Defence Research and Development Canada | Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

# Options for the Policy Server Component of the DRDC Architecture for Secure Access Management

*Revision 2006*

E. Bacic, S. Klump and A. Magar

**Defence R&D Canada – Ottawa**
CONTRACT REPORT
DRDC Ottawa CR 2006-166
August 2006

Canada

# Options for the Policy Server Component of the DRDC Architecture for Secure Access Management

*Revision 2006*

E. Bacic, S. Klump, A. Magar
Cinnabar Networks

Prepared by:

Cinnabar Networks
(A Division of Bell Security Solutions Inc.)
265 Carling Avenue, Suite 200
Ottawa ON K1S 2E1

Project Manager: Eugen Bacic (613) 236-4567
Contract number: W7714-6-3292
Contract Scientific Authority: Dr. S. Zeber (613) 991-1388

**Defence R&D Canada – Ottawa**

Contract Report
DRDC Ottawa CR 2006-166
August 2006

This work was sponsored by the Network Information Operations section of DRDC Ottawa under project 5BQ.

## Abstract

Defence R&D Canada – Ottawa (DRDC Ottawa) implemented a proof-of-concept system that combines Privilege Management Infrastructure (PMI) technology and Public Key Infrastructure (PKI) technology to demonstrate a caveat separation capability for the defence environment. A key component of the demonstrated system is a policy server product that provides content-based security. This report examines alternative products and solutions for the implemented policy server that would be consistent with the existing policy component of the proof-of-concept system. This report discusses existing technologies from industry, academia, the military, and research laboratories as well as the possibilities and complexities of designing and implementing a work-alike replacement. This report, which is a 2006 revision of a report originally written in 2003, documents progress in policy research over the ensuing three year period.

## Résumé

R & D pour la défense Canada - Ottawa (RDDC Ottawa) a mis en oeuvre un système de validation de concept qui combine la technologie de l'infrastructure de gestion des privilèges (IGP) et la technologie de l'infrastructure à clés publiques (ICP) afin de démontrer une capacité de séparation des communautés d'intérêt dans le contexte de la défense. Un élément clé du système est un produit de serveur de politique offrant une sécurité qui repose sur le contenu. Le présent rapport examine des produits de remplacement et des solutions au serveur de politique qui respectent la politique actuelle du système de validation de concept. Il traite des technologies existantes dans l'industrie, les universités, le monde militaire et les laboratoires de recherches, de même que des possibilités et des complexités de conception et de mise en œuvre d'une alternative similaire. Ce rapport, qui est une révision faite en 2006 d'un compte rendu rédigé en 2003, illustre les progrès réalisés sur la recherche en matière de politique durant ces trois ans.

This page intentionally left blank.

# Executive summary

## Options for the Policy Server Component of the DRDC Architecture for Secure Access Management: Revision 2006

**Bacic, E., Klump, S., Magar, A.,; DRDC Ottawa CR 2006-166; Defence R&D Canada – Ottawa; August 2006. [D9]**

## Introduction

Security has traditionally been viewed as a means of keeping malicious users out of critical computer systems. However, this approach to information security has been challenged over the past couple of years as the requirement for secure information sharing increases. In order to support this paradigm shift organizations must view security in a radically different way; as a means to facilitate secure information sharing between authorized individuals.

Policies can be defined as the rules governing the choices in behaviour of a system. A system that is governed by a policy is obliged under event-triggered conditions to perform particular rules that modify its actions (behaviour). Authorization policies define what services or resources an entity (subject, user, or role) can access. In addition, access violation policies define actions to be taken when a security violation occurs.

Unfortunately, the heterogeneity of today's systems and networks, and the variety of mechanisms implemented to handle access control and authorization, impedes the successful implementation of system-wide security management.

This document investigates a number of options that will allow DRDC to create a policy engine capable of mediating access to sensitive information resources according to the relevant security policy. The primary requirement for any policy engine candidate is that the policy language should be powerful enough to express all classical security policies as well as any new types of policies required. The logic behind the language must ensure that for every policy defined, the mediation result will always be complete, consistent, and deterministic with respect to the question: "is the user authorized to perform the action on the resource?" Furthermore, every policy expression must be relatively simple. It must be possible for anyone with a reasonable computer education and knowledge of security to define valid and strong policies.

This survey, which was originally written in 2003, was revised in 2006 in order to document progress in policy research over the ensuing three year period.

## Results

There are a couple of potentially viable candidates that appear to meet a number of the requirements of a policy engine. However, the true flexibility of each solution can only be determined by attempting to implement a variety of security policies and testing the efficacy of the implementations. In addition, it is anticipated that over time other interesting projects will be

released publicly as Open Source so that security practitioners can examine the workings of some modern interpretations of a security policy engine.

Since the original survey was conducted in 2003 there has been a great deal of activity, representing a corresponding level of interest, in policy engine research. Unfortunately, this activity has not resulted in an equivalent level of progress. While a number of viable candidates have progressed marginally, research on other candidates has ceased entirely. Most importantly, all of this research has resulted in very few actual implementations of generic policy engines capable of supporting complex policies.

Furthermore, much of this policy engine work has focused specifically on access control and confidentiality without addressing management policies. A holistic approach should encompass all aspects of information protection and accessibility, rather than just addressing part of the problem. In addition, a great deal of policy engine work attempts to restrict the definition of data rather than providing a solution that is applicable to the entire spectrum of information. This is at odds with the purpose of a uniform security policy system that can apply consistent policies to information regardless of where the information is stored or how it is accessed.

## Significance

The results of this investigation demonstrate that no single industrial, standards-based, or Open Source candidate stands out as a viable DRDC policy engine.

## Future plans

To continue to advance this project DRDC should perform a detailed examination of a small number of potentially viable policy engines. It is recommended that this detailed examination include efforts to implement a finite selection of classical security policies and new security policies of relevance to DRDC. Based on this detailed examination of viable policy engines, DRDC should be able to not only identify the most promising candidate but determine the level of effort required to extend it to meet DRDC’s specific requirements.

# Sommaire

## Options for the Policy Server Component of the DRDC Architecture for Secure Access Management: Revision 2006

**Bacic, E., Klump, S., Magar, A.,; DRDC Ottawa CR 2006-166; R & D pour la défense Canada – Ottawa; août 2006.**

## Introduction

On voit depuis toujours la sécurité comme un moyen de garder les utilisateurs malveillants loin des systèmes informatiques essentiels. Cette approche à l'égard de la sécurité de l'information est cependant remise en question depuis quelques années, à mesure que croît l'échange d'information sécurisée. Afin d'appuyer ce changement de paradigme, les organisations doivent considérer la sécurité de façon totalement différente, comme un moyen de faciliter l'échange d'information sécurisée entre des personnes autorisées.

On peut définir les politiques comme étant les règles qui régissent les choix dans le comportement d'un système. Un système régi par une politique doit, selon des conditions déclenchées par des événements, exécuter certaines règles particulières qui modifient ses actions (comportement). Les politiques d'autorisation définissent les ressources ou les services auxquels une entité (sujet, utilisateur, rôle) peut avoir accès. En outre, les politiques relatives à la violation des accès définissent les mesures à prendre lorsqu'un manquement à la sécurité survient.

Malheureusement, l'hétérogénéité des systèmes et des réseaux d'aujourd'hui de même que la variété des mécanismes mise en place pour assurer le contrôle et l'autorisation des accès nuisent à la mise en œuvre adéquate de la gestion de la sécurité à l'échelle des systèmes.

Le présent document analyse un certain nombre d'options qui permettront à RDDC de créer un moteur de politique capable de négocier l'accès à de l'information de nature sensible, conforme à une politique appropriée en matière de sécurité. Le besoin principal de tout candidat du moteur de politique est que la langue de la politique soit suffisamment forte pour exprimer toutes les politiques de sécurité classiques ainsi que tous les nouveaux types de politique nécessaires. La logique derrière la langue doit faire en sorte que, pour chaque politique définie, les résultats de la médiation soient toujours complets, cohérents et déterminants par rapport à la question : « l'utilisateur est-il autorisé à appliquer cette mesure à la ressource »? De plus, chaque expression de politique doit être relativement simple. Il doit être possible, pour quiconque possède des compétences raisonnables en informatique et une connaissance de la sécurité, de définir des politiques fortes et valides.

Cette étude, d'abord rédigée en 2003, a été révisée en 2006 afin d'illustrer les progrès réalisés sur la recherche en matière de politique au cours de ces trois ans.

## Résultats

Il y a quelques candidats potentiellement viables qui semblent satisfaire à quelques-unes des exigences relatives au moteur de politique. Cependant, la souplesse réelle de chaque solution ne peut être déterminée que par la mise en œuvre d'une variété de politiques de sécurité et l'essai de l'efficacité de ces mises en œuvre. De plus, il est prévu, qu'au fil du temps, d'autres projets d'intérêt seront diffusés dans le public en toute liberté de sorte que les spécialistes de la sécurité puissent examiner le fonctionnement de certaines interprétations modernes d'un moteur de politique de sécurité.

Depuis l'étude originale réalisée en 2003, il y a eu beaucoup d'activité, équivalant à un niveau d'intérêt correspondant, en matière de recherche sur les moteurs de politique. Malheureusement, cette activité n'a pas donné le niveau d'avancement proportionnel. Alors qu'un certain nombre de candidats viables ont légèrement progressé, la recherche sur d'autres candidats a cessé complètement. Mais le plus important est que l'ensemble de cette recherche a débouché sur très peu de mises en œuvre réelles de moteurs de politique générique capables d'appuyer des politiques complexes.

De plus, la majeure partie du travail sur les moteurs de politique a porté de façon bien précise sur le contrôle des accès et la confidentialité, sans traiter des politiques de gestion. Une approche holistique devrait englober tous les aspects de la protection de l'information et de son accessibilité plutôt que de ne porter que sur une partie du problème. En outre, une grande partie du travail sur les moteurs de politique tente de restreindre la définition des données plutôt que de fournir une solution applicable à l'ensemble du spectre de l'information. Ceci va à l'encontre de l'objectif d'un système uniforme de politique de sécurité pouvant appliquer des politiques cohérentes à l'information, peu importe l'endroit où est stockée cette information et comment il est possible d'y avoir accès.

## Importance

Les résultats de cette enquête montrent qu'aucun candidat industriel, axé sur les normes ou de source ouverte, ne ressort comme moteur de politique pour RDDC.

## Perspectives

Pour que ce projet continue d'avancer, RDDC doit faire un examen détaillé d'un petit nombre de moteurs de politique potentiellement viables. Il est recommandé que cet examen détaillé comporte une sélection finie de politiques de sécurité classiques et de nouvelles politiques de sécurité appropriées à RDDC. À partir de cet examen détaillé, RDDC devrait être en mesure, non seulement d'identifier le candidat le plus prometteur, mais aussi de déterminer le niveau d'effort requis pour l'adapter aux exigences particulières de RDDC.

# Table of contents

# List of figures

## List of tables

# 1. Introduction

Security currently and historically [Amoroso, Anderson 1972, Bell, Clark-Wilson, CTCPEC, Gasser, Gligor, CC] has been viewed as keeping malicious users out of critical computer systems. This view of restriction rather than sharing is becoming increasingly challenged as the desire for information sharing increases. In order to create such a system it is vital to view security in a radically different way, as a means of sharing information between authorized individuals [Bacic 1989, Bacic 1990a, Bacic 1990b, CTCPEC].

Sloman [Sloman 1994b] eloquently defines policies as the rules governing the choices in behaviour of a system. A system that is governed by a policy is obliged under event-triggered conditions to perform particular rules that modify the actions (behaviour) of a system. Policies can be defined to perform a wide variety of actions from Quality of Service to event monitoring to access control functions (i.e., authorization).

