# Implementing and Managing Policy Rules in Attribute Based Access Control

Vincent Hu[1], David F. Ferraiolo[1], D. Richard Kuhn[1], Raghu N. Kacker[1], Yu Lei[2]

[1] *National Institute of Standards and Technology*
*Gaithersburg, MD 20899, USA*
{ *vhu,david.ferraiolo,kuhn,raghu.kacker*}*@nist.gov*

[2]*Computer Science & Engineering*
*University of Texas at Arlington*
*Arlington, TX, USA*
*ylei@uta.edu*

*Abstract* – **Attribute Based Access Control (ABAC) is a popular approach to enterprise-wide access control that provides flexibility suitable for today's dynamic distributed systems. ABAC controls access to objects by evaluating policy rules against the attributes of entities (subject and object), operations, and the environment relevant to a request, but great care must be taken in setting up and maintaining the access control rules that allow such flexible operations.This article summarizes important considerations in ABAC deployment first introduced in the *Guide to Attribute Based Access Control* [1].**

***Keywords- access control; attribute based access control; XACML; NGAC.***

## I. INTRODUCTION

Access control mechanisms comprise several components that work together to bring about policy-preserving resource access. These components include access control data for expressing access control policies and a set of functions for issuing and trapping access requests, and computing and enforcing decisions over those requests in accordance with the policies. Most operating environments implement access control in different ways, each with a different scope of control (e.g., users, resources), and each with respect to different operation types (e.g., read, send, approve, select) and resource types (e.g., files, messages, work items, records).

This heterogeneity introduces a number of administrative and policy enforcement challenges. Administrators must contend with a multitude of security domains when managing access policies. Even if properly coordinated across operating environments, global controls are hard to visualize and implement in a piecemeal fashion. Furthermore, because operating environments implement access control in different ways, it is difficult to exchange and share information across operating environments. ABAC seeks to alleviate these challenges by creating a common and centralized way of expressing various policies, and computing and enforcing decisions, over the access requests of data services.

Most other access control approaches are based on the identity of a user requesting execution of a capability to perform an operation on an object (e.g., read a file), either directly via the user's identity, or indirectly through predefined attribute types such as roles or groups assigned to that user. Practitioners have noted that these forms of access control are often cumbersome to manage, given the need to associate capabilities directly to users or their attributes. Furthermore, the identity, group, and role qualifiers of a requesting user are often insufficient in the expression of real-world access control policies. An alternative is to grant or deny user requests based on arbitrary attributes of users and objects, and optionally environmental conditions that may be globally recognized and tailored to the policies at hand.

This approach to access control is commonly referred to as *attribute-based access control* (ABAC) [1][2], and is an inherent feature of both the eXtensible Access Control Markup Language (XACML) [3] and Next Generation Access Control (NGAC) [4][5] standards.

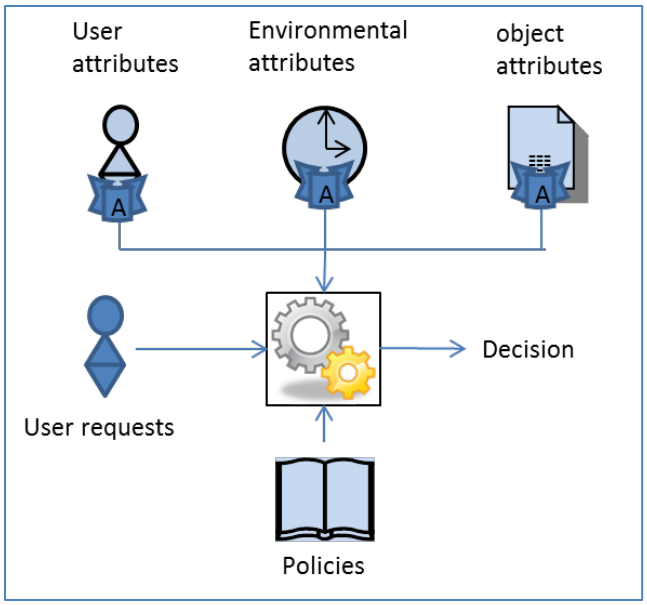


**Figure 1.** ABAC Overview

While largely developed in parallel, these standards were established under different timetables and circumstances. XACML was developed as collaboration among vendors with a goal to separate policy expression and decision-

