ITL BULLETIN FOR NOVEMBER 2016

EXPLORING THE NEXT GENERATION OF ACCESS CONTROL METHODOLOGIES

David Ferraiolo, Larry Feldman,[1] and Greg Witte,[1] Editors
Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology
U.S. Department of Commerce

Introduction

As more and more information becomes centralized, controlling and managing access to sensitive data becomes increasingly challenging. Attribute-based access control (ABAC), which represents the latest milestone in the evolution of logical access control methods, is designed to address these challenges. It provides an attribute-based approach to accommodate a wide breadth of access control policies and simplify access control management.

Many other access control approaches are based on the identity of a user that is requesting a particular operation on a resource – in these cases, the decision is rendered based on the user's identity, or indirectly through predefined attribute types (e.g., roles, groups) assigned to the user. Practitioners have noted that these forms of access control are often cumbersome to set up and manage, given the challenges of associating permissions (also referred to as capabilities) directly to users or their attributes. Furthermore, the identity, group, and role qualifiers of a requesting user are often insufficient for expressing real-world access control policies. An alternative approach is to express policies in terms of attributes and to grant or deny user requests based on arbitrary attributes of users and resources, and, optionally, environmental attributes. This approach to access control is commonly referred to as attribute-based access control and is an inherent feature of both Extensible Access Control Markup Language (XACML) and Next Generation Access Control (NGAC).

ITL has recently published NIST Special Publication (SP) 800-178, *Comparison of Attribute-Based Access Control (ABAC) Standards for Data Service Application*. The purpose of the document is to compare and contrast XACML and NGAC — two different access control standards with similar goals and objectives.

SP 800-178 explains the basics of each standard and provides a comparative analysis based on five criteria. The first criterion is the relative degree to which the access control functionality of an application or other data service can be separated from a proprietary operational environment such as that of an operating system. The other four criteria are derived from ABAC considerations identified by NIST SP 800-162, *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*; the

[1] Larry Feldman and Greg Witte are Guest Researchers from G2, Inc.

criteria are operational efficiency, attribute and policy management, scope and type of policy support, and support for administrative review and resource discovery.

Separation of Access Control Functionality from Proprietary Operating Environments

Both XACML and NGAC achieve separation of access control functionality of data services from proprietary operating environments, but to different degrees. XACML's separation is partial. XACML does not envisage the design of a Policy Enforcement Point (PEP) that is data service-agnostic. An XACML deployment consists of one or more data services, each with an operating environment-dependent PEP and operating environment-dependent operational routines and resource types that share a common Policy Decision Point (PDP) and access control information consisting of policies and attributes.

The degree of separation that can be achieved by NGAC is near complete. Although an NGAC deployment could include a PEP with an application programming interface (API) that recognizes operating environment-specific operations (e.g., send and forward operations for a messaging system), it does not necessarily need to do so. NGAC deployment can include a standard PEP with an API that supports a set of generic, operating environment-agnostic operations (read, write, create, and delete policy elements and relations). This API enables a common, centralized PEP to be implemented to serve the requests of multiple applications.

Operational Efficiency

An XACML request is a collection of attribute name, value pairs for the subject (user), action (operation), resource, and environment. XACML identifies relevant policies and rules for computing decisions through a search for Targets (conditions that match the attributes of the request). Because multiple policies in a policy set and/or multiple rules in a policy may produce conflicting access control decisions, XACML resolves these differences by applying a policy combining algorithm from a set defined by the standard. The entire process includes collecting attributes, matching conditions, computing rules, and resolving conflicts involving at least two data stores. There are two phases of policy evaluation that need to be considered. The first and costliest is loading policy from disk to Policy Decision Point (PDP) main memory, and the second is request evaluation. In both phases, performance is directly related to the number of policies considered.

An NGAC request is composed of a process ID, user ID, operation, and a sequence of one or more operands mandated by the operation that affects either a resource or access control data. NGAC identifies relevant Policies and attributes by reference when computing a decision. NGAC computes decisions by applying a single combining algorithm over applicable Policies that do not conflict. Unlike XACML, NGAC does not need to load policy from disk into memory when evaluating a request. Instead, and as treated in NGAC reference implementation version 1.6, all information necessary in computing

an access decision can reside in memory. Memory is initially loaded when the PDP is initialized, and is updated when an administrative change occurs. The NGAC specification describes what constitutes a valid implementation, but does not provide implementation guidance, thereby leaving room for multiple competing approaches with different efficiencies. A measure of the operational efficiency is the complexity of algorithm used for arriving at a policy decision. In its reference implementation version 1. 6 on GitHub, the NGAC computes a decision through an algorithm that is linear. Furthermore, it is not linear in relation to the entire access control data set, but only to the portion relevant to a particular user.

Attribute and Policy Management

Proper enforcement of data resource policies is dependent on administrative policies. This is especially true in a federated or collaborative environment, where governance policies require different organizational entities to have different responsibilities for administering different aspects of policies and their dependent attributes.

