Policy Machine

Towards a unifying access control mechanism

David Ferraiolo
National Institute of Standards and Technology

dferraiolo@nist.gov

Access Control: State of Practice

- The ability to control access to sensitive data in accordance with policy is perhaps the most fundamental security requirement
- However, specification and enforcement of enterprise policy remains in a dismal state of affairs.
- Most approaches have been ad hoc or have focused on management issues and/or specific policy problems and/or environments
- Controls as implemented are not comprehensive, typically do not offer control at the process/inter-process level, and/or lack expressive power.
- For instance, a user with read access to data can typically make a copy of that data and paste its contents into an email message and send it to anyone else in the world, regardless of enterprise policy, and a user process can can do anything its user can without the user's knowledge.

Policy Requirements

- Policy enforcement is instrumental in preventing the unauthorized disclosure of sensitive data, protecting the integrity of vital data, mitigating the likelihood of fraud, protecting privacy of individuals, and is what ultimately enables the sharing of information.
- The actions of users and processes may be governed under multiple policies and enterprise objects may be protected under multiple policies
- One size does not fit all – policies are enterprise and mission specific

Policies are also complex

- To perform an operation on an object, policy may dictate, for example that a user: has a need-to-know, is appropriately cleared, is competent, has not performed a different operation on the same object, the object was previously accessed by a different user, is incapable of accessing other enterprise objects, or is only capable of accessing an object or any copy of the object while in performance of a specific task.
- Enforcement must consider processes (possibly malicious) that actually access data.

Policy Problem

- While over the past four decades a large variety of policies and policy models have been proposed to address real world security issues, only a small subset of these policies are enforceable through off-the-shelf technologies
- Writing down policy and faithfully enforcing policy are different things!

Example Policies and Models

- – DAC
- – MLS
- – RBAC
- – SoD
- – Work flow
- – ORCON
- – OMB M-07-16
- – Chinese wall (conflict of interest)
- – ect.

Data Leakage

- The leakage of read access can occur through malicious or complacent user actions or malicious or flawed software.
- Can undermine the control objectives of a wide variety of policies
  – E.g., **Types of RBAC** - “only doctors can read medical records”, “only top secret users can read top secret data”, **Privacy and ORCON** - “I know who can currently read my personal information”, or **Conflict of Interest** - “a user with knowledge of information within one dataset can not read information in another dataset”.

Interoperability Problem

- A natural characteristic of dispersed heterogeneous mechanisms is a lack of interoperability.
- This lack of interoperability results in many of the identity and policy (privilege) management issues that enterprises struggle with today, as well as the lack of comprehensive policy enforcement.
- Email and external storage devices are big holes
- Processes and inter-process communication (e.g., copy and paste) are most often not controlled

Access Control Mechanisms are Logical Machines

- OS or application access control mechanisms can be thought of and analyzed as a complete and logically independent machine to that of its host environment
- Each of these machines include:
  - Access control data for the expression of policy, and
  - A set of functions for computing access control decisions based on a process request and the configuration of the data, and
  - Enforcement of policy based on these decisions
- Problem:
  - Each machine expresses policy, computes decisions and enforces policy differently

The Policy Machine (PM)

