principle. For example, in the Java authorization model, decisions depend on both the output of the policy object and logic hidden within the various permissions classes. Dynamic Roles provide another negative example. There are polices about what Roles can be assumed and policies about what can be done when a give Role has been assumed.

Use Existing Data

Traditionally, access control systems have defined user attributes, UNIX groups for example. Experience has shown that this does not work well in large-scale environments. Often security administrators are the last to hear about changes within the organization, unless it is quite small. Similarly for Resource attributes, the organizations responsible for documents or servers are more likely to know what their properties are than people exclusively responsible for security. XACML is consistent with this principle in as much as it provides many datatypes for attributes, the ability to define attribute schemas, and many extraction, conversion and data manipulation functions in addition to the typical Boolean operators.

Limit Complexity

Large-scale access control polices are very complex, regardless of how they are expressed. While its power and flexibility are major advantages of XACML, human comprehension is the ultimate limit on a maintainable system. XACML provides some useful capabilities in this area, such as policy references to reuse some constructs and combining algorithms to allow partial policies to be defined independently of each other. However, better tools are required for analysis of policies. This is discussed further below. Ultimately, it may be necessary for policy architects to neglect some details or push certain logic outside of the policy engine entirely.

General Requirements

A complete access control system requires a number of components to provide a complete solution. In many cases the individual components will be designed to provide redundancy of one sort or another. An access control system does not exist in a vacuum. It must be integrated with other infrastructure and application components at various points.

In addition to at least one PDP and PEPs located at points where enforcement is possible, a complete system will also contain some or all of the following elements: interfaces to allow a PEPs to invoke policy decisions from local or remote PDPs, tools for creating and modifying policies, libraries of polices and policy templates, repositories to store polices, tools for distributing policies, tools for analyzing and debugging policies, logs and tools for auditing and analyzing past policy decisions, integration points to repositories and other sources of attribute data, configuration and management tools. Some of these components are well specified by standards and multiple open source and proprietary implementations exist. Other components are unspecified and non-existent. In other cases implementations exist, but their current design is unsuitable for large-scale environments.

Standards

XACML 2.0 is an OASIS Standard and an ITU/T Recommendation. It was completed in 2005 and has been implemented in scores of products and dozens of open source projects. XACML 3.0 is under development in OASIS and the core specification and an initial set of Profiles are nearing completion. The XACML 3.0 Profiles mostly are version independent or define specific behaviors for different XACML versions. As a consequence and because of the relative complexity of the XACML 3.0 reduction

algorithm, it can be expected that most deployments will use a mixture of XACML 2.0 and 3.0 for the next few years.

The XACML specifications mostly concern PDP behavior. Only a network interface between PEP and PDP was defined in XACML 2.0. Some aspects of a complete access control system are assumed to be covered by other standards. For example, attributes can be obtained by using SAML, LDAP, SQL or X.509 PKI. Other aspects have deliberately not been standardized to provide opportunities for innovation and product differentiation. Lastly, some technology requirements have only come to be understood as field experience develops and as IT environments mature generally.

Gaps

This section briefly describes various aspects of an access control system, attempts to characterize their current state in terms of standards and implementations and offers opinions about what is needed.

Policy Enforcement Points

Some categories of access control enforcement, such as protecting mobile data seem beyond the state of the art or at least require assumptions about the environment that are currently unreasonable for most organizations. However, currently even implementing a PEP in the most common server environment is more difficult than it needs to be. We at Oracle have made some efforts to address this problem and are eager to have others join us in this work.

Recently we have proposed a PEP/PDP API in Java. This is intended to permit both high performance access to a local PDP and easy integration with a remote PDP. We hope to define similar interfaces in other popular languages, including scripting languages. Our approach has been to develop a draft interface that we have contributed to the OASIS XACML TC and simultaneously to start an open source project called OpenAz to validate the design, experiment with solution alternatives and provide useful components that can be publicly downloaded.

The API, like the XACML abstract interface that it is based on, is not really intended for use by applications directly. Our hope is that it will be used by infrastructure components, such as containers and code generators, thus making access control an administrative activity, rather than a programming task. This project is described in more detail by a separate paper from Oracle.