making from proprietary operating environments in support of the access control policy needs of data services. XACML first appeared in 2003, and was recently revised in 2013 by providing support for decentralized policy management. NGAC's origin stems from the NIST Policy Machine [6][7][8], a research effort that began in 2003 to develop a general-purpose attribute-based access control framework.

From a management perspective, ABAC has advantages over other access control approaches. ABAC avoids the need for capabilities (operation, object pairs) to be directly assigned to requesting users or to their roles or groups before the request is made. Instead, when a user requests access, the ABAC engine (see Figure 1) can make an access control decision based on the assigned attributes of the requesting user and the object, environmental attributes, and a set of policies that are specified in terms of those attributes. Under this approach, policies are managed without direct reference to potentially numerous users and objects, and users and objects can be provisioned without reference to policy details.

In its most basic form, ABAC relies upon the evaluation of attributes of the *subject*, attributes of the *object*, *environment conditions*, and the formal relationship or access control rule or policy defining the allowable operations for subject-object attribute combinations. All ABAC solutions contain these basic core capabilities to evaluate attributes and enforce rules or relationships between those attributes. While XACML explicitly defines rules as a basic policy building block, the NGAC approach uses relations that imply rules in formulating policy. For readability, this paper generically uses the term rules to refer to either approach.

Even within a small isolated system, ABAC relies upon the assignment of attributes to subjects and objects, and the development of policy implemented in terms of attributes. Each object within the system must be assigned specific object attributes that characterize the object. Some attributes pertain to the entire instance of an object, such as the owner. Other attributes may only apply to parts of the object. For example, a document object could be owned by *Organization A*, have a section with intellectual property from *Organization B*, and be part of a program run by *Organization C*. As another example, consider a document residing in a directory within a file management system. This document has a title, an author, a date of creation, and a date of last edit—all object attributes that are determined by the creator, author, or editor of the document. Additional object attributes may be assigned such as owning organization, intellectual property characteristics, export control classification, or security classification. Each time a new document is created or modified, these object attributes must be captured. These object attributes are often embedded within the document itself, but they may be captured in a separate table, incorporated by reference, or managed by a separate application.

Each subject that uses the system must be assigned specific attributes. Consider the example of a user accessing a file management system. The user is established as a subject within the system by an administrator and characteristics about that user are captured as subject attributes. This subject may have a name, a role, and an organization affiliation. Other subject attributes may include US citizenship status, nationality, and security clearance. These subject attributes are assigned and managed by an authority within the organization that maintains the subject identity information for the file management system. As new users arrive, old users leave, and characteristics of subjects change, these subject attributes may need to be updated.

Every object within the system must be included in at least one policy that defines the access rules for the allowable subjects, operations, and environment conditions to the object. Policies are normally derived from documented or procedural rules that describe the business processes and allowable actions within the organization. For example, in a hospital setting, a rule may state that only authorized medical personnel shall be able to access a patient's medical record. In a system with this policy, a *MedicalRecordRule* may ensure that a document object with a *RecordType* attribute of *PatientMedicalRecord* will cause a subject with a *PersonnelType* attribute value of *NonMedicalSupportStaff* trying to perform the Read operation to be denied access and the operation disallowed. This is only one approach to implementing the connection between attributes and rules, and organizations may implement similar policies in different ways.