Authorization policies are used to define what services or resources an entity (subject, user, or role) can access. In addition, access violation policies are required that define actions to be taken when a security violation occurs. Simple examples, such as a string of login failures for a given user, to complex ones, such as a concerted attack on the system, require sufficient flexibility to allow administrators the ability to take corrective actions. Unfortunately the heterogeneity of today's systems and networks, and the variety of mechanisms implemented to handle access control and authorization, impedes the successful implementation of system-wide security management.

One system that was designed to address this issue of standardizing access control policies within a heterogeneous environment was Texar SecureRealms. This survey will examine other existing technologies that perform a similar function and how flexible these technologies are in terms of programmable access control policies, extensibility, and interoperability with existing security infrastructure. Unfortunately, even though a lot of literature exists describing generic policy engine-like technologies, few have been implemented beyond the mathematical model stage. For completeness, this document will discuss as many policy engines as were discovered during the survey, including those that are only mathematical models. There are probably a good many other policy engines that have been developed but have very little exposure. For obvious reasons these would be difficult to discover in a reasonable amount of time.

The primary requirement for any policy engine implementing a security policy language is that the language should be powerful enough to express all classical security policies as well as any new types of policies. The logic behind the language must ensure that for every policy defined the mediation result will always be complete, consistent, and deterministic with respect to the question: "is the user authorized to perform the action on the resource?".

Furthermore, every policy expression must be relatively simple. It must be possible for anyone with a reasonable computer education and knowledge of security and policy particulars to define valid and strong policies. Even though it is possible to represent all logical relations (and, or, not, etc.) with nand operators, any non-trivial policy would become extremely complex to express and understand. It is crucial that policy writing be made as easy as possible for the policy writer to define policies in the language of the policy engine. Hence, any security policy language that is examined in this survey must, at a minimum, support rich set of logical expressions.

This survey, which was originally written in 2003, was revised in 2006 in order to document progress in policy research over the ensuing three year period.

# 2. Definition of Policy

The word "Policy" is overloaded in the computer arena. It means one thing to privacy advocates, another to network experts, a third to modellers, a fourth to security experts, a fifth to executives and "policy wonks", and a myriad of other things to a slew of individuals. And to computer experts it means yet another thing, namely the logical, perhaps codified, description of business rules that govern the operational security of a particular platform. For our purposes we will define "policy" as business rules codified into a logic that permits the mediation of requests for access to valuable resources or assets. In this regard policies reflect the corporate intention of adequately protecting valuable assets according to a predefined set of rules and guidelines.

Policies are implemented as persistent code elements that define the behaviour of a system when encountering specific conditions resulting in predefined actions being invoked. These actions can be as innocuous as notifying an administrator or as proactive as changing the functionality of the actual operations themselves. Even though many policy systems described in this survey are simply rule-based systems, in each case policies are persistent in ensuring that the system behaves in a deterministic fashion. So-called one-off policies that perform a given action are not true security policies as they have come to be known. With the higher performance systems currently available it has become possible to create policy languages that are similar to scripting languages which enhance portability and adaptability while introducing new functionality into distributed network components. Such systems allow a single policy to operate in the heterogeneous environments common in today's IT centres.

Typical large-scale IT centres contain millions of resources and users. Although it may seem desirable to specify policies to individual entities, it is almost equivalent in capability if the relationship were restrained to grouping users and resources to a specific policy. In this regard this survey will not limit itself to those systems capable of providing 1:1 relationships between entities and policies but rather also include those systems that allow for the grouping of entities under a given policy while allowing multiple policies to co-exist. In other words, any system that allows multiple policies to be attached to any given grouping of entities will be considered for this survey, whether the relationship is 1:1 or not.

For any policy to be deemed useful it must permit the specification of policies relating to groups of entities or nested groups such as sections within departments, within sites in different countries in an international organization, etc. It is useful, though not necessary, to be able to group policies according to a role's duties or rights within an organization (i.e., network operator, medical clerk, cardiologist, director, staff nurse, etc.).

Policies are derived from business logic – the way in which a business operates. These abstract definitions of "policy" typically relate to specific services, functions, or operations of the organization as a whole. Within these abstract policies is defined what information is sensitive, who should see it, and how it should be protected. Such a policy often does not specify a mechanism to protect the information but rather articulates the sensitivity of the resource to the organization and that it should be "adequately protected". Converting this abstract policy into a precise and explicit specification of codeable logic is what defines the emerging policy engine arena and the systems capable of codifying abstract logic that achieve an organizations security goals.

It is the job of a policy engine actually to manage and to utilize policies. This policy engine provides authorization services for client applications or agents by answering mediation requests with whether or not a particular request for access should be granted. This survey examines the current state of policy engines and policy languages and how they may be applicable to ongoing work at DRDC.

# 3. Caveat Separation Proofs-of-Concept

## 3.1 Overview

In 2000, Defence Research & Development Canada (DRDC) identified Privilege Management Infrastructure (PMI) as an area of research relevant to DND. An initial investigation by Alan Magar [Magar 2001a] provided a broad overview of PMI technology as a whole and described areas of particular interest to DND. A follow-on study examined the use of PMI in conjunction with Public Key Infrastructure (PKI) technology in the classified defence environment [Magar 2001b]. Subsequently, Magar implemented a prototype system, using an architecture proposed in [Zeber 2002] that demonstrated caveat separation using commercial-off-the-shelf (COTS) PMI and PKI components. This prototype was eventually named the Secure Access Management Proof-Of-Concept (SAMPOC) I. The success of this prototype led to the development and demonstration of a second such prototype, SAMPOC II, and finally to the initiation of a Technology Demonstrator (TD) project.

## 3.2 SAMPOC I

The objectives of the SAMPOC I demonstration are detailed in [Magar 2002]. The primary objective was to demonstrate secure information sharing using a combination of COTS PKI and PMI technologies. Secondary objectives included: an investigation of the proposed theoretical model for identity management, authentication, and resource access in accordance with a given security policy; a demonstration of the use of PMI technology combined with PKI technology to support DND requirements; a demonstration of multi-caveat separation using content-based information security; providing advice on the use of content-based information security to achieve multi-level security; and demonstrating how such technologies can be used to facilitate information sharing in coalition environments. SAMPOC I also defined a baseline against which DRDC could perform additional R&D in the area of PKI, PMI, and policy work.

The infrastructure required for SAMPOC I included an Entrust PKI and a Texar SecureRealms PMI. Coupled to these systems were ancillary products from Chrysalis and Oracle as well as miscellaneous personal computer (PC), virtual private network (VPN), and networking hardware. For a full description of the Proof-of-Concept environment interested readers are directed to [Magar 2003a].

## 3.3 SAMPOC II

To address the comments and feedback from SAMPOC I, [Magar 2003b] proposed a new architecture, detailed design and project plan, which was completed in December 2003. A second prototype system, SAMPOC II, was subsequently built and demonstrated in 2004. The results of this demonstration are documented in [Magar 2004a] and [Magar 2004b].

## 3.4 SAMSON TD

Although the SAMPOC I and II demonstrations successfully demonstrated a technical capability, and although they generated substantial interest within DND, these prototype systems were not sufficiently robust, secure, or large enough to be considered for operational deployment. To develop this demonstrated technical capability further towards an operational capability, the Secure Access Management for Secret Operational Networks (SAMSON) technology demonstrator (TD) project was initiated. The SAMSON TD seeks to address the deficiencies of the SAMPOC I & II prototypes and to demonstrate a prototype system that could transition, through an appropriate exploitation plan, to operational use. The SAMSON concept of system operation is documented in [Zeber 2006].

# 4. Policies

## 4.1 Overview

The Orange Book [DoD 1985] and subsequent Criteria [CTCPEC, CC] defined the necessity of an explicit and well-defined security policy that is enforced by the system. Said security policy is typically assumed to be a set of laws, rules, and practices that regulate how an organization manages, protects, and distributes sensitive information. These rules, in a system, are defined as code that defines in logic how the organization expects the computer systems to behave when mediating requests for access to various resources. In other words, constraints are defined as to what a user can do directly, or for that matter, what programs executing on behalf of the user are allowed to do.

Originally the Orange Book defined two basic components that governed all mediation requests handled by a security policy: subject and object. The CTCPEC, and subsequently the Common Criteria, defined a third component: actions. Hence, systems can be reduced to three basic components:

- ***subject*** (e.g., user or process acting on behalf of a user): the uniquely identified user attempting to access a system resource;
- ***action*** (e.g., read, write, execute, etc.): the means by which entities interact with each other; and
- ***object*** (e.g., user, process, file, etc.): the uniquely identified resource that a user is attempting to access.

Often subjects and objects are collapsed into entities, namely active entities (subjects) and passive entities (objects). In fact, this can be simplified into the following expression:

$$\textit{Policy} : (\textit{activeEntity action passiveEntity}) \rightarrow [\textit{allow} \mid \textit{deny}].$$

Since access controls are concerned with allowing only authorized users to access services or resources, they thereby limit the activity of legitimate users who have been successfully authenticated. An authorization or access control policy defines the high-level rules that specify the conditions under which one entity is permitted to access another. Many systems implement their “security policy” not through a proper policy specification but rather through low-level access control lists or similar mechanisms. Even so, the study of these mechanisms has yielded a number of useful policy models, such as Bell-LaPadula and Lee, and has provided the baselines necessary to prove specific properties about access control systems.

Historically, access control policies have been divided into discretionary and mandatory policies. Over the years derivations on these types of policies have gradually been developed. However, in the last few years there has been a profusion of security policies. While many of these new types of policies have been developed for a specific application (e.g., Risk Adaptive Access Control (RAdAC) for the Global Information Grid (GIG)), others have been developed to address emerging privacy regulations (e.g., Sarbanes-Oxley, Health Insurance Portability and

Accountability Act (HIPAA), Personal Information Protection and Electronic Documents Act (PIPEDA)).

This section examines the following types of security policies:

- Discretionary Policies;
- Mandatory Policies;
- Non-Discretionary Policies;
- Administrative Policies;
- Brewer-Nash Chinese Wall;
- Role-based Access Control (RBAC);
- Distributed Access Control;
- Attribute-based Access Control (ABAC);
- Governance-based Access Control (GBAC);
- Risk Adaptive Access Control (RAdAC);
- Network Policies;
- Policy-Oriented Logic; and
- Trust.

## 4.2 Discretionary Policies

Discretionary security models control access to an object on the basis of an individual user's permission and/or denial. Discretionary security models are based on the notion that individual users are "owners" of (passive) entities and therefore have complete discretion over which subject should be authorized to perform which action on which object. Ownership is usually acquired as a consequence of creating the entity.

Discretionary Access Control (DAC) policies restrict access to objects based on the identity of the subjects and/or groups to which they belong, and are discretionary in the sense that a subject can pass its access permissions on to another subject (i.e., the controls are at the discretion of any subject who has current access to an object). Obviously the delegation of access rights is an important part of any system that supports DAC. DAC policies use the access matrix model as a framework for reasoning about the permitted accesses. In the access matrix model the state of the system is defined by a triple (S,O,A), where S is the set of subjects, O is the set of objects and A is the access matrix where rows correspond to subjects, columns correspond to objects and entry A[s,o] reports the privileges of s on o. Discretionary policies do not enforce any control on the flow of information once this information is acquired by a process, making it possible for processes to leak information to users not allowed to read it.

### 4.2.1 Lampson Access Matrix Model

Originally formulated by Lampson in 1971 [Lampson 1971] this policy viewed access control as a large global matrix with rows corresponding to subjects and columns to objects, as shown in Figure 1. Each matrix entry [i,j] states the rights (authorized actions) granted to subject i with respect to object j. An entry is composed of a list of rights with a copy flag attached to each of them. The copy flag is used for the transmission of rights.