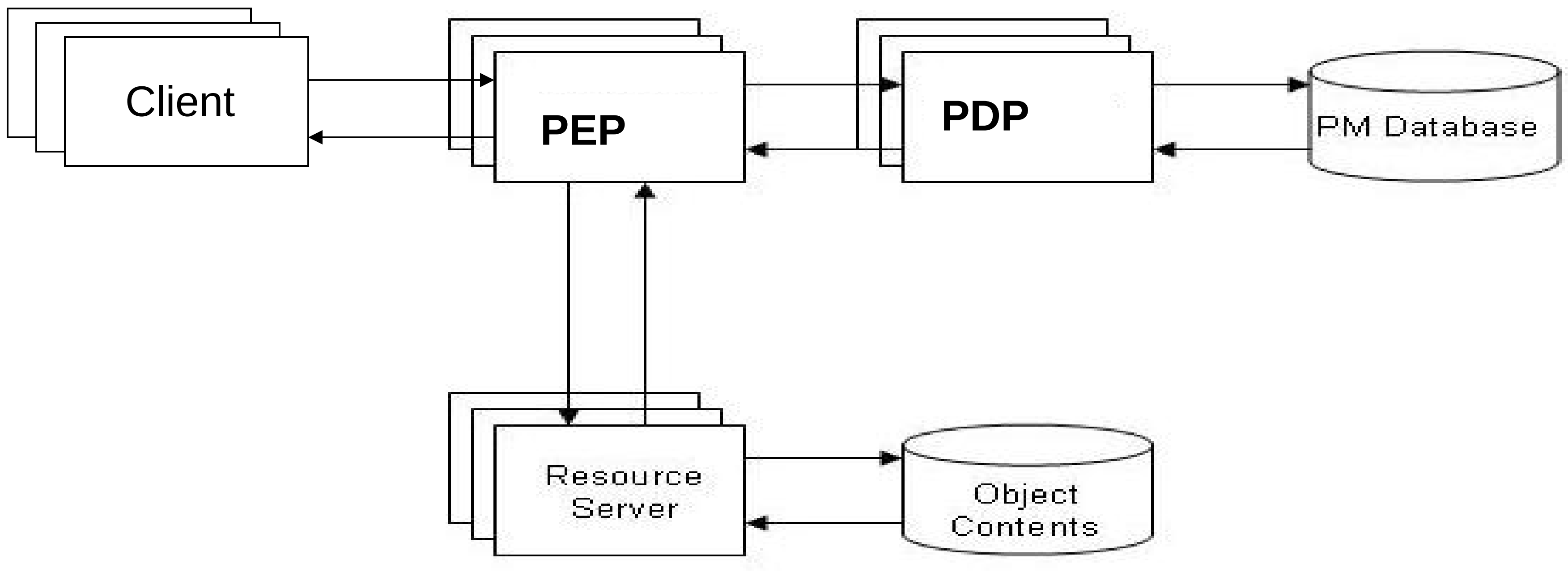
A logical “machine” comprising:

- a fixed set of data and relations used to express (combinations of) attribute-based policies
- a fixed set of administrative operations for configuring the data and relations
- a fixed set of functions for making access control decisions and enforcing the policies

PM Features

- One single mechanism for a large variety of policies
  - Standard interfaces, interoperability
- Controls processes
- Per-user and per-object reviews possible
- Dynamic update of policy configuration:
  - History-based policies
  - Confinement policies