Rules binding subject attributes, object attributes, and environment conditions indirectly specify privileges (i.e., which subjects can perform which operations on which objects in what environment condition). Allowable operation rules can be expressed through many forms of computational language such as:

- A Boolean combination of attributes and conditions that satisfy the authorization for a specific operation (XACML)
- A set of relations associating subject attributes, object attributes, and environment conditions and allowable operations (NGAC)

Once object attributes, subject attributes, and policies are established, objects can be protected using ABAC. Access control mechanisms mediate access to the objects by limiting access to allowable operations by allowable subjects. The access control mechanism (ACM) assembles the policy, subject attributes, object attributes, and environment conditions, then renders and enforces a decision based on

the logic provided in the policy. Access control mechanisms must be able to manage the process required to make and enforce the decision, including determining what policies and attributes are relevant, and where to retrieve attributes. The ACM must then perform the computation necessary to render a decision.

The policies that can be implemented in an ABAC model are limited only to the degree imposed by the computational language or set of relations, and the richness of the available attributes. This flexibility enables the greatest breadth of subjects to access the greatest breadth of objects without having to specify individual relationships between each subject and each object. For example, a subject is assigned a set of subject attributes upon employment (e.g., Nancy Smith is a *Nurse Practitioner* in the *Cardiology Department*). An object is assigned its object attributes upon creation (e.g., a folder with *Medical Records* of *Heart Patients)*. A designated authority creates rules to govern the set of allowable operations (e.g., all *Nurse Practitioners* in the *Cardiology Department* can *View* the *Medical Records* of *Heart Patients).* Adding to the flexibility, attributes and their values may then be modified throughout the lifecycle of subjects, objects, and attributes.

Provisioning attributes to subjects and objects governed by a policy that specifies what operations can take place enables an unlimited number of subjects to perform operations on the object—all without prior knowledge of the specific subject by the object-owner or rule-maker. As new subjects join the organization, rules and objects do not need to be modified. As long as the subject is assigned the attributes necessary for access to the required objects (e.g., all *Nurse Practioners* in the *Cardiology Department* are assigned those attributes), no modifications to existing rules or object attributes are required. This benefit is often referred to as accommodating the external (unexpected) user and is one of the primary benefits of employing ABAC.

Contrary to XACML's ABAC approach, under the definition of ABAC presented here, operations do not have "attributes". As defined attributes contain information given by a name-value pair. For example, "read = all" (or "all = read") is not appropriate. Operations can have many types or classes, which are not "attributes" but a fixed set of values. It would be possible to make operation itself an "attribute name", such as "operation = read", but this would then be the only attribute for operation, which would be redundant.

To meet accountability requirements, there will be a need to track accesses of objects to specific subjects linked to specific users. Accountability could be lost if access decisions are based on attributes, if subject or user IDs are not tracked to specific access requests and decisions.

While ABAC is an enabler of information sharing, when deployed across an enterprise, the set of components required to implement ABAC gets more complex. At the enterprise level the increased scale requires complex and sometimes independently established management capabilities necessary to ensure consistent sharing and use of policies and attributes and the controlled distribution and employment of access control mechanisms throughout the enterprise.

Some enterprises have existing capabilities that can be leveraged to implement ABAC. For example, most enterprises have some form of identity and credential management to manage population of subject attributes, such as name, unique identifier, role, clearance, etc. Similarly, many enterprises may have some organizational policy or guidelines to establish rules authorizing subjects' access to enterprise objects. However, these rules are usually not written in a machine-enforceable format that can be integrated consistently across all applications. ABAC policies must be made available in machine-enforceable format, and stored in repositories and published for access control mechanism consumption. These digital policies include subject and object attributes, with environment conditions, required to render access control decisions. The enterprise subject attributes must be created, stored, and shared across organizations within the enterprise through a subject attribute management capability. Likewise, enterprise object attributes must be established and bound to objects through an object attribute management capability. At this point, the ABAC-enabled access control mechanisms must be deployed. The remainder of this paper provides more detail on each of these major components of enterprise ABAC.