***Objects***

***Subjects***

| | Subject 1 | Subject 2 | Subject 3 | File 1 | File 2 |
|---|---|---|---|---|---|
| Subject 1 | Owner * control | Owner * control | Call * | Owner * Read * Write * | |
| Subject 2 | | | Call | Read | Write |
| Subject 3 | | | Owner Control | Read | Owner * |

*Figure 1: Sample Discretionary Access Matrix*

Entries in any access matrix are created and deleted according to the following rules:

1. a subject, $S_1$, can remove access attribute from *Matrix*[$S_2$, j], if it has “control” access to $S_2$ (i.e., subject 1 can remove attributes from subject 1 and subject 2 rows).
2. a subject, $S_1$, can copy to *Matrix*[$S_2$, $O_1$] any access attributes it has for $O_1$ which have the copy flag set (i.e., subject 1 can copy “*write” to *Matrix*[subject 2, file 1]).
3. a subject, $S_1$, can add/remove any access attributes to *Matrix*[i, $O_1$] if it has the “owner” access to $O_1$ (i.e., subject 3 can add “write” to Matrix[subject 2, file 2]).

Note that the copy flag can be considered to be an extension that determines rights propagation and that other discretionary models can describe different mechanisms by which rights may propagate. It is also possible to introduce groups of subjects and/or groups of objects, such as is seen in UNIX. As mentioned above, the primary problem with discretionary policies is that the dissemination of information is not controlled [Castano]. This makes discretionary control vulnerable to malicious attacks such by Trojan Horses, Worms, and Viruses.

## 4.3 Mandatory Policies

Mandatory Access Control (MAC) policies enforce access control on the basis of fixed regulations mandated by a central authority, as typified by the Bell-LaPadula, lattice-based model [Bell-LaPadula]. Lattice-based models were defined to deal with the issue of data confidentiality and concentrate on restricting information flow in computer systems. This is achieved by assigning a security classification to each subject (an active entity that can execute actions) and each object (a passive entity storing information) in the system. Subjects and objects form a lattice based on their classification, which is used to enforce some fixed mandatory policies regarding the actions that subjects can execute on objects. The conceptual framework of the Bell-LaPadula model forms the basis of other derived models such as the Biba Integrity Model [Biba 1977].

### 4.3.1 Bell Lapadula Security Policy Model

The Bell-LaPadula model [Bell-LaPadula] is an extension of the access matrix model [Castano] that models a hierarchical entity classification model typically found within governments and the military. A security level consists of:

4. *a hierarchical classification:*
   each entity is a member of the set [Top Secret (TS), Secret (S), Confidential (C), Unclassified (U)], where TS > S > C > U;
5. *a non-hierarchical set of categories and caveats:*
   each entity is assigned specific categories and/or caveats to which the entity pertains or where it can be used (e.g., NATO, Nuclear); these determine whether a particular user can access the entity.

A security level $L_1$ is said to dominate security level $L_2$ if and only if:

- the hierarchical classification of $L_1$ is greater than or equal to that of $L_2$; and
- the non-hierarchical categories of $L_1$ include all those of $L_2$ as a subset.

Entries in the access matrix should satisfy the following properties:

- **Simple security property** (also called read-down property)
  A subject may have read access to an object only if the security level of the subject dominates that of the object; and
- **Star property** (also called write-up or confinement property)
  A subject may have write access to an object only if the security level of the subject is dominated by that of the object.

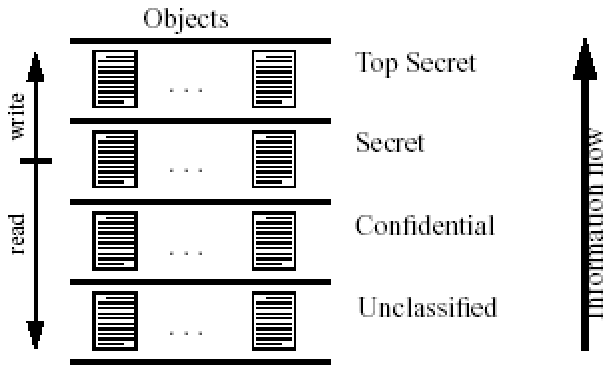

*Figure 2: Simple Security and *-Properties for Classifications*

The simple security and the *-properties ensure that information cannot flow from a given security level to a lower one (Figure 2). The model also introduces trusted subjects to whom the star property does not apply. This allows declassification of documents by authorized subjects.

The Bell-LaPadula model focuses on control of disclosure that is of particular interest to classified data as utilized by governments. In a commercial environment, preventing disclosure is important but preventing unauthorized modification is paramount [Clark-Wilson].

### 4.3.2 Biba Security Policy Model

The Biba model [Biba] takes a similar approach to the Bell-LaPadula model. It associates an integrity level with each entity. The integrity level consists of a *set of categories* and of a hierarchical *integrity classification*. The classification can take one of the three following values: Critical (C), Very Important (VI) and Important (I), with C > VI > I.

An integrity level $L_1=(C_1,S_1)$ is said to dominate integrity level $L_2=(C_2,S_2)$ if:

1. the hierarchical classification of $L_1$ is greater than or equal to that of $L_2$: $C_1 \gtrsim C_2$; and
2. the non-hierarchical categories of $L_1$ include all those of $L_2$ as a subset: $S_1 \supseteq S_2$.

The two following rules have to be verified for an action to be authorized:

- **Simple integrity property** (read-up)

  A subject may have read access to an object only if the integrity level of the subject is dominated by that of the object; and

- **Integrity * property** (write-down)

  A subject may have write access to an object only if the integrity level of the subject dominates that of the object.

These two conditions are analogous to the ones expressed in the Bell-LaPadula model. This is illustrated in the example of Figure 3, where Alice has the integrity level classification Very Important (we will not consider the categories in this example). In accordance with the simple integrity property she cannot read objects with Important classification. On the other hand she cannot write Critical objects, because of the integrity star property.

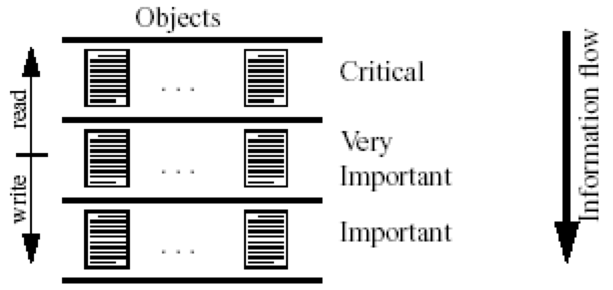

*Figure 3: Simple Integrity and *-Properties for Classifications*

The two integrity properties are easy to understand if we consider that the integrity level is a measure of the trust placed in the information. If Alice is trusted as *Very Important* and she reads information from an *Important* file that would mean she possesses information that is less trustworthy than her actual integrity level. And if Alice writes in a *Critical* object, there would be information trusted as only *Very Important* in the *Critical* object.

## 4.4 Non-Discretionary Policies

In 1993 Abrams [Abrams 1993] identified a third category, Non-Discretionary Access Control (NDAC) Policies. NDACs identify situations in which authority is vested in some individuals but there are explicit controls on the delegation and propagation of that authority. These controls model the method of delegated responsibility typically found in hierarchical organizations ensuring that responsibility for particular actions are delegated down to trustworthy individuals.

## 4.5 Administrative Policies

Administrative policies [Sandhu 1994 and Samarati] determine who is authorized to modify allowed access rights; these exist only within discretionary policies. Mandatory policies determine access rights entirely on the basis of the security classification of their subjects and objects. Administrative policies can be divided into the following categories:

1. ***Centralized,*** where administrators are allowed to grant and revoke authorizations to users;
2. ***Hierarchical,*** where a central super user is responsible for assigning administrative responsibilities to other administrators; the administrators can then grant and revoke access authorizations to the users of the system according to an organization chart;
3. ***Cooperative,*** where authorizations to a resource are granted by two or more authorized users acting in concert (i.e., two-man rule);
4. ***Ownership,*** where a user is considered the owner of the entities he creates; the owner can grant and revoke privileges for other users to those entities; and
5. ***Decentralized,*** where the owner or administrator of an entity can also grant other users the privilege of administering authorizations on the entity.

Over the years other sophisticated security models have been proposed to formalize security policies required for commercial applications. The Clark-Wilson model [Clark-Wilson] is a well-known one for commercial data processing practices. Its main goal is to ensure the integrity of an organization's accounting system and to improve its robustness against insider fraud. The Clark-Wilson model recommends the enforcement of two main principles: the principle of well-formed transactions, where data manipulation can occur only in constrained ways that preserve and ensure the integrity of data; and the principle of separation of duty. The latter reduces the possibility of fraud or damaging errors by partitioning the tasks and associated privileges so cooperation of multiple users is required to complete sensitive tasks. Authorized users are assigned privileges that do not lead to execution of conflicting tasks.

## 4.6 Brewer-Nash Chinese Wall

[Anderson 1996] defined a security policy model that specified clear and concise access rules for clinical information systems. This model is based on access control lists and the authors claim it can express lattice-based policy models. In 1989 [Brewer-Nash] defined the Chinese-wall policy as a formal model of a security policy to prevent information flows that cause conflict of interest for individual consultants, originally within a financial model. The basis of the model is that people are only allowed to access information that is not in conflict with other information that they already possess. It, too, can be modelled utilizing lattice-based access control models.

The Chinese Wall is a policy defined to describe the functioning of the financial and consulting businesses. Financial analysts and consultants have access to private corporate information. The model describes a way to ensure that this information will not be used by a consultant who advises a competitor.

In the Chinese Wall Model objects are grouped into company datasets. Datasets corresponding to competing companies are grouped into conflict of interest classes (see Figure 4).

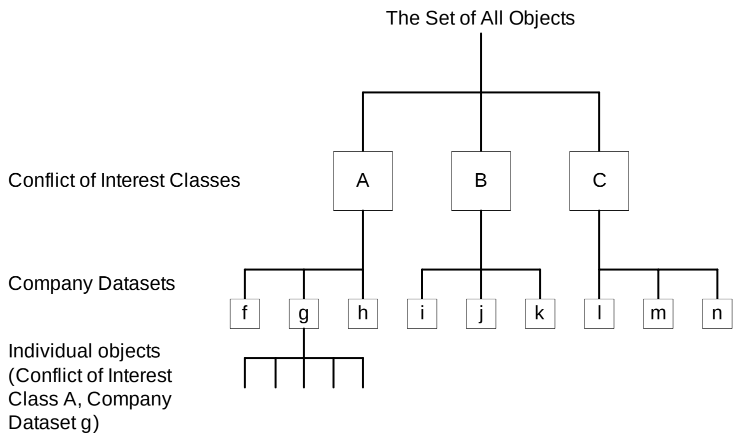
The Set of All Objects
Conflict of Interest Classes
A
B
C
Company Datasets
f
g
h
i
j
k
l
m
n
Individual objects (Conflict of Interest Class A, Company Dataset g)

*Figure 4: Chinese Wall (adapted from [Brewer-Nash])*

Brewer and Nash specified the following rules as the basis of the Chinese Wall security policy model:

- **Read rule**
  A subject S can read an object O only if:
  - S has already read an object of the dataset to which O belongs or
  - O belongs to a conflict of interest class within which S has not read any object
- **Write rule**
  A subject S can write an object O only if:
  - the read rule is satisfied and
  - no object can be read which is in a different company dataset than O.

## 4.7 ROLE-BASED ACCESS CONTROLS (RBAC)

Role-based Access Control (RBAC) [Ferraiolo, Sandhu 1996] has its root in the simple observation that in many organizations, authorizations are based on employee functions. With RBAC, system administrators create roles according to the job functions performed in a company or organization, grant permission (access authorization) to those roles, and then assign users (owner of subjects) to the roles on the basis of their specific job responsibilities and qualifications, as shown in Figure 5.

