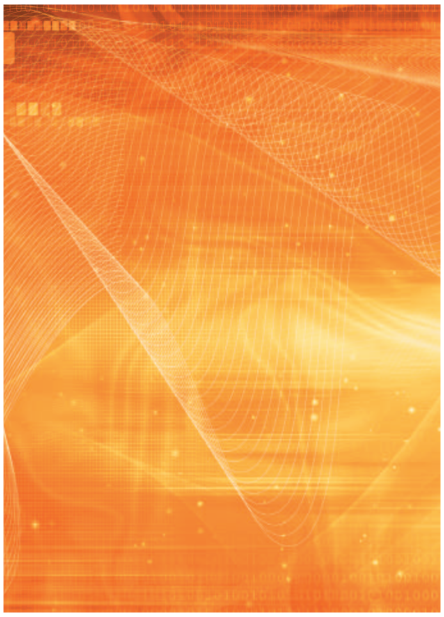

Enabling an Enterprise-Wide, Data-Centric Operating Environment

David Ferraiolo and Serban Gavrila, *National Institute of Standards and Technology*

Wayne Jansen, *Booz Allen Hamilton*

The Policy Machine can execute arbitrary data services and specify and enforce arbitrary but mission-tailored access control policies over those executions.

A primary objective of enterprise computing via a datacenter, the cloud, and so forth is the controlled delivery of data services—operations on objects that enable data reading, manipulation, presentation, management, and sharing. Typical DSs include applications such as email, workflow management, enterprise calendars, and records management as well as system-level features such as file, access control (AC), and identity management.

AC currently plays an important role in securing DSs; however, if properly conceived and designed, it can serve an even more substantial function in computing. A single AC framework can accommodate the program logic that deals with the implementation, distribution, and enforcement of individual DS capabilities.

The National Institute of Standards and Technology (NIST) has developed the Policy Machine (PM) with this objective in mind. The PM evolved from a concept to a prototype implementation and is now an open source project.

DATA SERVICES

To appreciate the PM's benefits, it's important to recognize the way DSs are delivered today.

Each DS runs in an operating environment (OE), such as an operating system, a Web service, middleware, or database and database applications. The OE implements its own routines to enable the execution of DS-specific operations such as read, send, and view on different data types—for example, files, messages, and fields.

To impose control over DS execution, each OE typically implements a method to identify and authenticate its users. Many OEs also implement finer-grained controls to selectively limit a user's ability to perform operations on objects.

This heterogeneity among OEs introduces several administrative challenges as well as user inconveniences. Administrators must contend with multiple security domains when implementing access policy, and ordinary users and administrators alike must authenticate to and establish sessions in different OEs to exercise legitimate DS capabilities.

Even if AC policies are properly coordinated across OEs, they aren't always enforced globally. For example, an email application might distribute files to users regardless of an operating system's protection settings on those files. Also, although researchers, practitioners, and policymakers have specified various AC policies, only a small subset of these can be enforced using off-the-shelf technology, and any one OE can enforce only an even smaller subset.

THE POLICY MACHINE

The PM provides an enterprise-wide OE that can dramatically alleviate many of these issues. Like most other AC mechanisms, it consists of

- AC data used to express AC policies and deliver DS capabilities to perform operations on objects,
- a set of administrative operations for configuring the AC, and
- a set of functions for enforcing policy on requests to execute operations on objects as well as for computing access decisions to accommodate or reject those requests on the basis of the AC data's current state.

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

The PM can implement arbitrary DS capabilities and can specify and enforce mission-tailored AC policies over these executions through configuration of its AC data. The PM-enabled OE is object-type agnostic—users can view and consume all data regardless of their type in a manner consistent with the defined policies under a single authenticated session.

What makes this setup possible is the fact that the different DS data types are fundamentally just data. Many DS operations 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 offers read and write routines and an AC that controls user capabilities to execute read or write operations on data objects can implement a large variety of DS operations. These include not only create, read, write, and delete operations that are typical in operating systems but also operations such as send, forward, approve, and reject that are commonly found in applications and middleware. Other kinds of operations, such as font manipulation, spell checking, and ordering by date or sender, must be implemented in DS logic.

USER AND DATA OBJECT CONTAINERS

Although essential, the ability to abstract DS operations from read, write, and administrative operations isn't sufficient for PM-enabled OE properties. In contrast to other AC mechanisms, PM can implement many DS features as well as represent and treat them as AC data. These features include containers to express DS capabilities.