## II. Mapping from Natural Language Policy to ABAC Rules

Natural Language Policies (NLPs) are high-level requirements that specify how information access is managed and who, under what circumstances, may access what information. NLPs are expressed in human understandable terms and may not be directly implementable in an access control mechanism. NLPs may be ambiguous and thus hard to derive in formally actionable elements, so the enterprise policy may be difficult to encode in machine-enforceable form. While NLPs can be application-specific and thus taken into consideration by the application system, NLPs are just as likely to pertain to subject actions that span multiple applications. For instance, NLPs may pertain to object usage within or across organizational units or may be based on need-to-know, competence, authority, obligation, or conflict-of-interest factors. Such policies may span multiple computing platforms and applications.

Given that relevant NLPs exist for each organization in an enterprise, the next step is to translate those into a common set of rules that can be enforced equally and consistently within the ACMs across the enterprise. In order to accomplish this, it is necessary to identify all required subject/object attribute combinations and allowable operations. Often these values will vary from organization to organization and may require some form of consensus or mapping to each organization's existing attributes to accommodate enterprise interoperability. The agreed-upon list of subject and object attributes, the allowable operations, and all mappings from existing organization-specific attributes are then translated into machine-enforceable format. NLPs must be codified into Digital Policy (DP) algorithms or mechanisms. For efficiency of performance and simplicity in specification, an NLP may require decomposition and translation into different DPs that suit the infrastructure of operation units in the enterprise.

Multiple DPs may require Metapolicies (MPs), or policies dictating the use and management of DPs to handle DP hierarchical authorities, DP deconfliction, and DP storage and updates. MPs are used for managing DPs. Depending on the level of complexities, hierarchical MPs may be required based on the structures for the priority and combination strategies specified by NLP.

Once DPs and MPs are developed they need to be managed, stored, validated, updated, prioritized, deconflicted, shared, retired, and enforced. Each of these operations requires a set of capabilities that will often be distributed across the enterprise and is collectively termed Digital Policy Management (DPM). There may be multiple policy authorities and hierarchies within organizations that have variations on enterprise policy. The rules for how DPs and MPs are managed may be determined by a central authority.

Proper DP definition and development are critical to the identification of subject and object attributes that are needed to render an access control decision. Remember that a DP statement is comprised of the subject and object attribute pairings as well as environment conditions needed to satisfy a set of allowable operations. Once the full set of subject and object attributes needed to satisfy the entire set of allowable operations for a given set of enterprise objects is identified, this set of attributes comprises the entire set of attributes needed to be defined, assigned, shared, and evaluated for enterprise ABAC access decisions. For this reason, identifying the NLP and DP must be accomplished by the support of attributes when implementing an enterprise ABAC capability.

## III. ATTRIBUTE MANAGEMENT

Next, consider the lists of attributes developed while examining the NLPs and DPs. Without a sufficient set of object and subject attributes, ABAC does not work. Attributes need to be named, defined, given a set of allowable values, assigned a schema, and associated to subjects and objects. Subject attributes need to be established, issued, stored, and managed under a governance policy. Object attributes must be assigned to the objects. Attributes shared across organizations should be located, retrieved, published, validated, updated, modified, and revoked.

Subject attributes are provisioned by attribute authorities—typically authoritative for the type of attribute that is provided and managed through an attribute administration point. Often, there are multiple authorities, each with authority over different attributes. For example, Security might be the authority for Clearance attributes, while Human Resources might be the authority for Name attributes. Subject attributes that need to be shared to allow subjects from one organization to access objects in another organization must be consistent, comparable, or mapped to allow equivalent policies to be enforced. For example, a member of *Organization A* with the role *Job Lead* wants to access information in *Organization B*, except *Organization B* uses the term *Task Lead* to denote the equivalent role. This problem also applies to mapping between an enterprise attribute schema and an application-specific schema, particularly ones built before the enterprise schema is defined and/or COTS products that come with their own built-in schema. Organizations must normalize subject attribute names and values, or maintain a mapping of equivalent terms for all organizations. This should be managed by a central authority.