General Architecture

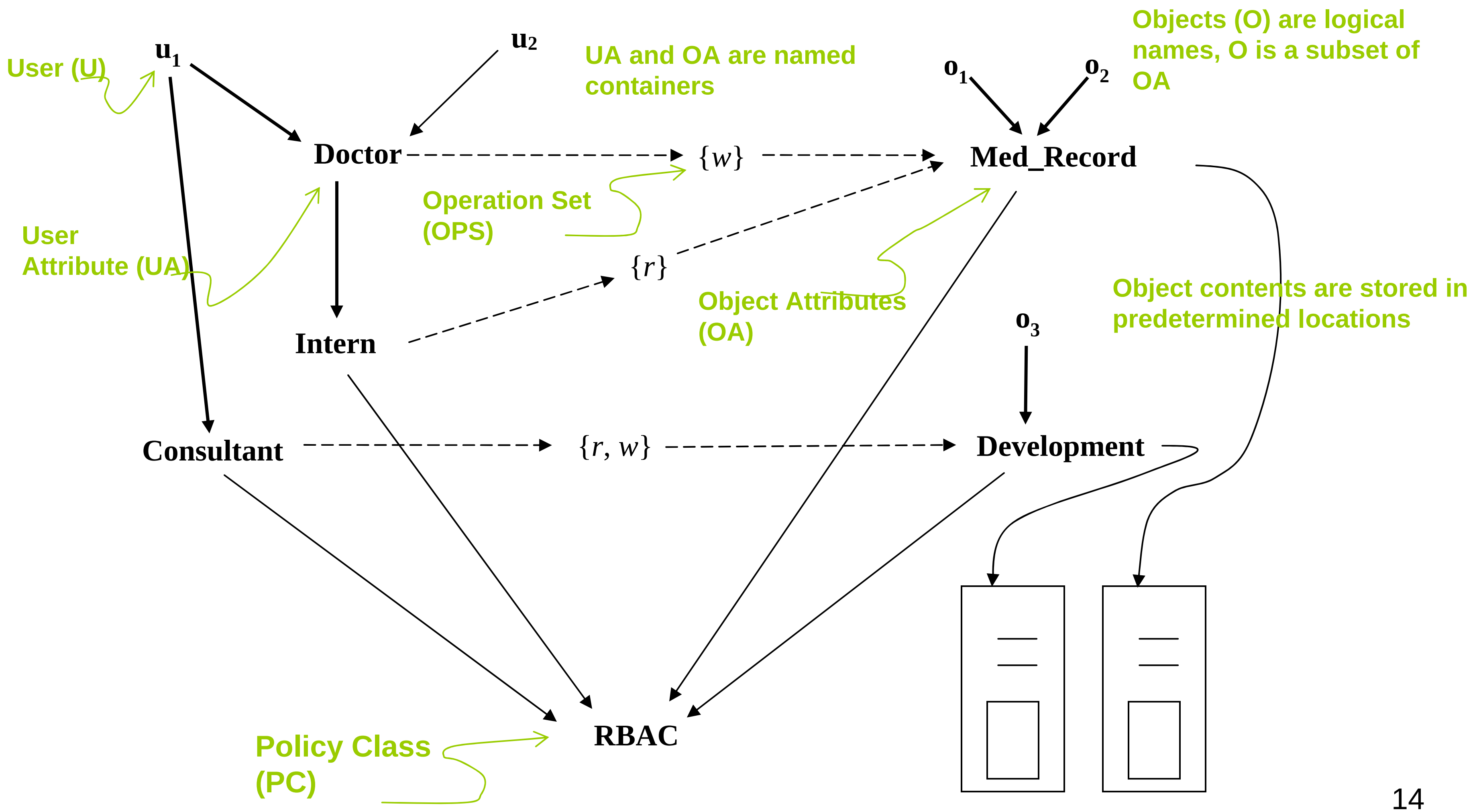


Can be implemented in a wide variety of environments (e.g., virtual, cloud, SOA, OS)

PM Data & Relations

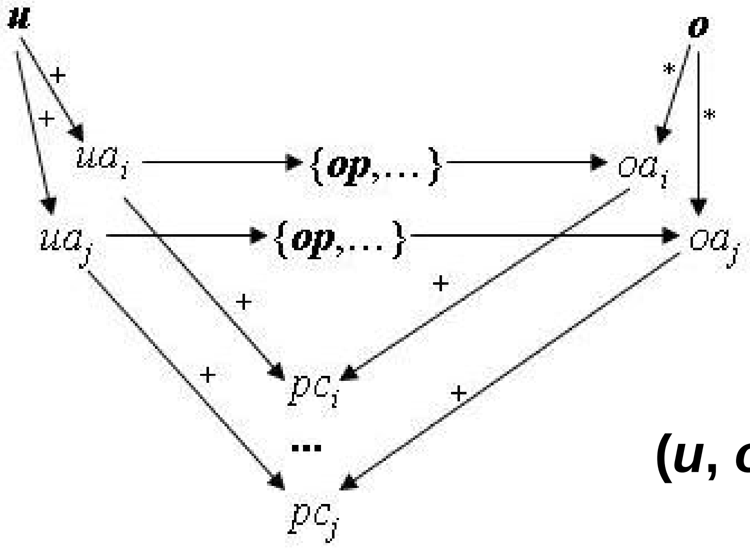
- Data sets
  - Users, objects, attributes, operations, policies
- Relations
  - Assignments (used to derive permissions)
  - Prohibitions
  - Obligations (Event/Response)