Attributes for Policy Decisions

Probably the biggest practical barrier to the effective use of XACML is a variety of issues relating to the attribute information required to make policy decisions. First, there has been little standardization or even publication of their formats and semantics. Recently this has begun to change. The Open Geospatial Consortium (OGC) has developed specifications relating to access control of geospatial data. Healthcare standards developed originally in ASTM, Healthcare Information Technology Standards Panel (HITSP) and Health Level 7 (HL7) have been profiled for the implementation of access

control in healthcare. Work is occurring in the OASIS XACML TC to specify attributes for US Export Control and for Intellectual Property.

Much more is required. Without specifications like these, every vendor or end user organization must define attributes as a prerequisite to creating a working system. Further, policy portability and interoperability is reduced. This seems an area where further standardization is needed. Perhaps semantic web technology could play a role in better integrating authorization attributes with other enterprise data.

A second problem is that of distribution of attribute data. A number of low level standards have been developed for the distribution of Subject attributes. These include SAML, WS-Federation, WS-Trust and Identity Metadata Interoperability, as well as more generic repository access methods such as SQL and LDAP. However higher level issues such as privacy and security properties are not addressed by these alternatives. Another project started by Oracle, called the Identity Governance Framework (IGF) is intended to meet these requirements.

In contrast, in the area of Resource attributes, almost no standards or even best practices exist. In some cases, such as the security classification of documents, attributes may be imbedded in the documents as text, but not in any standard way. Currently there is an effort to define a placeholder in the Open Document Format to carry access control-related metadata. Formats and semantics for Resource attributes are also generally undefined.

A third problem is that of simply configuring systems to provide the data necessary to make policy decisions. Because of the encapsulation of the PDP, other components have no direct way of discovering what attributes must be provided for a decision. Recently Oracle has proposed a specification called the Attribute Manifest File (AMF). This is a simple XML format for exchanging information about attributes may be required for access control decisions. It contains naming and type information as well as retrieval information. This also has been contributed to the OASIS XACML TC and is a part of the OpenAz open source project.

Tools

Currently there are few tools that support the use of XACML. Two general types of tool are obviously needed: policy authoring tools and policy analysis tools. Very likely other types of tools will be identified in the future.

There are at least two approaches to authoring tools. One is an XACML oriented tool something like an IDE that simplifies and streamlines the creation of policies and helps prevent simple typing errors. Such a tool could consume AMF naming and type information. A tool like this should have analysis tools integrated into it for debugging. ExpressRules from Soph-Ware Associates is an example of this general approach, although it does not have all the features mentioned.

Another alternative is to model Enterprise policies at a higher level and then generate policies that can be directly evaluated, such as XACML policies. Object Security is a proponent of this approach.

In addition to specific tools for policy authoring there is a need to develop policy idioms and libraries as well as training and best practices. One major potential advantage of a standard policy language is having a pool of people skilled in its use, such as exists for SQL or Java, but this currently not the case for XACML.

Tools for analysis of XACML are not simple to implement, but will have great value. There are two aspects to this problem. The first is that the structure of the language makes it more difficult than traditional, reversible access control schemes and simple, less practical policy languages. I believe that this can be overcome by ingenuity. The second is that many existing tools are designed to operate in limited environments. For example, a list of all the users who can access a certain resource is not useful if it contains one million names.

The usual request for analysis is to report what users are allowed to do a certain thing under some set of circumstances. However this kind of report is infeasible in a large-scale environment for several different reasons. Many query mechanisms cannot be reversed. For example, SAML will provide the attributes of a user, but there is no easy way to ask for all the users with a given attribute value. Even where it is possible, not all the future sources of attributes may be known. Finally, a query may take so long that the data may have changed while it is in process.

I believe there are three general approaches that may be useful. One is to partially evaluate policies to a form where they are relatively easy to comprehend. (Partial evaluation would also be useful for improving PDP performance.). A second is to define sets of inputs and run a whole series of decisions (what if) over all the combinations. A third would be to reduce XACML to a logical language by factoring out constructs used to extract and format attribute values. Possibly some commenting conventions for annotating policies would be useful in producing more meaningful analyses.

In general I do not see much need for new standards in the area of tools.