Object attributes need to be established, maintained, and assigned to objects as objects are created or modified. While it may not be necessary to have a common set of object attributes in use across the enterprise, object attributes should be consistently employed to fulfill enterprise policy requirements, and available sets of object attributes should be published for those wishing to mark, tag, or otherwise apply object attributes to their objects. At times, it might be necessary to ensure that object attributes are not tampered with or altered to satisfy an access request. Objects can be cryptographically bound to their object attributes to identify whether objects or their corresponding attributes have been inappropriately modified. Mechanisms must be deployed to ensure that all objects created are assigned the appropriate set of object attributes to satisfy the policy being employed by the ACM. It may be necessary to have an Enterprise Object Attribute Manager to coordinate these requirements.

In the course of managing attributes, the concept of "metaattributes"—or characteristics of attributes— arises. Metaattributes apply to subjects, objects, and environment conditions as extended attribute information useful for enforcing more detailed policy that incorporates information

about the attributes and for managing the volumes of data needed for enterprise attribute management. For example, metaattributes giving the source and origination date of attribute values may be used in ensuring that attribute values meet the organization's required level of assurance.

## IV. Access Control Mechanism Distribution

Finally, consider the distribution and management of ACMs throughout the enterprise. Depending on the needs of the users, size of the enterprise, distribution of the resources, and sensitivity of the objects that need to be accessed or shared, the distribution of ACMs can be critical to the success of an ABAC implementation. The functional components of an ACM may be physically and logically separated and distributed within an enterprise rather than centralized as described in the system-level view of ABAC.

Within the ACM are several functional "points" that are the service node for retrieval and management of the policy, along with some logical components for handling the context or workflow of policy and attribute retrieval and assessment. These include the Policy Enforcement Point (PEP), the Policy Decision Point (PDP), the Policy Information Point (PIP), and the Policy Administration Point (PAP). When these components are in an environment, they must function together to provide access control decisions and policy enforcement.

A PDP performs an evaluation on DPs and MPs in order to produce an access control decision. The PEP enforces decisions made by the PDP. PDP and PEP functionality can be distributed or centralized, and may be physically and logically separated from each other. For example, an enterprise could establish a centrally controlled enterprise decision service that evaluates attributes and policy, and renders decisions that are then passed to the PEP. This allows for central management and control of subject attributes and policy. Alternatively, local organizations within the enterprise may implement separate PDPs which draw on a centralized DP store. The design and distribution of ACM components requires a management function to ensure coordination of ABAC capabilities.

To compute access decisions, the PDP must have information about the attributes. This information is provided by the PIP. Before these policies can be enforced, they must be thoroughly tested and evaluated to ensure they meet the intended need.

Some systems may include an additional component within the ACM, a Context Handler that manages the order of policy and attribute retrieval. This can be important when time critical or disconnected access control decisions must be made. For example, attributes may be retrieved in advance of an access request, or cached to avoid the delay inherent in retrieval at the time of the access request. The Context Handler also coordinates with PIPs to add attribute values to the request context, and converts authorization decisions in the canonical form (e.g., XACML) [2] to the native response format.

## V. Enterprise Considerations

Access control policies are expressed in terms of attributes. Consequently all required attributes must be established, defined, and constrained by allowable values required by the appropriate policies. The schema for these attributes and allowable attribute values must be published to all participants to help enable object owners with rule and relationship development. Once attributes and allowable values are established, methods for provisioning attributes and appropriate attribute values to subjects and objects need to be established as well as an architecture for any attribute repositories, retrieval services, or integrity checking services. Interfaces and mechanisms must be developed or adopted to enable sharing of these attributes.