Example of Assignments



Deriving Permissions From Assignments

A triple (*u*, *op*, *o*) where *u* is a user, *op* is an operation, and *o* is an object, is a PM permission iff for each policy class pc_k under which *o* is protected, user *u* has an attribute ua_k in pc_k, object *o* has an attribute oa_k in pc_k, and there exists an operation set ops_k containing *op* that is assigned to both ua_k and oa_k.



Note: Users and objects need not be included in all policies

(*u*, *op*, *o*) is a PM permission

Prohibitions (Denies)

- User denies
  - u-deny(*u*, *opset*, *oset*}). Any process executing on behalf of user *u* cannot perform any operation in *opset* on any object in *oset*.
- Process denies
  - p-deny(*p*, *opset*, *oset*). Process *p* cannot perform any operation in *opset* on any object in *oset*.
- The object set can be specified as the complement of *oset*, meaning that the user or process can only perform the operations in *opset* on objects in *oset*.

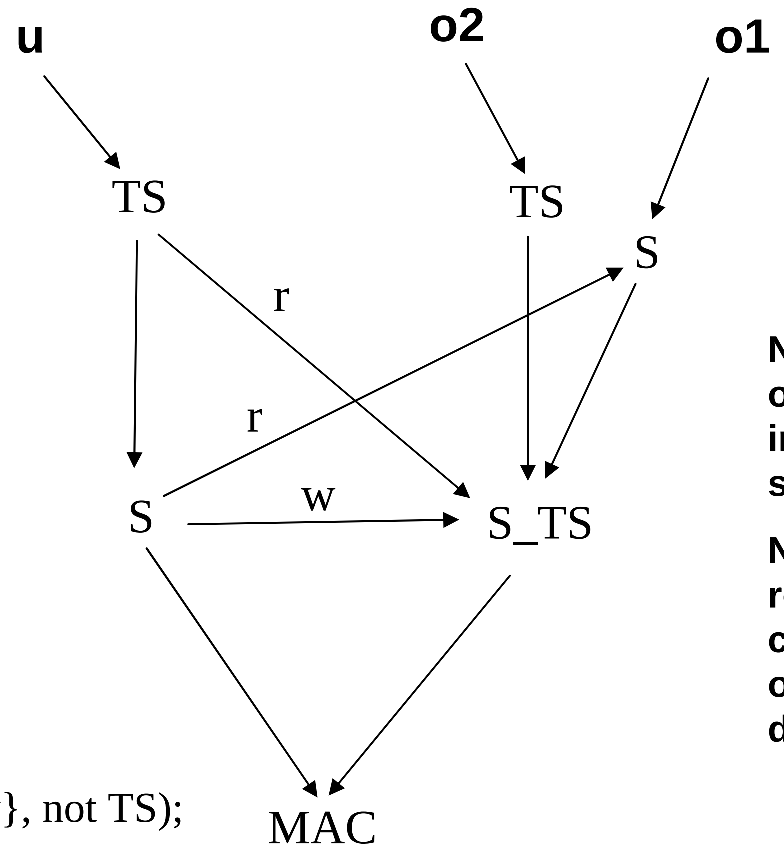
Computing an Access Decision

A process access request $<op, o>_p$ is granted if and only if there is a PM permission (u, op, o) where u is process p's user, and (op, o) is not denied to u or p.