XACML and NGAC differ dramatically in their ability to impose policy over the creation and modification of access control data (attributes and policies). NGAC manages attributes and policies through a standard set of administrative operations, applying the same enforcement interface and decision-making function as it uses for accessing data resources. XACML does not recognize administrative operations, but instead manages policy content through a Policy Administration Point (PAP) with an interface that is different from that for accessing data resources. XACML provides support for decentralized administration of some of its access policies. However, the approach is only a partial solution in that it is dependent on trusted and untrusted policies, where trusted policies are assumed valid, and their origin is established outside the delegation model. Furthermore, the XACML delegation model does not provide a means for imposing policy over modification of access policies, and offers no direct administrative method for imposing policy over the management of its attributes.

NGAC enables a systematic and policy-preserving approach to the creation of administrative roles and delegation of administrative capabilities, beginning with a single administrator and an empty set of access control data, and ending with users with data service, policy, and attribute management capabilities. NGAC provides users with administrative capabilities down to the granularity of a single configuration element, and it can deny users administrative capabilities down to the same granularity.

Scope and Type of Policy Support

Although resources may be protected under a wide variety of different access policies, these policies can generally be categorized as either discretionary or mandatory controls. Discretionary access control (DAC) is an administrative policy that permits system users to allow or disallow other users' access to resources that are placed under their control. Although XACML can theoretically provide users with

administrative capabilities necessary to control and give away access rights to other users, the approach is complicated by the need to create and maintain additional metadata for each and every object/resource (e.g., Owner attribute). Conversely, NGAC has a flexible means of providing users with administrative capabilities to include those necessary for the establishment of DAC policies.

In contrast to DAC, mandatory access control (MAC) enables ordinary users' capabilities to execute operations on resources, but not administrative operations that may influence those capabilities. MAC policies unavoidably impose rules on users in performing operations on resources. MAC policies can be further characterized as controls that accommodate confinement properties to prevent indirect leakage of data to unauthorized users, and those that do not.

Expression of non-confinement MAC policies is perhaps XACML's strongest suit. XACML can specify rules and other conditions in terms of attribute values of varying types. There are undoubtedly certain policies that are expressible in terms of these rules that cannot be easily accommodated by NGAC. This is especially true when treating attribute values as integers. For example, when considering a purchase request, the system may add the two attribute values of a person's credit limit and account balance to determine if adequate funds are available. Furthermore, XACML takes environmental attributes into consideration in expressing policy, and NGAC does not. However, there are some non-confinement MAC properties, including a variety of history-based policies, that NGAC can express but XACML cannot.

In contrast to NGAC, XACML does not recognize the capabilities of a process independent of the capabilities of its user. Without such features, XACML cannot support confinement or enforce a variety of confinement policies. These confinement-dependent policies include some instances of role-based access control (RBAC), e.g., "only doctors can read the contents of medical records," originator control (ORCON) and privacy, e.g., "I know who can currently read my data or personal information," or conflict of interest, e.g., "a user with knowledge of information within one dataset cannot read information in another dataset." Through imposing process-level controls in conjunction with event-response relations, NGAC has shown support for these and other confinement-dependent MAC controls.

Administrative Review and Resource Discovery

A desired feature of access controls is the ability to review the capabilities of users and the access control entries of objects. These features are often referred to as "before the fact audit" and resource discovery. "Before the fact audit" is one of RBAC's most prominent features. Being able to discover or see a newly accessible resource is an important feature of any access control system. NGAC supports efficient algorithms for both per-object and per-user review. Per-object review of access control entries is not as efficient as a pure access control list (ACL) mechanism, and per-user review of capabilities is not as efficient as that of RBAC. However, this is due to NGAC's consideration of conducting review in a multi-policy environment. NGAC can efficiently support both per-object and per-user reviews of combined policies, where RBAC and ACL mechanisms can do only one type of review efficiently, and

logical formula-based mechanisms such as XACML, although able to combine policies, cannot do either type of review efficiently.

Conclusion

XACML is similar to NGAC in that they both employ attributes in computing decisions, and both provide flexible, mechanism-independent representations of policy rules that may vary in granularity. However, XACML and NGAC differ significantly in their expression and management of policies, treatment of attributes, computation of decisions, and representation of requests. Many of these differences stem from the methods with which they represent policies and attributes. XACML's approach is to define policies by using logical formulas involving attribute values, while NGAC uses enumeration involving configurations of relations. As a consequence of this and other factors, XACML and NGAC have comparative advantages and disadvantages.

Although the criteria used in this document to compare XACML and NGAC are significant factors for users to consider in choosing future ABAC deployments and for vendors considering future product offering, this set is by no means exhaustive. From a user's perspective, NGAC is new with few products available for testing and evaluation. From a vendor's perspective, the NGAC specification describes only what constitutes a valid implementation using set theoretic notation, thereby leaving room for multiple competing implementations. XACML, on the other hand, has served as a basis for a number of proprietary and open-source product offerings that cover virtually all aspects of its deployment.

Additional Resource

NIST's NGAC reference implementation ("Harmonia"), versions 1.5 and 1.6 – https://github.com/PM-Master

ITL Bulletin Publisher: Elizabeth B. Lennon
Information Technology Laboratory
National Institute of Standards and Technology
elizabeth.lennon@nist.gov