***Subject Attributes*** Many human subject attributes are typically provisioned upon employment with the organization and may be provisioned by several different authorities (human resources, security, organization leadership, etc.) For these, approaches to obtaining authoritative data are well known. As an example, only security authorities should be able to provision and assert clearance attributes and attribute values based on authoritative personnel clearance information; an individual should not be able to alter his or her own clearance attribute value. Other subject attributes may involve the subject's current tasking, physical location, and the device from which a request is sent; processes need to be developed to assess and assure the quality of such subject attribute data.

Authoritative subject attribute provisioning capabilities should be appropriately dependable in regards to quality, assurance, privacy, and service expectations. These expectations may be defined in an Attribute Practice Statement (APS). An APS provides a listing of the attributes that will be used throughout the enterprise, and may identify authoritative attribute sources for the enterprise. Still further network infrastructure capabilities (including the ability to maintain attribute confidentiality, integrity, and availability) are required to share and replicate authoritative subject attribute data within and across organizations.

***Object Attributes*** Object attributes are typically provisioned upon object creation and may be bound to the object or externally stored and referenced. It is to be expected that access control authorities cannot closely monitor all events. Frequently, this information is driven by non-security processes and requirements. Good attribute data that support

good access decisions are essential, and measures must be taken to ensure that object attributes are assigned and validated by processes that the object owner or administrator considers appropriate for the application and authoritative. For example, object attributes must not be modifiable by the subject to manipulate the outcome of the access control decision. The object attributes must be made available for retrieval by access control mechanisms for access control decisions. Additional considerations for creating object attributes include:

- In general, users will not know the attributes of an object (e.g., to which sensitive compartment a given user is authorized). This should be accounted for in ACMs, so that users only see the attributes that are applicable to them.
- As with subject attributes, a schema is required for object attributes defining attribute names and allowed values.
- Attributes need to be kept consistent in DP, MP, and NLP.

There have been numerous efforts within the Federal Government and commercial industry to create object attribute tagging tools that provide not only data tagging, but also cryptographic binding of the attributes to the object and validation of the object attribute fields to satisfy access control decision requirements.

***Environment Condition*** Environment condition refers to context information that generally is not associated with any specific subject or object but is required in the decision process. They are different from subject and object attributes in that they are not administratively created and managed, but instead are intrinsic and must be detectable by the ABAC system. Environment conditions such as the current date, time, location, threat, and system status, usually are evaluated against current matching environment variables when authorizing an access request. Environment conditions allow ABAC policies to specify exceptional or dynamic access control rules that cannot be described by subject/object attributes only. When composing ABAC rules with environment conditions, it is important to make sure that the environment condition variables and their values are globally accessible, tamper proof, and relevant for the environments where they are used.

***Access Control Rules*** In ABAC, all access control rules must include some combination of attributes and allowable operations. They may also include conditions, hierarchical inheritance, and complex logic. Together these provide a rich array of options when implementing ABAC. Rule sets and the application of rule sets to objects must be governed and managed appropriately. Rules must accurately and completely reflect the NLP, and be authoritatively developed (some by organizations, some by resource owners), applied, maintained, shared, and asserted. ABAC allows multiple rules from multiple stakeholders. New techniques are needed to coordinate and obtain the proper balance of sharing and protection. In some settings, one might limit the visibility of which rules apply to which objects to limit the likelihood of unauthorized subjects manipulating attributes to obtain authorization. In other circumstances, subjects that are denied access should have a method to verify or rectify the circumstances that caused the denial. Some organizations may wish to track the denials to see if the rules were appropriate. Similarly, rule definition and employment mechanisms and processes should include a robust rule deconfliction (resolution for the different decisions of rules) capability to determine rule conflicts and resolution processes.

***Access Control Mechanism and Context Handling*** The distribution and orchestration of ACM must be predetermined to avoid conflicts and weaknesses in object protection. For example, if an identical object is held by two different organizations, an unauthorized subject should not be able to access the version held by the organization with lesser restrictions. ACMs should be managed, maintained, and employed in a consistent manner to ensure interoperability and comprehensive security.