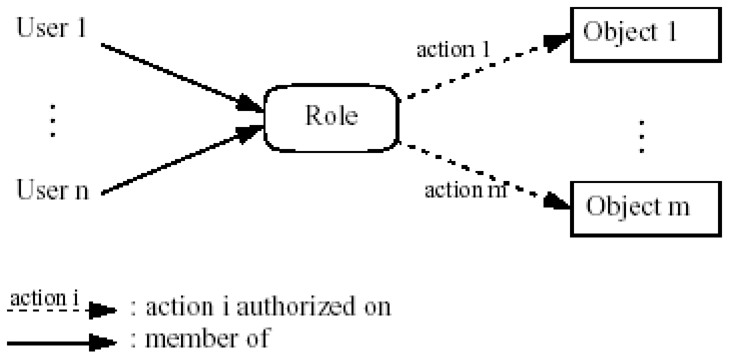

*Figure 5: Role Relationships*

Before being given the permission corresponding to a role, a user would have to activate it by some means, say login. Role activation ensures that the user will have no more privileges than necessary to perform a job, namely ensuring the principle of least privilege.

Roles can be nested, effectively organizing them into hierarchies. By utilizing rules of inheritance a sub-role could immediately inherit additional roles. This simple addition greatly simplifies the maintenance of role-based systems. The example illustrated in Figure 6: Example of a Role Hierarchy shows a user who is a member of the role "Physician" and as such is also allowed to activate the "Health-care provider" role.

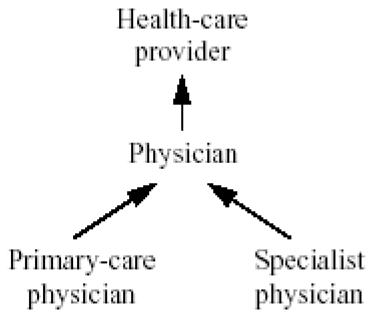

*Figure 6: Example of a Role Hierarchy*

One of the fundamental differences between DAC and RBAC is that users in RBAC cannot pass access permission on to other users at their discretion. The principal concern of RBAC is protecting the integrity of information. Roles permit the grouping of a set of permissions related to a position in an organization such as finance director, network operator, ward-nurse or physician. This allows permissions to be defined in terms of the position rather than the person assigned to the permission, so that policies do not have to be changed when people are reassigned to different positions within the organization. Another motivation for RBAC has been to reuse role specification by a form of inheritance whereby one role (often a superior in the organization) can inherit the rights of another role and thus avoid the need to repeat the specification of permissions.

RBAC contains users, roles, permissions and sessions. Permissions are attached to roles and users can be assigned to roles to assume those permissions. A user can establish a session to activate a subset of the roles to which the user is assigned. Implementations of RBAC introduce role hierarchies [Sandhu 1998] in order to structure the roles to reflect an organization's lines of authority and responsibility. Further extensions include the introduction of constraints to restrict the assignment of users or permissions to roles, or the activation of roles in sessions. Constraints are used to specify application-dependent conditions, and to satisfy well-defined control principles such as those of least-privilege and separation of duties. Sandhu et al [Sandhu 2000] proposed an updated set of RBAC models in an effort to formalise RBAC called flat RBAC, hierarchical RBAC, constrained RBAC and symmetrical RBAC. It is unknown whether any implementations of the updated model have been implemented. However, several efforts have advanced the codification of calculi to model RBAC: in particular, [Ahn-Sandhu 1999], [Ahn 2000], and [Braghin 2004].

For those wishing to read further on RBAC, Sandhu et al. [Sandhu 1996] specified four conceptual models in an effort to standardise RBAC. A number of variations of RBAC models have been developed, and several proposals have been presented to extend the model with the notion of relationships between the roles [Barkley 1999], with recall of temporal state information [Mossakowski 2003], and with the idea of a team, to allow for team-based access control (TMAC) where a set of related roles belonging to a team are activated simultaneously [Thomas 1997]. C-TMAC [Georgiadis 2001] augments TMAC to allow for greater incorporation of context (environment) variables in access control decisions. A more in-depth discussion is beyond the scope of this work.

## 4.8 Distributed Access Control

Distributed computing presents its own challenges regarding access control: scalability, revocation, and interoperability. This section discusses approaches undertaken to date to address them.

### 4.8.1 Distributed Access Control

Identity-based Access Control is a formalization of Access Control Lists (ACLs). An ACL is a list, per-resource, specifying each permitted user of the resource and the set of actions the user may perform on it. Under this approach, the server checks the client’s authenticated identity against the pertinent ACL to determine whether the user may perform a requested action on a resource.

ACL implementations can easily become complicated. Each resource (e.g., file) and each resource container (e.g., directory) can have its own ACL governing resource creation, several forms of use, and destruction. Typically, a central administrator maintains ACLs, since the model does not support the delegation of administrative privilege. Hence, the approach does not scale well as systems increase in complexity.

The model was originally designed for access control decisions in closed systems. It assumes that a server knows the identities of all of its users. As such, the method does not scale well to open, distributed systems.

Yao [Yao 2003] summarizes the limitations of IBAC. This approach:

- does not scale well to larger systems;
- does not support delegation;
- does not allow for efficient revocation;
- does not incorporate auditing;
- governs only access control; and
- does not incorporate obligations on positive or negative access control decisions.

### 4.8.2 Capability-based Access Control

Capability-based Access Control addresses some of the shortcomings of Identity-based Access Control.

[Yao 2003] outlines the approach. When the system creates an object, it generates a random secret to associate with it. To construct a capability, it computes a hash of the object identifier, access rights, and the secret. It then embeds the hash into the digital capability as its check digits; i.e., for hash function *f*,

$$hash = f(secret, protected\ fields)$$

$$capability = (protected\ fields, hash).$$

The system distributes such capabilities to authorized users. When a user presents a capability for access to a resource, the system checks if the capability is genuine by recomputing the hash. If the hash is correct, the system makes an access control decision based on the capability.

Although the model supports delegation (identifying delegation itself as a capability), the approach still has several drawbacks. In particular, it:

- does not protect against replay attacks using stolen capabilities;
- does not govern the propagation of capabilities; and
- does not allow for efficient revocation.

### 4.8.3 Capability-based Access Control

Credential-based Access Control arose in recognition of the problems in applying capabilities to distributed systems. Credentials are a more elaborate form of data structure, distributed to and handled by individual users. Where capabilities encoded only permissions per-resource, credentials store per-user information such as user identifier, to guard against stolen credentials. Some credentials may only have meaning to some servers or to some applications; there may be many issuers of credentials per system. In this model, we see a motion to accommodate the needs of federation. [Yao 2003] summarizes the evolution of this approach; the following subsections elaborate.

#### 4.8.3.1 Early Efforts

Early efforts adopted more sophisticated delegation models and built towards federated authorization. In particular,

- Li's ICAP [Li 1989];
- Bull et al's Open Distributed Processing (ODP) [Bull 1992]; and
- Neuman's Restricted Proxy [Neuman 1993].

#### 4.8.3.2 Open Architecture For Secure Interoperating Services (OASIS)

More recently, efforts at the University of Cambridge's Computer Laboratory [Bacon 2000] have produced OASIS (Open Architecture for Secure Interoperating Services), adopting principal-specific capability and parameterized roles within an RBAC framework [Hayton].

In OASIS, credentials appear as certificates of one of three kinds:

- role membership certificates (RMC), for short-lived role assignments;
- appointment certificates (AC), persistently attesting to role assignment or delegation capacity; and
- revocation certificates (RVC), revoking instances of appointment.

OASIS views the system as a collection of services, both (certificate-issuing) OASIS and (policy-enforcing) OASIS-aware. OASIS admits policies for:

- role activation;
- assessment of appointment certificate validity; and
- service use/method invocation.

The OASIS framework allows each OASIS service to define access rights per user category. A service can refer to its own categories or those of other services. OASIS provides a Horn clause-based role definition language (RDL) to specify conditions for clients to enter each role. Policies can include tertiary computations, but OASIS does not provide for the definition of consequences arising from failed access requests. Neither does OASIS allow for context-sensitive policies.

Finally, OASIS certificates and policies pertain to services, not to resources in particular. As such OASIS does not represent a likely candidate on which to build a DRDC policy engine.

#### 4.8.3.3 Secure Environments For Collaboration Among Ubiquitous Roaming Entities (SECURE)

SECURE is a European Union Information Society Technologies (EU-IST) project adding environmental predicates to OASIS's ability to call external services. SECURE defines interfaces allowing OASIS policy rules to include trust and cost/benefit computations [Dimmock 2004].

Fundamental to SECURE is the consideration of tertiary computation. No full implementations currently exist in any language. However, tracking SECURE may provide ideas useful to the progress of a DRDC policy engine.

## 4.9 Attribute-based Access Control (ABAC)

ABAC provides a means for local authorities to base access control decisions on authenticated attributes of subjects. [Yuan 2005], [Li 2002]. ABAC associates attributes with:

- the Subject (user, application, process);
- the Resource (Web service, system function, data); and
- the Environment (situation or context).

To begin, ABAC policy formulation sets subjects, resources and environments and predefined attributes for all of them. Then, the operation represents assignment relations of attributes to subjects, resources, and environments. Policy Rules then decide whether a given subject can access a given resource in a given environment. The access control decision process is the evaluation of applicable policy rules in the policy store.

The ABAC approach allows for multivariate access controls, with the policy set semantics accommodating the combinatorial complexity in logic, rather than a multiplicity in roles. Complete ABAC systems allow for:

- ***decentralized attributes –*** an entity may assert that another entity has an attribute;
- ***delegation of attribute authority –*** an entity may delegate authority over an attribute;
- ***inference of attributes –*** an entity may use one attribute to make inferences about another;
- ***attribute fields –*** an attribute may carry fields such as age or credit limit; and
- ***attribute-based delegation of authority –*** avoiding the need to know every delegated authority individually.

Li et al [Li 2002] use ABAC to motivate the formulation of the *RT* framework, a family of logic-based Role-based Trust-management languages to represent policies an credentials in distributed authorization.

ABAC is highly granular in its approach access control, but maintenance of attribute-based policies can become expensive. Its multidimensional approach to attribute specification may prove useful to DRDC in caveat assignment and validation.

### 4.9.1 U.S. Navy’s EDAC

The U.S. Navy COMPACFLT's Enterprise Dynamic Access Control (EDAC) project [Fernandez 2006] builds on [ANSI 359], Security Assertion Markup Language (SAML) v2, and eXtensible Access Control Markup Language (XACML) v2. EDAC v2 was published in early 2006. EDAC is an implementation of Attribute-based Access Control.

EDAC's role and permission assignment technology is capable of evaluating resource access based on:

- attributes;
- environmental context;
- business rules;
- questionnaires; and
- workflow.

EDAC anticipates dynamic, real-time corporate and user profile attribute changes, and can accommodate resource access policies sensitive to tertiary computations such as (U.S.) Homeland Security Advisory threat levels.

The system bases access control decisions not on user identity as such, but on the current values of selected user attributes, as defined by policy. As such, EDAC is an Attribute-based Access Control (ABAC) system, although the documentation labels such as a Metadata-based Access Control system (MBAC).

EDAC provides the following:

- comprehensive access control features;
- web service modularity offering standard interchangeable access control components from various vendors;
- automated assignment of users to proper roles;
- separation of duties avoiding conflict of interest;
- hierarchical role and resource specification; and
- a role engineering mechanism.

Throughout EDAC, the text recasts some terminology in a manner that strays from typical practice, as illustrated by the following:

- Object – “an object is a person or thing seeking resource access” (i.e., the entity generally known as the subject as opposed to the resource itself);

- Rules Engine Service (RES) – "evaluates object and conditions to determine resource access" (comprising functionality generally attributed to a PDP and a PEP);
- Resource Manager (RM) – never properly specified, but appears to correspond to the operator of a Policy Authorization Point (PAP);
- Condition Manager Service – an interface for the Resource Manager (effectively a PAP); and
- Object Profile Manager Service – with the Structure Format Service, performs as a Policy Enforcement Point (PEP).

