Using a Policy Machine to Enable an Enterprise-wide, Data Centric Operating Environment

David Ferraiolo and Serban Gavrila, ***US National Institute of Standards and Technology***
Wayne Jansen, ***Booz Allen Hamilton***

A primary objective of enterprise computing (via a data center, cloud, etc.) is the controlled delivery of data services to its users. Data services (DSs) are capabilities consisting of operation and object pairs that when executed, enable the reading, manipulation, computation, presentation, management, and sharing of data. Typical DSs include applications such as email, workflow management, enterprise calendar, and records management, as well as system level features, such as file, access control and identity management. Although access control (AC) currently plays an important role in securing DSs, if properly envisaged and designed, access control can serve a more vital role in computing than one might expect. That is, the program logic that deals with implementation, distribution, and enforcement of capabilities of individual DSs can be accommodated by a single AC framework. The Policy Machine (PM), a framework for AC developed at NIST, has been designed with this objective in mind. The PM has evolved beyond just a concept to a prototype implementation and is now being directed toward an open source project.

To appreciate the PM's benefits in computing, it is important to recognize the methods in which DSs are delivered today. Each DS runs in an Operating Environment (OE) and an OE can be of many types (e.g., operating systems, web services, middleware, and database and database applications), each implementing its own routines to enable the execution of DS specific operations (e.g., read, send, and view) on their respective data types (e.g., files, messages, and fields). To impose control over executions of DS capabilities, each OE typically implements a method for identifying and authenticating its users. In addition to authentication, many OEs implement finer grained controls to selectively limit a user's ability to perform operations on objects.

This heterogeneity among OEs introduces a number of administrative and policy enforcement challenges and user inconveniences. Administrators must contend with a multitude of security domains when implementing access policy, and ordinary users and administrators alike must authenticate to and establish sessions within different OEs in order to exercise legitimate DS capabilities. Even if properly coordinated across OEs, access control policies are not always globally enforced. An email application may, for example, distribute files to users regardless of an operating system's protection settings on those files. Also, while researchers, practitioners, and policy makers have specified a large variety of access control policies to address real-world security issues, only a relatively small subset of these policies can be enforced through off-the-shelf technology, and even a smaller subset can be enforced by any one OE.

It is our experience that the PM can provide an enterprise-wide OE that dramatically alleviates many of the administrative, policy enforcement, data interoperability, and usability issues that enterprises face today.

What is the Policy Machine?

Like most other AC mechanisms, the PM comprises: (1) AC data used to express access control policies and deliver capabilities of DSs to perform operations on objects; (2) a set of administrative operations for configuring the AC; and (3) a set of functions for enforcing policy on requests to execute operations on objects and for computing access decisions to accommodate or reject those requests based on the current state of the AC data.

What distinguishes the PM from other mechanisms are the data elements and relations that define its AC data, the type of operations that are recognized, and the functions that it implements. These specifics are driven by a redefinition of AC and DSs in terms of what is believed to be their common and underlying elements, relations, and functions.

What Can the Policy Machine Do?

The PM offers a number of properties. It provides an enterprise-wide Operating Environment (OE) that can implement and execute capabilities of arbitrary DSs and can specify and enforce mission-tailored AC policies over those executions through configuration of its AC data alone. The PM-enabled OE is object type agnostic—the data of its DSs naturally interoperate, and users can view and consume all data in a manner consistent with the defined policies, under a single authenticated session.

How is this Possible?

Among other concepts that make this possible is the fact that the data types of different DSs (e.g., files, messages, fields), are fundamentally just data, and many of the operations of different DSs can be implemented as simple read or write operations on data or as sequences of administrative operations that alter the access state in which users can read or write data. As such, an OE that implements read and write routines, and an AC that controls users capabilities to execute read or write operations on data objects can implement a large variety of DS operations. These DS operations not only include create, read, write and delete operations (typical in operating systems), but also operations such as send, forward, approve, and reject that are typical in applications and middleware. Other kinds of operations (e.g., font manipulation, spell checking, ordering by date or sender) must be implemented in DS logic alone.

While essential, the ability to abstract DS operations from read, write and administrative operations is not sufficient to meet the properties of the PM-enabled OE. In contrast with other AC mechanisms, many features of DSs can be represented and treated as AC data and implemented by the PM. These features include the concept of containment in expressing DS capabilities.