Obligations (Event-Response)

- Format: **when** *event-pattern* **do** *response*
- Event: successful execution of an operation (e.g., reading of an object’s content, or creation of a user).
- Event pattern: the context in which an event occurs (operation, object, user, containers, etc.)
- Response: sequence of administrative operations that may dynamically change the configuration of PM relations.
- Example: **when** process reads object from “Top Secret” **do** create p-deny(process, {write}, not “Top Secret”).

Example MAC Policy



Note1: Unclassified objects are not included in the scope of the policy

Note2: A user can read a TS object and can still write to a S object through a different process

Event pattern ♏ response
when: read object in TS **do:**
  create p-deny(crt process, {w}, not TS);
When: read object in S **do:**
  create p-deny(crt process, {w}, not S_TS).

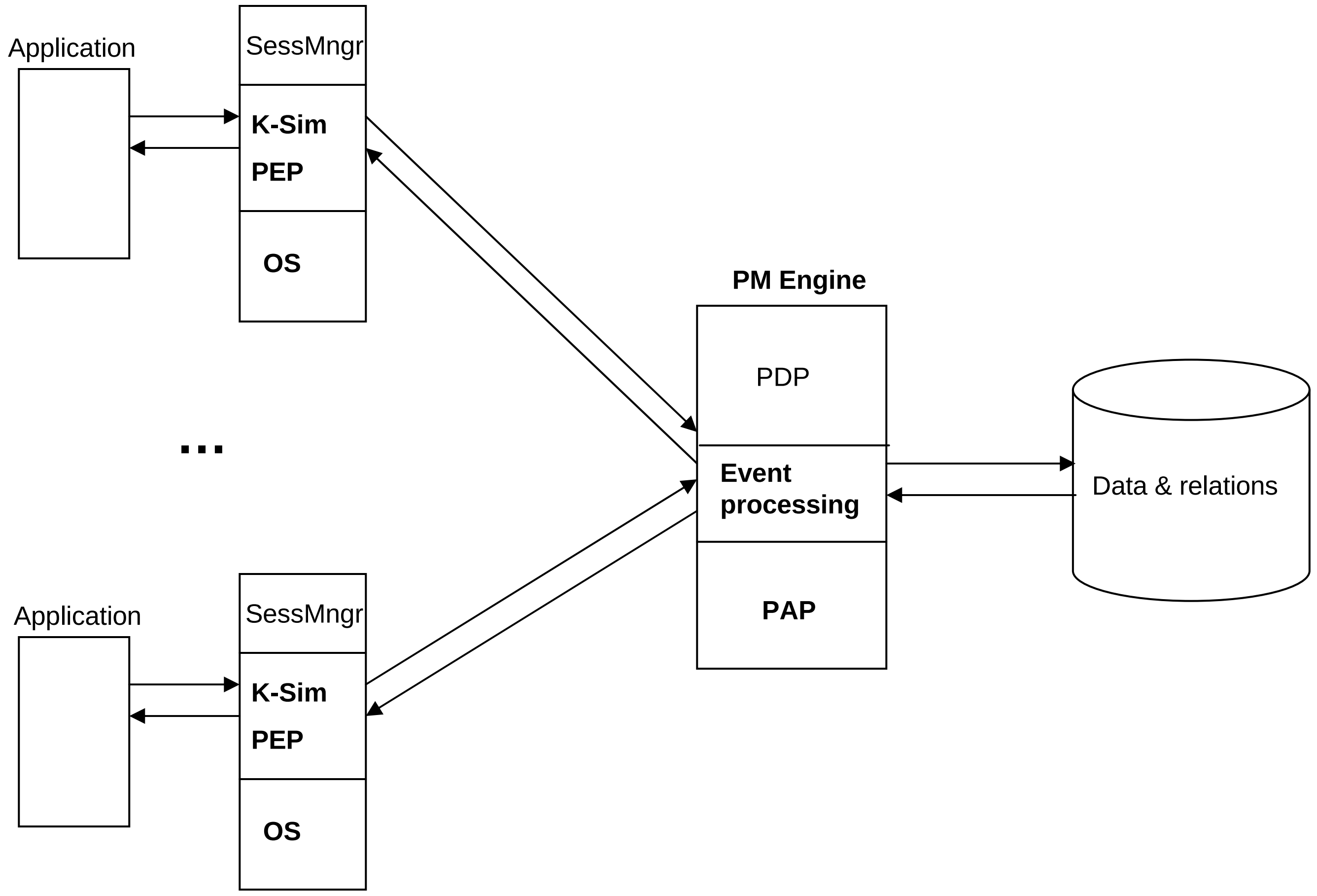
PM Applications

- May use new system calls (openObject(), etc.): word processors
- May use policy configurations to provide access control services (an email client, form and record management apps, workflow management app).

Requirements

- PEP cannot be bypassed in accessing PM objects
- PM applications cannot access non-PM objects
- {PEP satisfies client request, PEP notifies PDP about this event, PDP processes this event} is a transaction

Reference Implementation Architecture



Our Reference Implementation

- We can demonstrate the expression and enforcement of a wide variety of policies (e.g., instances, combinations and hybrids of DAC, MAC, RBAC, Chinese wall, ORCON, history-based Separation of duty, OMB M-07-16, etc.)
- Policies are not only enforced on files, but comprehensively enforced across a rich user environment that includes the Open Office suite of applications, email, workflow, records, and forms management, and Copy/Cut & Paste
- General forms of confinement (e.g., only doctors can read medical records, I know who has access to my data and I can revoke access)

How does the PM work

- User logs on to the PM, through any PM client
- PM presents the user with or allows the user to quarry all his/her accessible resources (e.g., files, inbox, work items …) This is a logical view called the Personal Object System (POS)