While within the scope of the EDAC effort, the remainder of this section employs the EDAC semantics of these terms. Table 1 introduces EDAC entities and their function within the implementation. Figure 7 illustrates the setup and function of the system. The discussion then elaborates.

*Table 1: EDAC Entities*

| EDAC Entity | Legend (Figure 7) | EDAC Function |
|---|---|---|
| Administrative Service<br><br>Repository Service | A | Establishes resource containers, Client Meta Database (CMD) referrals, Resource Manager (RM) accounts<br>Stores resource access conditions, RM accounts, CMD connection parameters |
| Condition Manager Service<br>Customer Meta-Database | B | Establishes and edits conditions to access resources (i.e., sets access control policy)<br>Contains structured user metadata used for access control policy |
| Condition Status Service | C | Listens for CMD content changes and flags unmatched or unreachable conditions |
| Customer Environmental Interface | D | Monitors environmental stati |
| Customer Object Profile Manager Service<br>Customer Personnel Database<br>Structure Format Service | E | Object characteristic compilation, selection, formatting<br><br>Represents a human resource database<br><br>Converts profile and environmental status inputs to DN format using CMD |
| Rules Engine Service<br>Customer Portal<br>Customer Resources | F | Evaluates objects and conditions to determine object resource access<br>Interfaces with EDAC to list accessible resources<br>Represents Web services, software applications, etc. |

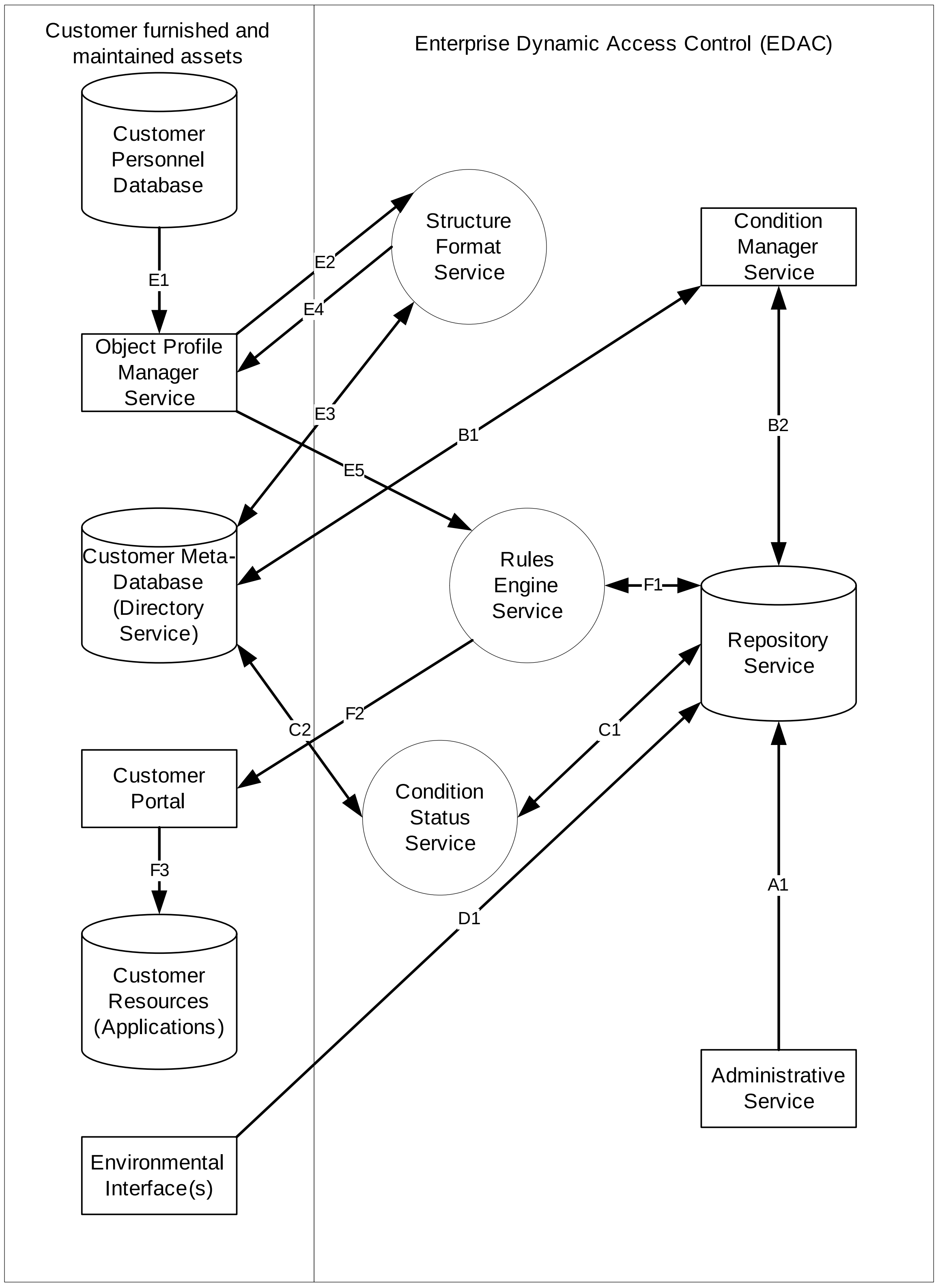
Customer furnished and maintained assets
Enterprise Dynamic Access Control (EDAC)
Customer Personnel Database
Object Profile Manager Service
Customer Meta-Database (Directory Service)
Customer Portal
Customer Resources (Applications)
Environmental Interface(s)
Structure Format Service
Condition Manager Service
Rules Engine Service
Repository Service
Condition Status Service
Administrative Service
E1
E2
E3
E4
E5
B1
B2
F1
F2
F3
C1
C2
D1
A1

*Figure 7: EDAC Operation (adapted from [Fernandez 2006])*

The operation of the system proceeds as follows:

- During setup:
    - the Administrative Service establishes resource containers, CMD referrals, RM accounts, etc.; and
    - the Condition Manager Service establishes and edits conditions to satisfy to access resources.
- At all times:
    - the Condition Status Service listens for CMD content changes and flags unmatched or unreachable conditions as they occur; and
    - the Customer Environmental Interface furnishes environmental updates as they occur.
- When a client requests access to a resource:
    - the Customer's Object Profile Manager Service intercepts the request and contacts the Structure Format Service, to formulate a query to the Customer Meta-Database and so to pose a properly-formatted access request to the Rules Engine Service for resolution; and
    - the Rules Engine Service validates the request against policies in the Repository Service and, if deemed legitimate, contacts the Customer Portal to access the requested Customer Resource.

The system uses standards-based tokens and message protocols (e.g., XACML, SAML, and the Lightweight Directory Access Protocol (LDAP)).

The identification of a Role with a set of Attribute and Environmental values presents a danger of Role explosion, which could affect the scalability of the system. The system requires perfect network connectivity, in particular for the Condition Status Service and for the Rules Engine service to perform their duties.

Note that EDAC is U.S. Government Proprietary and may be related to one or more U.S. Government-owned inventions.

## 4.10 Governance-based Access Control (GBAC)

GBAC [CGI 2005] is an emerging model, based on governance, to meet the following assumptions:

- many organizations may require access to information;
- information may be accessed by, or shared with, external users; and
- everyone may be subject to compliance with multiple authorities and jurisdictions.

GBAC intends to allow organizations to address the requirements of sharing information externally, while embracing and enhancing traditional access control models such as RBAC.

GBAC posits that when an organization collects, uses, manages, or shares sensitive information, the relevant legislation to which the organization is accountable must govern access to the resulting asset throughout its lifespan, in transit or in stasis. GBAC access control policy rests on the following data:

- jurisdiction;
- collection authority;
- collection purpose;
- security designation;
- disclosure authority; and
- disposition authority.

GBAC's strength is in its maintenance of associations of each information asset against the above metadata throughout its lifespan. GBAC is primarily geared to compliance with the letter of the law (e.g., U.S. SOX 404, Canadian Bill 198) with respect to sharing clients' sensitive personal data. The GBAC model is still very underspecified. No implementations appear in the field.

## 4.11 Risk-Adaptive Access Control (RAdAC)

IBM and the U.S. National Security Agency (NSA) developed the RAdAC approach [McLaughlin 2004 and McGraw 2004] to model assured information sharing for the GIG.

Traditional access control approaches cannot satisfy a need-to-share paradigm as they are inflexible to situational considerations, to differences in people and IT components, and to history. RAdAC represents a shift to a flexibly, policy-based access control model supporting need-to-know and need-to-share [Choudhary 2005]. The RAdAC approach models access control based on:

- security risk;
- operational necessity; and
- policy for the balance between the two, for various situations.

As such, operational necessity can trump security risk. Access Control Policy codifies security risk by people, IT components, content, environment, and history. Risk measurement will proceed algorithmically from a set of inputs.

Figure 8 and Figure 9 demonstrate the RAdAC model shift from need-to-know to need-to-share.