Containers

Common to AC policies and DSs is the notion of user containers and data object containers that characterize and group their members. User containers serve as access control attributes to distinguish classes of DS users. User containers may represent roles such as Doctor or Teller; affiliations such as Divisions or Teams; or even a user’s name such as Smith. Data object containers serve as data object attributes and also as DS data types. Data object containers may represent sensitivities like Secret or Proprietary, but can also represent folders, inboxes, table columns, or records. The PM explicitly recognizes these containers as elements in its AC data.

The PM further recognizes that users and objects may be assigned to one or more containers, and containers may be contained by or contain other containers. For objects, this enables the representation of complex data structures such as relational database tables or forms with distinguished fields.

Defining Capabilities

The PM specifies DS capabilities and AC policies in terms of association relations. Associations are triples of the form (*user container*, *ops* ¯{*read*, *write*}, *data object container*). As examples, (Smith, {read}, Smith Inbox) enables Smith to read the content of his/her Inbox, and (Doctor, {read, write}, Medical Records), enables Doctors to read and write medical records.

The PM also recognizes another kind of container referred to as a policy class. Policy classes map user and object containers to policy reference points as a way to organize the DSs and AC policies through containment, allowing policies to be combined and enforced in a consistent and comprehensive manner.

Derivation of capabilities through associations, policy classes, and combinations of policies also enable fine grained expressions of capabilities for accessing complex data structures, such as relational database tables where specific users are limited to performing specific operations on specific fields of specific records. For example, in an email system serving a medical establishment, users that receive a message containing an attached medical record would only be able to read and/or write fields in accordance with their role, identity, and assigned ward.

PM's Data and Relations

The set of PM data and relations used in expressing and enforcing policy goes beyond those already mentioned [1]. Features include access control prohibitions on users and processes pertaining to classes of objects, and automation of administrative actions based on access control events. The PM is capable of supporting a wide-range of policies including well documented policies such as Role-based Access Control [2, 3], Discretionary Access Control [4], and Mandatory Access Control [5], as well as combinations of those policies. The PM can also accommodate Separation of Duty, Conflict-of-Interest, data tracking, and confinement policies, and should likely accommodate other unanticipated policies of the future.

"Cloud Like" Deployment

Although the PM can be implemented in a number of architectures, NIST has implemented its prototype within a virtualized environment providing "Cloud Like" features. In particular, our infrastructure is an OE in which the PM's functional components run in virtual machines. In this deployment, users and data objects can be provisioned, and DSs can be selected by the subscriber. DSs can be provided as SaaS or PaaS if they conform to the PM's API (i.e., read, write, or administrative operations). Our PM motivated cloud differs from other types of clouds in the properties that it provides to its subscribers (see What Can the Policy Machine Do?) as well as the degree of control offered. AC Policies can be imported from a library of predefined configurations, or can be configured from scratch by the subscriber, conferring PM the attributes of a POLICYaaS provider.

Summary and Conclusion

Through an experimental implementation, NIST has demonstrated that AC can play a more fundamental and consequential role in computing than it currently does. The PM has been designed to showcase such an AC mechanism, enabling an enterprise-wide OE that can execute capabilities of arbitrary DSs, and that can specify and enforce arbitrary, but mission-tailored AC policies over those executions.

Perhaps what is most appealing about the AC framework are the properties that it offers—users and objects are global, the framework is object type agnostic, DSs naturally interoperate, and AC policies are managed and enforced comprehensively across all DSs.

The practical advantages of this PM-enabled OE are many. Through a single authenticated session, users are offered capabilities of a variety of DSs to include office applications, file management, email, workflow, and records management. Data is naturally protected across DSs. Instead of deploying and managing different AC schemes for different DSs, select capabilities (of different DSs) are delivered to select users, under combinations of arbitrary, but mission-tailored forms of discretionary, mandatory, and history-based ACs. This interoperability property is not achieved through features or interfaces built into the DS, but rather through the OE that inherently provides a foundational basis for interoperability.

References

[1] David Ferraiolo, Vijayalakshmi Atluri, Serban Gavrila, The Policy Machine: A novel architecture and framework for access control policy specification and enforcement, Journal of Systems Architecture, 2010

[2] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, R. Chandramouli, Proposed NIST standard for role based access control, ACM Transactions on Information and System Security (TISSEC) 4 (3) (2001) 224–274.

[3] ANSI INCITS 359-2004, Role-Based Access Control.

[4] G. Scott Graham, Peter J. Denning, Protection: Principles and Practice, AFIPS Spring Joint Computer Conference, pp. 417-429, May 1972.

[5] Department of Defense, *Trusted Computer Security Evaluation Criteria*, DoD 5200.28-STD, 1985.