- User, u requests access to resources through a process request <op, o>p where p is u’s process. The physical location of the resource is transparent to the user
- PM grants the request iff (1) permission (u, op, o) exists as a derivation of assignment relations, (2) there does not exist a user deny (u, {op}, {o}), or (3) a process deny (p, {op}, {o}), where op♍{op} and o ♍{o}.
- Machine policy state may dynamically change as a consequence of a successful access
- Policy is created through data configuration alone
- Library of policies

Benefits to the User

- General Purpose Protection Machine (one mechanism fit for many purposes)
- Large library of policies available for immediate configuration
- Naturally provides interoperability and single sign-on
- Addresses the insider threat (Société Générale, Barings PLC)
- Operational Assurance
  - Can render many Trojan horse attacks harmless
  - No enforcement or decision making at the application level
  - Can prevent "leakage" of sensitive data to unauthorized principals, through email, and storage devices (hard-drives, memory sticks)(can view but can't store locally)
- Fine-grained, flexible and comprehensive protection
  - One scope of control with logically decentralized administration
- Promotes greater sharing of information (through protection)
- Promotes greater sharing of computers (through logical access)
- Truly secure application services through PM configuration and enforcement
- Access information through any PM client
- Data can exist anywhere, but locally discovered
- Collaborations rather than federations

Benefits to the Vendor

- OS Vendor
  - No need to change system to accommodate the policy de jour,
  - No need to cater to special needs of different user communities
  - No need to make access control decisions, or maintain or manage access control data
- Application developers
  - No need to provide functionality for making access control decisions or policy enforcement
  - No need to maintain or manage access control data

Status

- Fully Functional Reference Implementation
- Formal ANSI/INCITS CS1 Working Group to develop PM family of standards
- SE Linux version is under development by Intelligence Automation Corp. (Phase I SBIR)
- SUN and NIST investigating CRADA
- HHS through Harris Corporation is currently “productizing” PM for protection Healthcare records in Connect architecture
- Collaborating with Rutgers and Purdue