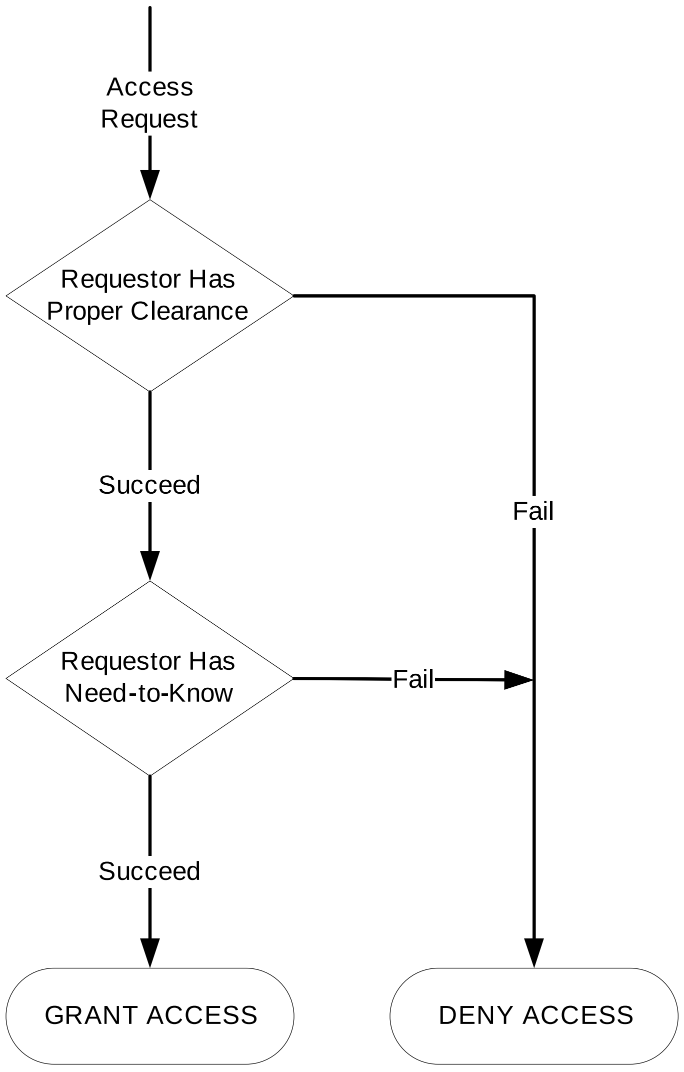

*Figure 8: Need To Know (adapted from [McGraw 2004])*

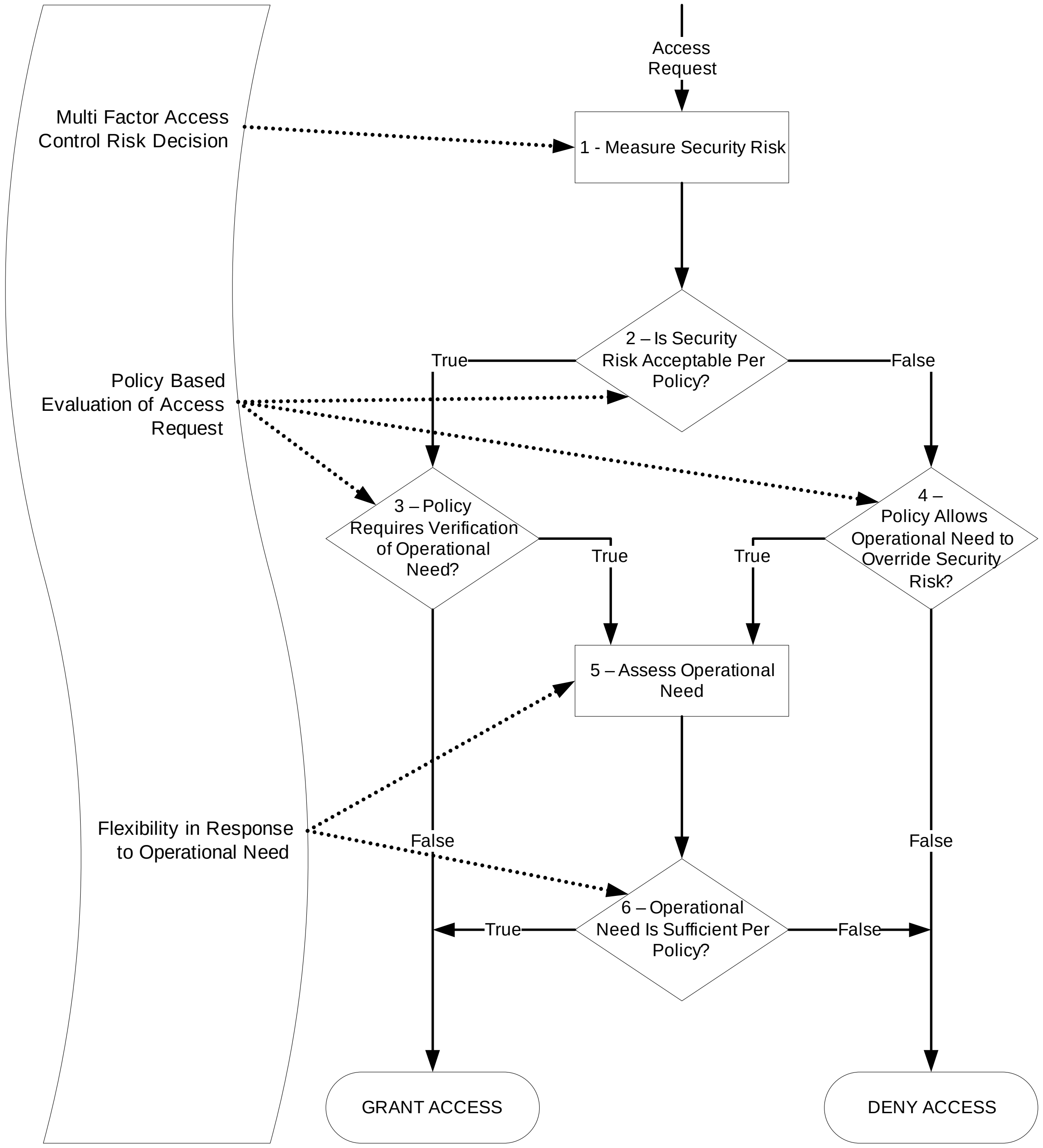
Multi Factor Access Control Risk Decision
Policy Based Evaluation of Access Request
Flexibility in Response to Operational Need
Access Request
1 - Measure Security Risk
2 – Is Security Risk Acceptable Per Policy?
True
False
3 – Policy Requires Verification of Operational Need?
4 – Policy Allows Operational Need to Override Security Risk?
True
True
5 – Assess Operational Need
False
False
6 – Operational Need Is Sufficient Per Policy?
True
False
GRANT ACCESS
DENY ACCESS

*Figure 9: Need To Share (adapted from [McGraw 2004])*

The RAdAC model calls for:

- the identification and association of all GIG assets with set of privileges;

- the metadata labelling of all GIG assets to identify access and protection requirements; and
- the GIG to make nuanced authorization decisions based on several factors.

The model adopts a familiar architecture as per Figure 10 to support policy-based protection.

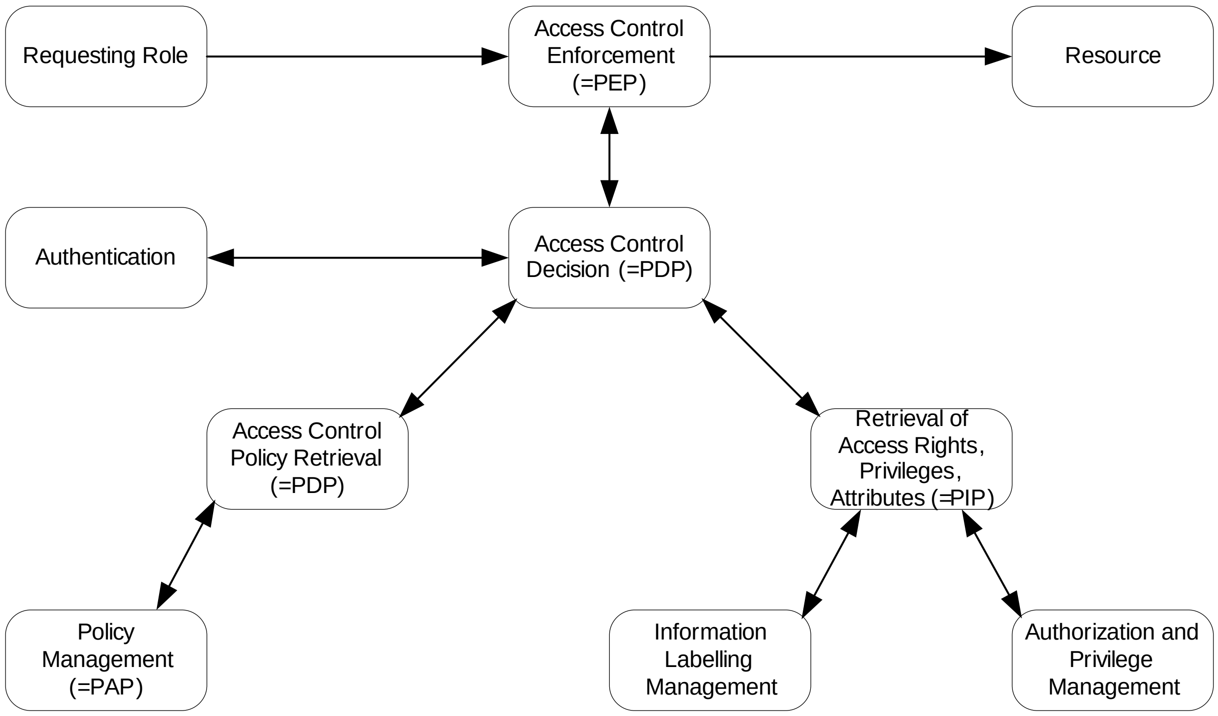
Requesting Role
Access Control Enforcement (=PEP)
Resource
Authentication
Access Control Decision (=PDP)
Access Control Policy Retrieval (=PDP)
Retrieval of Access Rights, Privileges, Attributes (=PIP)
Policy Management (=PAP)
Information Labelling Management
Authorization and Privilege Management

*Figure 10: RAdAC and Policy-based Protection (adapted from [McLaughlin 2004])*

To continue to develop towards implementation, RAdAC requires cross-domain solutions to evolve to support:

- enhanced content filtering;
- access control by privilege and label; and
- interconnection of enterprise services.

In stasis and in transit, the model calls for cryptographic protection of resources based on their protection requirements. Clients must identify and authenticate themselves with appropriate robustness, since object-level access control will become the primary protection mechanism for segregating information at different sensitivity levels.

This model is still early in development. Its implementation will require further development of several access control concepts, including but not limited to:

- real-time risk assessment of access control decisions;
- quantifying trust in people;

- determining operational need; and
- heuristics for access control decisions.

## 4.12 Network Policies

Network policies, as epitomized by RFC 2748 from the Internet Engineering Task Force (IETF), focus on the deployment of network level policies near the physical layer of a network stack. As such these policies are heavily tied to network level syntax and often utilize Internet Protocol (IP) v4 or IPv6 formats to define protected entities, typically IP addresses. Although these policy engines use a client-server architecture and utilize a mediation enforcement model wherein the mediation is handled centrally and the enforcement is handled at the end-points in a distributed manner, that is where the similarity ends. Network policy engines define and control access strictly on inbound and outbound messaging pathways.

Firewall and intrusion detection system (IDS) companies heavily utilize network policies and it is instructive to examine the future evolution of these two security technologies. Firewalls represent a mature, well understood security technology incapable of understanding complex context-oriented requests. Rules are typically enforced at the IP or port-level and rely on limiting access to specific addresses. IDSs have slowly begun to morph into context-based systems as they move away from the current bandwidth- and processor-heavy packet-based detection systems towards systems that comprehend actual access requests. The difference in bandwidth requirements is substantial. Where a simple “Open File XYZZY” may require a handful of system calls that are easily understood, the equivalent packet-level request may be hundreds of thousands of bits long. To address this problem IDS vendors are moving towards a context-sensitive model. This model requires IDS agents to be embedded into a variety of systems, with the advantage that resource access requests are monitored instead of IP traffic.

Network policies are also heavily utilized to manage Quality of Service (QoS). QoS requires the insertion of context into a packet to permit routers to determine how they should handle particular packets. These QoS markers are utilized to ensure voice receives priority on a network while Hypertext Transfer Protocol (HTTP) or similar requests can be delayed. Similarly, it is possible to examine destinations and determine whether the packet is destined for an FTP server or a VoIP phone. Network policies that define packet-level controls are adequate to define QoS.

Network policies as currently defined, heavily dependent upon simple IP addresses and rules, are unable to address the concerns of authorization and PMI requirements. Complex rules such as “Joan is only allowed access to the Payroll System from 0900 to 1700, Monday to Friday” are implementable by network policy rules. But rules such as “Joan must be authenticated to at least SECRET in order to access SECRET documents” are not.

It is highly likely that in the future network policy engines and authorization policy engines will move closer together. The simple fact that any reasonably complex policy engine can implement network policy rules is indicative that such a merger will in time occur. At present it would be easier to extend an access control language such as XACML to handle network policies than it would be to extend RFC 2748, or Simple Network Management Protocol (SNMP) management information bases (MIBs) to handle complex authorization and PMI requirements. It is likely that the two areas will converge, especially for firewalls, VPNs, and IDSs. However, the convergence is a longer term process. It should not, however, preclude developers of authorization policy engines from examining the network policy engine research to determine how best to resolve similar problems.

Since the 2003 survey, refinements and extensions to the IETF/Distributed Management Task Force (DMTF) Policy Core have followed, but the principal aim of such efforts has been to specify bindings for use with LDAP servers and SNMP MIBs [RFC 3460, RFC 3703, RFC 4011, and RFC 4104]. Currently, there is no standard means of expressing IETF/DMTF Policy Core items in eXtensible markup language (XML), although it would appear to be a straightforward exercise.

## 4.13 Policy-Oriented Logic

This section briefly describes some of the better-known logic-based specification languages categorized according to the type of logic used. Although logic-based languages are an attractive means by which to specify security policies due to their well-defined formalisms and suitability to analysis they are usually thought to be difficult to use and similarly difficult to translate into efficient implementations. The various forms of logic are included for completeness.