The order in which the ACM retrieves information, evaluates for a decision, and enforces the decision can differ greatly based on the specific requirements of the implementation, and may even take into account environment conditions during access control decision rendering. This is referred to as Context Handling and simply refers to the workflow the ACM undertakes when gathering the data needed for a decision.

Additionally, where and how policy, attribute, and decision information are stored and exchanged throughout the enterprise is an important consideration, for performance and scalability purposes.

## VI. Implementation and Assessment Phase

In the implementation and assessment phase, the organization installs or implements the system, configures and enables system security features, tests the functionality of these features, and finally, obtains a formal authorization to operate the system. Most of the considerations during this phase are focused on optimizing performance and ensuring security features work as expected.

***Attribute Caching*** When an ABAC solution moves from the prototype or pilot to deployment, attribute caching may be considered to enhance performance. Performance of the ABAC solution can be negatively affected if each access

decision requires an across-the-network attribute request. This is especially apparent in low-bandwidth, high-latency environments.

In addition to performance issues regarding attribute caching, the organization may evaluate a tradeoff regarding the freshness of attributes and the impact upon security. Attributes that are not refreshed as often will ultimately be less secure than attributes that are refreshed in real time. For example, a subject's access privileges may have changed since the last refresh, but those updates will not be reflected in their available access privileges until the next refresh.

Environments with sporadic connectivity will need to cache attributes at the local level. The security ramifications of using cached attributes locally need to be determined within the implementing organization at a policy level, and addressed with appropriate technical controls. In these disconnected environments, administrators may employ risk-based analysis as a basis for access decisions, as some attributes at the local (disconnected) level may change or be removed before the system refreshes its attributes. The local (and disconnected system's) possible use of stale cached attributes could introduce a level of risk to the system, because the local system is not making use of the most recently available attributes. Therefore, a risk-based analysis may be warranted as to whether or not to deploy this type of solution.

An example is a deployed ship with only intermittent, non-ideal connections to enterprise network fabrics. Because the deployed user population will have only minor changes throughout their transit, supporting the "unanticipated" system user is less of a concern. In this case, a bulk download and local storage of subject attributes may be sufficient for most local access control decisions. Therefore, subject attribute data could be stored locally on the ship throughout a deployment, and local applications and services could use the data from the local store without the need to reach to an authoritative enterprise attribute source. While this is one example of a solution to an austere environment problem, it should not be inferred that this is the only solution.

***Attribute Source Minimization*** Minimizing the number of attribute sources used in authorization decisions may improve performance and simplify the overall security management of the ABAC solution. Organizations planning to deploy an ABAC solution may benefit from establishing a close working relationship among all of the organization's stakeholders who will be involved in the solution's deployment.

***Interface Specifications*** To help ensure consistently reliable access to ABAC services, all organizations that participate in information sharing through enterprise ABAC capabilities should fully understand the interface, interaction, and precondition requirements for all types of requests, including attribute and DP requests. It is also important to ensure that as changes occur in the infrastructure and interface requirements, all relying parties are provided notification of updates so they can plan to modify their components accordingly.

## VII. Operations and Maintenance Phase

In the operations and maintenance phase, systems and products are in place and operating, enhancements and/or modifications to the system are developed and tested, and hardware and/or software is added or replaced. During this phase, the organization should monitor performance of the system to ensure that it is consistent with preestablished user and security requirements, and needed system modifications are incorporated.