The policy machine can support a wide range of well-documented policies including role-based, discretionary, and mandatory access control as well as combinations of these.

User and data object containers that characterize and group their members are common in AC policies and DSs. User containers serve as AC attributes to distinguish DS user classes. They can represent roles, such as doctor or bank teller; affiliations, such as divisions or teams; or even a user's name, such as Smith. Data object containers serve as data object attributes as well as DS data types. They can represent sensitivities, such as 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 might be assigned to more than one container, and containers might 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 AND POLICIES

The PM specifies DS capabilities and AC policies in terms of association relations. Associations are triples of the form (*user container*, *ops*, *data object container*), where ops is a subset of {*read*, *write*}. For example, (Smith, {read}, Smith Inbox) lets Smith read the content of his or her inbox, and (Doctor, {read, write}, Medical Records) lets doctors read and write medical data.

The PM also recognizes another kind of container—policy class. A policy class maps user and object containers to policy reference points to organize the DSs and AC policies through containment. This makes it possible to combine and enforce policies in a consistent and comprehensive manner.

Deriving capabilities through associations, policy classes, and combinations of policies also enables fine-grained expressions of capabilities to access complex data structures, such as relational database tables in which specific users are limited to performing specific operations on specific fields of specific records. For example, in a medical establishment's email system, users who receive a message containing an attached medical record would only be able to read and write fields in accordance with their role, identity, and assigned ward.

The PM uses other types of data and relations to express and enforce policy (D. Ferraiolo, V. Atluri, and S. Gavrila, "The Policy Machine: A Novel Architecture and Framework for Access Control Policy Specification and Enforcement," *J. Systems Architecture*, vol. 57, no. 4, 2011, pp. 412-424). Features include AC prohibitions on users and processes pertaining to object classes and automation of administrative actions on the basis of AC events.

The PM can support a wide range of well-documented policies including role-based, discretionary, and mandatory AC as well as combinations of these. The PM can also accommodate separation of duty, conflict of interest, data tracking, and confinement policies and can likely accommodate other unanticipated policies in the future.

CLOUD-LIKE DEPLOYMENT ENVIRONMENT

The PM can be implemented in many architectures; NIST has implemented its prototype in a virtualized OE providing cloud-like features. In particular, the PM's functional components run in virtual machines. In this deployment, administrators can provision users and data objects, and subscribers can select DSs.

The PM provides DSs as software as a service or platform as a service if they conform to its API—that is, its read, write, or administrative operations. The PM's cloud-like environment differs from other cloud types in the properties it provides to subscribers—for instance, data interoperability and policy enforcement across DSs and single-sign-on)—as well as the degree of control it offers. Administrators can import AC policies from a library of predefined configurations, or subscribers can configure them from scratch, conferring to PM the attributes of a "policy as a service" provider.

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

Perhaps what's 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 across all DSs.

The PM offers many practical advantages. Through a single authenticated session, users can access various DSs including office applications, email, and file, workflow, and records management. The PM naturally protects data across DSs. Instead of deploying and managing different AC schemes for different DSs, it delivers different DS capabilities to select users, under combinations of arbitrary but mission-tailored forms of discretionary, mandatory, and history-based ACs. The PM achieves this not through features or interfaces built into DSs but rather through the OE, which inherently provides a basis for data interoperability. C

David Ferraiolo *is a computer scientist and manages the Secure Systems and Applications Group of the Computer Security Division at the National Institute of Standards and Technology. Contact him at david.ferraiolo@nist.gov.*

Serban Gavrila *is a computer scientist in the Computer Security Division at the National Institute of Standards and Technology. Contact him at serban.gavrila@nist.gov.*

Wayne Jansen *is a computer scientist at technology consultancy Booz Allen Hamilton. Contact him at Jansen_wayne@bah.com.*

Showcase Your Multimedia Content on Computing Now!

IEEE Computer Graphics and Applications seeks computer graphics-related multimedia content (videos, animations, simulations, podcasts, and so on) to feature on its Computing Now page, **www.computer.org/portal/web/computingnow/cga**.

If you're interested, contact us at **cga@computer.org**. All content will be reviewed for relevance and quality.

IEEE Computer Graphics and Applications

Editor: Jeffrey Voas, National Institute of Standards and Technology; j.voas@ieee.org