Efforts in policy-oriented logic since the 2003 survey have largely focused on codifying knowledge representation into XML, which itself continues to evolve on many fronts. Notably, the World Wide Web Consortium (W3C) has undertaken work on the Web Ontology Language (OWL) [Smith 2004]. OWL represents a merger of the DARPA Agent Markup Language (DAML) with European Union Information Society Technologies' (EU-IST) Ontology Inference Layer (OIL) for the Semantic Web and builds on the Resource Description Framework (RDF) to allow semantic markup depicting knowledge representations. In effect, OWL codifies Description Logic into XML, rebranding some key terms in the process. OWL already admits two profiles, OWL-Lite and OWL-DL (where DL refers to 'Description Logic'). OWL-Lite promises easier adoption; use of OWL-DL guarantees decidability in polynomial-bounded time.

### 4.13.1 First Order Logic

There are several examples that illustrate the application of first order logic to the specification of security policy. These include the logical notation introduced in [Chen]; the Role Definition Language (RDL) presented in [Hayton], and RSL99 [Ahn-Sandhu 1999]. All of these approaches are based on the RBAC model. In addition to these RBAC examples, there are some examples of the application of Z to defining security policies for a system. Z is a formal specification language that combines features of first order predicate logic with set theory [Spivey]. [Boswell] describes the use of Z to specify and validate the security model for the NATO Air Command and Control System. The aim of this work was to develop a model for both mandatory and discretionary access controls based on Bell-LaPadula.

One of the main problems encountered when using first order logic for policy specification arises when negation is used together with recursive rules. This leads to logic programs that can be neither decided nor be evaluated [Dantsin, Eiter et al. 1997]. Although it is possible to avoid the use of negation or recursion, this significantly diminishes the expressive power of the language.

### 4.13.2 Stratified Logic

Stratified logic permits a constrained use of recursion and negation while disallowing combinations that lead to undecidable programs. A stratified program is one where it is possible to order the clauses such that for any clause containing a negated literal in its body there is a clause later in the program that defines the negated literal. Another way of describing stratified theories makes use of directed dependency graphs.

The concepts of stratified theories and stratification were originally developed in the context of databases [Chandra] and were later adopted into the area of logic programming as described in [Apt] and [Van Gelder]. Programs that use stratified logic can use negation to extend their expressive power and can be evaluated in a flounder free manner. Indeed there are numerous studies that identify stratified logic as a class of first order logic that supports logic programs that are decidable [Jager, Dantsin]. Moreover, such programs are decidable in polynomial time [Jajodia 1997]. A more detailed analysis of the computational complexity and expressive power of stratified logic can be found in [Dantsin].

Jajodia et al's authorization specification language [Jajodia 1997] is an example of a stratified first order logic language for specifying access control policies. [Barker 2000] adopts a similar approach to express a range of access control policies using stratified clause-form logic, with emphasis on RBAC policies. [Barker-Rosenthal] revisit their earlier work to show how policies specified in stratified logic can be automatically translated into a subset of the structured query language (SQL) to protect a relational database from unauthorized read and update requests.

### 4.13.3 Deontic Logic

Deontic logic was developed starting in the 1950s by [Von Wright], [Castaneda] and [Alchourron] by extending modal logic with operators for permission, obligation and prohibition. Known as Standard Deontic Logic (SDL), traditionally it has been used in analysing the structure of normative law and normative reasoning in law. Because SDL provides a means of analysing and identifying ambiguities in sets of legal rules, there are many examples of the application of SDL to represent legislative documents [Sergot; Jones]. An excellent overview of the applications of SDL is in [Wieringa].

Some of the earliest work using deontic logic for security policy representation can be found in [Glasgow]. The focus of this work was to develop a means of specifying confidentiality policies together with conditional norms. Others who have utilized deontic logic include:

- [Cholvy-Cuppens], whose SDL is used to represent security policies with the aim of detecting conflicts in the policy specifications; [Cuppens-Saurel] have extended this work to describe how delegation can be represented using deontic logic notation; and
- [Ortalo], who describes a language to express security policies in information systems-based deontic logic whereby he accepts the axiom $Pp = \neg O \neg p$ ("permitted p is equivalent to not p being not obliged") as a suitable definition of permission.

An inherent problem with the deontic logic approach is the existence of a number of paradoxes. For example, Ross' paradox can be stated as $Op \Rightarrow O(p \vee q)$, i.e. if the system is obliged to perform the action send message, then it is obliged to perform the action send message or the action delete message. Although [Prakken] offers resolutions to these paradoxes the existence of paradoxes can make it confusing to discuss policy specifications using deontic logic notations.

### 4.13.4 Temporal Logic

Temporal logic represents and reasons about propositions qualified in terms of time. Any logic that treats time as a sequence of ordered states is a temporal logic. Temporal logic dates back to Aristotle. More recently, scientists and logicians (notably, Amir Pneuli) have introduced temporal logic into computer science. Mossakowski et al [Mossakowski 2003] introduce temporal logic to dynamic RBAC to model a dynamic separation of duties that maintains execution history.

### 4.13.5 Trust

Networked applications require connectivity between entities that often do not know each other. In such situations the traditional assumptions for establishing and enforcing access control do not hold: subjects of requests can be remote, previously unknown users, making the separation between authentication and access control difficult. A possible solution to this problem is the use of digital certificates or credentials representing statements certified by trusted entities, which can be used to establish properties of their holder (e.g., identity, accreditation). Access control makes the decision whether to grant a party access based on properties that the party may have, and can prove by presenting one or more certificates. However, it is crucial that the credentials being so used are based on a sufficient level of trust as to ensure a valid mediation. Grandison and Sloman [Grandison 2000] have examined the issues surrounding trust management frameworks in order to combine authentication with authorization in such a way as to create viable security solutions useful in applications such as e-commerce, e-mail, and web-based labelling. Yao [Yao 2003] interprets trust management axioms into a framework of logic and subsequently as XML.

# 5. High-Level Description of a Policy Engine

## 5.1 Overview

Authorization of access to resources is typically viewed as being the responsibility of the applications that access specific resources. This leads to incoherent implementation of security within an organization as the protection of a resource becomes dependent upon the path through which it is accessed. This paper proposes that authorization be provided as a service that is utilized by applications that require security. An architecture is presented that supports fine grained access control wherein policies are associated with individual resources. The requirements for, and properties of, policies are described and several examples are provided.

The central component of a policy-based authorization service is a policy engine. This section addresses the following aspects of a policy engine:

- deficiencies with the current approach;
- policy engine overview;
- security essentials;
- policy analysis;
- policy refinement;
- entities;
- policy language; and
- centralized policy-based authorization.

Later sections examine the literature, determine which of the policy-based authorization systems discussed have actually been implemented, and evaluate which can be considered viable choices for policy-related research and development at DRDC.

## 5.2 Deficiencies with the Current Approach

Security currently and historically [Amoroso, Anderson 1972, Bell, Clark-Wilson, CTCPEC, Gasser, Gligor, CC] has been viewed as keeping malicious users out of critical computer systems. This view of restriction rather than sharing is becoming increasingly challenged as the desire for information sharing increases. In order to create such a system it is vital to view security in a radically different way, as a means of sharing information between authorized individuals [Bacic 1989, Bacic 1990a, Bacic 1990b, CTCPEC].

While the requirement to share information has been realized within applications, security has generally been provided by each element within the components making up the application. For example, web applications generally consist of web, application and database servers chained together; each server providing a point solution to the authorization problem. An application should not have to secure and protect a given resource; such a service should be provided independently of the application, much like an operating system provides services to access the file system, the user interface, the mouse, or the network.

The inclusion of security within applications becomes untenable with the introduction of distributed architectures, peer-to-peer computing and other networking initiatives where there is no guarantee of consistent platforms [Abrams 1993a, Abrams 1993b, Abrams 1990, Bacic 1990a, Hosmer 1991, Hosmer 1992]. Security at the application level has resulted in a plethora of security designs and solutions, few of which are interoperable [Amoroso, Anderson 1972, Bacic 1989, Bacic 1990a, Bacic 1990b, Bacic 2002]. Security within workflow systems, for example, requires that state be used to answer the authorization question. While state may be passed from application to application, this clearly requires a degree of interoperability that exceeds current standards. This lack of interoperability has resulted in well-documented security holes that have increased the reluctance of organization to adopt widespread resource sharing as a model either within or between enterprises.

## 5.3 Policy Engine Overview

There has been considerable recent interest in using programmable policies for specifying dynamic policies that can be easily modified to change the behaviour of a system without recoding the entire system. Most of the policy-based approaches use condition-action rules, either with or without event triggering, although some also use interpreted scripting languages. This survey describes the various approaches discovered and focuses on those that are the most promising. The following sections outline the various pieces that comprise a policy engine.

## 5.4 Security Essentials

With the emergence of the need for real-time collaboration [Bacic 2002], stopgap measures are no longer sufficient. Security officers and system administrators must now fulfill five security requirements:

- integration with existing products;
- uniform, ubiquitous security across the enterprise;
- controlled trust between Intranet and extranet systems;
- centralized and uniform controls; and,
- programmable policies that reflect the business rules.

With the emergence of distributed architectures for sharing the focus is now on sharing, not restricting. As advances in network computing continue, the requirements to share information and other resources among physically separate locales will continue. Creating walls of preventative access due to restrictive rules will no longer suffice. Further exacerbating the problem is the fact that when two or more computer systems are linked, their security policies often clash and overall security actually diminishes. Until recently the act of restriction was sufficient. Now, with the desire to share comes the need to create a new security paradigm - one that utilizes programmable security policies to determine access dynamically.

## 5.5 Policy Analysis

Conflicts between policies can arise due to a specification containing policies that have opposite modalities, with the same subjects and targets, but both permit and forbid the same actions. In this situation, an agent interpreting the policies will not be able to perform an action appropriately because one policy negates the effect of the other. For a policy-based system to work effectively, it is important to have a means of detecting and resolving any conflicts that arise. In this section, we discuss different types of conflicts and present strategies for resolving them.

A classification of policy conflicts is presented in [Lupu 1999], which discusses both modality conflicts and application specific conflicts. Modality conflicts can be categorised into three distinct types:

3. **Authorization conflicts,** which arise when a positive and a negative authorization policy is defined for the overlapping subjects, targets, and actions;
4. **Obligation conflicts,** which arise when one policy obliges a subject to perform a given action whilst at the same time another policy forbids the action from being performed (e.g., in the context of Ponder, this situation would arise if an obligation and a refrain policy were defined on overlapping subjects and targets with identical actions); and
5. **Unauthorised obligation conflicts,** which arise when a subject is obliged to perform an action that it does not have the authorization to do (e.g., in a system with a default negative authorization policy in which actions must be explicitly authorised, this could occur if an obligation policy is defined without an associated authorization policy).

Application specific conflicts are those that arise because of constraints defined for the particular application in which the policies are being used. For example, a system that enforces the principle of separation of duties would define a conflict if the same person who submits an expense report is also allowed to approve it.

[Jajodia 1997] identifies that conflicts can be either static or dynamic. The distinction is that analysing the syntax of a policy statement can identify static conflicts. These conflicts will occur irrespective of the state of the system enforcing the policies - this is often the case for simple modality conflicts. Dynamic conflicts are those that occur at run-time and arise because a particular state of the system results in a conflict. These are harder to detect in advance given that it is necessary to analyse the system in all possible states to do so.

Jajodia et al. propose that a conflict, once detected, could be handled in one of three ways. The most obvious and simplest one is for the system to declare an error condition whenever a conflict arises. However, this solution is not particularly interesting since it does not allow for the system to recover automatically from the conflicting scenario. Other solutions are to allow the positive policy to override, or to let the negative policy override. The latter strategy is adopting an approach of ‘do no harm’, based on the assumption that the negative policy (i.e., the one that prevents an action being performed) has a more benign effect on the system than its conflicting counterpart. As would be expected, the positive policy override strategy is the exact converse of the negative policy override approach.