***Availability of Quality Data*** As the information needed to render access control decisions, and in some cases the decisions themselves, is externalized from the objects and consumers, access to information and services will become more dependent on an outside service's ability to provide timely and accurate data. It is important that the infrastructure be robust, well-tested, resilient, and scalable to mission needs. This is important to support attribute services, attribute stores, policy stores, policy and attribute generation and validation components, decision engines, and metaattribute repositories and conduits through which this information must pass. If outsourced, service agreements should detail availability, response time, and data quality and integrity requirements. For example, failover, redundancy, and continuity of operations must be considered for data and services that are considered mission critical. Maintaining high availability of quality data requires that addition, updating, and deleting of attribute values is performed by trained, authorized individuals, and regularly audited.

Formal agreements between providers and consumers of attributes and services should meet an appropriate standard of service, quality, availability, protection, and usage. Various laws and regulations establish responsibilities, liabilities, and penalties related to the appropriate protection of information. The agreements should capture these requirements as well as those related to responsibility for data.

Agreements establishing an appropriate level of trust between organizations are important. These agreements would serve to formalize that trust relationship with a series of requirements and, possibly, penalties for nonconformance. Inter-organization agreements for attribute services and authoritative and accountable attribute sources can also serve to translate organizational policy into operational procedures. The purpose, usage, participants,

responsibilities, and administration of these services are described in these formal agreements.

## VIII. Summary and Conclusions

Attribute Based Access Control facilitates standards-based, policy-preserving user executions of data service capabilities (data service operations on data service resources). Data services can take on many forms, including applications such as time and attendance reporting, payroll processing, and health benefits management as well as system level utilities such as file management. In lieu of a standards based approach, control over access capabilities is achieved by an access control mechanism implemented in an underlying and often proprietary operating environment. The complexities of implementing any access control method require care in initial analysis and design. This article introduced some of the important issues in the use and deployment of ABAC. Readers may find a more complete treatment of these issues in the Guide to Attribute Based Access Control [1].

Acknowledgment: This article was derived from the Guide to Attribute Based Access Control, NIST SP800-162 [1]. We are grateful to Adam Schnitzer, Ken Sandlin, Robert Miller, and Karen Scarfone for their contributions to the publication.

Disclaimer: Products may be identified in this document, but identification does not imply recommendation or endorsement by NIST, nor that the products identified are necessarily the best available for the purpose.

## References

[1] Hu, V. C., Ferraiolo, D., Kuhn, R., Schnitzer, A., Sandlin, K., Miller, R., & Scarfone, K. (2014). Guide to attribute based access control (ABAC) definition and considerations. *NIST Special Publication*, 800-162, January 2014.

[2] V. Hu, D. Ferraiolo, R. Kuhn, "Attribute Based Access Control", *IEEE Computer*,

[3] The eXtensible Access Control Markup Language (XACML), Version 3.0, OASIS Standard, January 22, 2013, <URL: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf

[4] Information technology - Next Generation Access Control - Functional Architecture (NGAC-FA), INCITS 499-2013, American National Standard for Information Technology, American National Standards Institute, March 2013.

[5] Working DRAFT Information technology - Next Generation Access Control –Generic Operations and Data Structures (NGAC-GOADS)), INCITS 499-2013, American National Standard for Information Technology, American National Standards Institute, April 2014.

[6] D.F. Ferraiolo, .I. Gavrila, V.C. Hu, and D.R. Kuhn, "Composing and Combining Policies Under the Policy Machine," *Tenth ACM Symposium on Access Control Models and Technologies (SACMAT '05)*, Stockholm, Sweden, 2005, pp. 11-20. https://csrc.nist.gov/staff/Kuhn/sacmat05.pdf [accessed 3/6/14].

[7] D.F. Ferraiolo, V. Atluria, and S.I. Gavrila, "The Policy Machine: A Novel Architecture and Framework for Access Control Policy Specification and Enforcement," *Journal of Systems Architecture*, vol. 57, no. 4, pp. 412-424, April 2011. http://dx.doi.org/10.1016/j.sysarc.2010.04.005

[8] D. Ferraiolo, S. Gavrila, W. Jansen, NIST IR 7987, "Policy Machine: Features, Architecture, and Specification " May 2014