In addition to the negative and positive override strategies mentioned above, [Lupu 1999] also identifies some alternatives. One approach suggested is to assign explicit priorities to every policy. This way, when a conflict arises, the agent enforcing the policy could simply compare the priority values and enforce the policy that has the highest priority. However, this approach could easily lead to inconsistent behaviour of the system if, as is common in distributed systems, multiple people are responsible for defining policies and assigning their priorities. Other strategies suggested include giving priority to the policy that is closest to the managed object; or using the specificity of the policy definition to determine its priority.

Work done by Chomicki and Lobo [Chomicki] describes how conflicts can arise between Event-Condition-Action[1] (ECA) rules and action constraints defined in the Policy Description Language (PDL). Here, a policy monitor is defined to detect conflicts between the ECA rules and any action constraints. In order to resolve the conflict, the monitor will either choose to ignore certain events, thus preventing the ECA rule from activating and causing the conflict; or will cancel any actions that are specified in an action constraint. The latter scenario is an example of a negative policy override strategy.

## 5.6 Policy Refinement

The need for policy refinement techniques has been apparent from the outset of research into policy-based systems management. [Moffett 1993] introduces the idea of policy hierarchies and the application of policy refinement to derive lower-level, more specific policies from high-level ones. The motivation to refine policies can be defined as follows:

- to determine the resources that are required to satisfy the needs of the policy;
- to translate high-level policies into operational policies that can be enforced by the system; and
- to allow analysis that would verify that the set of lower level policies actually meet the requirements of the high-level policy.

Moffett and Sloman [Moffett 1993] suggest goal refinement and arbitrary refinement of objectives as possible approaches to policy refinement. Goal refinement is a technique that has been further developed in the area of requirements engineering. Work done by Darimont et.al. [Van Lamsweerde 1995, Van Lamsweerde 1996, Van Lamsweerde 1999] present patterns of goal refinement that allow high-level goals to be stated in terms of a combination of lower level ones. In this work, by specifying goals in terms of temporal logic rules, it is demonstrated how to derive provable refinement patterns. The approach taken for proving a given pattern is to assume that each of the sub-goals holds and then show that it is possible to infer the truth of the base goal from the conjunction (or disjunction) of the sub-goals.

The goals are specified using a language called KAOS, which supports both a formal and informal definition of the system. The informal definition is specified in natural language while the formal definition uses the temporal logic notation introduced by [Manna]. It does not appear

[1] Event-Condition-Action rules are similar in format to most security rules following the on Event if Condition then perform Action format. It allows the system to sort on incoming event rather than triggering each rule and embedding the event within the rule set. There are obvious performance advantages to this structure.

that a public implementation of KAOS exists nor does it appear that KAOS has been commercialized by anyone at the present time.

The underlying logic technique used in goal refinement is called goal regression. Originally developed as a means of deriving plans of action for intelligent agents, goal regression provides a way of deriving the set of actions that, when applied to the system that is in some initial state S0, results in the system being in some goal state SG. This could be usefully applied in a policy refinement technique to derive a set of actions (or policies that use those actions) that have the equivalent effect on the system state as some other, high-level action.

As mentioned previously, the objective of policy refinement is to transform high-level policy specifications into more specific policies that would be better suited for use in different execution environments. Most of the above work is aimed at refining goals into implementation specifications but could be adapted to policy refinement. Before delving into the details of how policy refinement techniques work it would be useful to identify the desired properties of a refinement. In order to describe these properties a policy refinement is defined as follows:

**Definition (Policy Refinement):** If there exists a set of policies $P_{rs}$: $p_1$, $p_2$, .. $p_n$, such that the enforcement of a combination of these policies results in a system behaving in an identical manner to a system that is enforcing some base policy $P_b$, it can be said that $P_{rs}$ is a refinement of $P_b$. The set of policies $P_{rs}$: $p_1$, $p_2$, .. $p_n$ is referred to as the *refined policy set*.

Using this definition and drawing on work done to identify the properties of goal refinements [Darimont 1996] the following properties are proposed:

6. **Correctness:** a refinement is said to be correct if there exists a subset of the refined policy set such that the conjunction of all the members of that subset is also a refinement of the base policy;
7. **Consistency:** a refinement is said to be consistent if there are no conflicts between any of the policies in the refined policy set; and
8. **Minimality:** refinement is said to be minimal if it is correct and if removing any policy from the refined policy set causes the refinement to be incorrect.

A policy refinement can be said to complete iff all the properties defined above hold. The goal refinement approach also specifies a fourth property, non-trivial, that requires there to be more than one element in the refined set. However, in the policy refinement domain it may be acceptable to have a single policy that is a refinement of some base policy, provided that the refinement uses subjects, targets and actions that map to different physical entities.

It is also important to distinguish what is meant by a policy refinement pattern in contrast to a policy refinement as defined above. Once again it is possible to apply the concept presented in the goal refinement work of Darimont et al. [Darimont 1996], and define that a policy refinement pattern is a single level refinement that directly relates some base policy $P_b$ to a refined policy set $P_{rs}$ such that $P_{rs}$ is a complete refinement of $P_b$.

Other work on requirements refinement has concentrated on the mathematical models required and few have focused on actual physical implementations. Although it may be of interest to examine the works of Antón [Anton 1996, Anton 2000] in particular, especially his desire to attach goal specifications to Unified Modeling Language[2] (UML) design artefacts and use cases, such concerns are out of scope for this survey. Interested readers are directed towards the work of Antón as well as work by Brandozzi and Perry [Brandozzi] that maps KAOS specifications into an architecture model via their Architecture Prescription Language (APL), and work at Hewlett-Packard and its POWER policy wizard tool [Casassa] specifically designed to refine policies.

## 5.7 Entities

In order to define what transpires within a system we must utilize a generic term to define the objects upon which a security system will operate. We define an entity as the unified security object against which all security related functions are performed. Both the base security application and any ancillary security relevant applications utilize entities in order to manage and maintain security attributes. Table 2 summarizes the basic attributes that each entity possesses with respect to the actual resource being protected.

*Table 2: Entity Attributes*

| | |
|---|---|
| **Entity Identifier** | A reference to the entity (i.e., resource, user, group, application or policy) being protected. |
| **Authentication** | The means by which authentication will or has happened, with the possible inclusion of information pertaining to the login session, its state, locale, strength, and mechanism used. |
| **Security Policy (Authorization)** | The security policy or codified business rules to be applied to protect this entity. Additional information, such as access control lists, roles, etc. can be obtained from external 3rd party repositories such as LDAP, X.500, operating systems, web servers, etc. |
| **Access Control Violation Policy** | A policy that is invoked should the security policy be evaluated and deny the mediation request. This policy might notify an administrator of an unauthorized access or cause the requestor to be logged out of the realm. |
| **Audit (Accountability)** | A history of events describing what has transpired to this entity, by whom, temporal information, the policy invoked, and its logical outcome. |
| **Entity State** | Information specific to this entity that defines particularly important state information, such as who last invoked this entity and when, what the most recent and relevant condition of access has been, iterative/invocation limitations, etc. |

Thus, the entity provides the basic abstraction by which we can define security for a distributed, collaborative security system. It provides the handles to the policy and to the entity state, so that complex rules can be uniformly enforced across any distributed networked environment.

The entity is meant to represent all resources of interest being secured. This includes security and access violation policies. By making policies *first class entities*, security of the generic policy

[2] Information pertaining to UML can be found at http://www.uml.org/, the UML main site as well as vendor sites such as http://www.rational.com/uml/index.jsp.

engine can also be effectively provided. Management activity is also mediated through policy. This is something frequently overlooked in policy languages and policy-based systems.

## 5.8 Policy Language

The requirements for a policy language are that it be capable of reacting to, controlling and monitoring the system access event stream. Clearly, this goes beyond saying "yes or no" to a request to access a file. A policy language should not merely be passive, but should be capable of interacting with the system being secured to modify it; e.g., reconfiguring a web server if the policy determines that the server is under attack. Most security-related policy languages [Policy 2001] seem to restrict themselves to the control problem. We believe that this limits their utility.

Much of the work in computer security has delved into verifiable code and mathematical models of policy, often relying on variations of lambda calculus [Amoroso, Bell, Bell-LaPadula].

The primary requirements for a policy language are: verifiability, small size, efficient execution, expressive functionality, dynamicity and portability. The dynamicity requirement is important as it must be possible to update a policy in real time without taking the policy engine offline. The expressiveness of the language is also important – policies should be concise.

The ability of security policies being able to fetch pertinent information from external (to the policy) information stores such as LDAP, X.500, Certificate Authorities, operating systems, web servers, databases, etc., allows the policy to properly determine whether to grant access or not and what to do about the access event. Clearly, standards-based interfaces available through application programming interfaces (APIs) are required. The information retrieved need not be security related; it could be information on whether an individual is on vacation, what their physical location is, whether another individual is also logged on, and so on. In order to access certain entities it may be necessary to provide additional authentication and this, too, can be provided for in the policy via codified rules requesting additional authentication information.

### 5.8.1 Policy Mediation

Having defined an entity and policy, mediation need be no more complicated than evaluating:

*Mediate (activeEntity action passiveEntity env)*

where the *activeEntity* is the entity requesting access, as represented by the *action* upon the *passiveEntity*. It is the job of the mediation authority to determine whether such a relationship is allowed within the confines of the defined environment (*env*).

The importance of the mediation request is two-fold. First, it matches existing security policy modelling nomenclature regarding subject-object interaction and policy modelling. Second, it is can easily define iterative and reflective definitions. For example, if a user, George, wishes to access a particular application which in turn wishes to access an information resource, the agents would request two separate mediations:

*Mediate (George execute Application env)*

*Mediate (Application open Resource env).*

If state information is required in order to efficiently or more safely process any mediation request then it can be stored in the environment for later use. Using the above example, the environment would remember that it was George that initiated the application and that it was operating on his behalf. Therefore, if any access restrictions were placed on the resource as to which application and which user could access it, the information would be contained within the environment. Since mediations can be logically chained there is no restriction in the number or complexity of the mediation requests that can be handled.

### 5.8.2 Policy Language

Idyllic, the policy language of SecureRealms, was based upon Scheme (a LISP variant); a decision based upon its denotational semantics and small virtual machine. It is also familiar, being widely taught at university. Proof of the power of utilizing a symbolic language to define policies within the Policy Engine is exemplified by the example of a policy implementing NTFS security as shown below. The *nt-policy* function is the entry point to the policy that states:

- If you are the system user return *true*.
- If you are editing permissions (isMeta flag) and are the owner of the file, return *true*.
- Otherwise, if you or a group of which you are a member are denied access, return *false*.
- If you are not denied access, test whether you or a group of which you are a member is explicitly allowed access.

As can be seen, the policy is concise; the equivalent policy within the operating system is probably tens of thousands of lines of code. The natural language equivalent above is longer and more verbose than the code that defines the NT Security Policy (Figure 11) below[3].

[3] The policy was actually tested extensively with SecureRealms. During testing across a wide range of file system scenarios a potential hole in NT's security was observed. The scenario has to do with a file with _WXD permissions for User_1 in a directory with no access specifically for User_1. Without the above policy, User_1 could append to the file but since he did not have read permission the content of the file was overwritten. It was assumed that append required a read to determine where to append but since it could not read the file it just appended over the whole file. With the policy User_1 was denied access to the directory and therefore denied access to the file.

```
(define (nt-policy subj action obj isMeta)
   (if (isSystem? subj)
;  then
      #t
;  else
      (if isMeta
;     then
         (owner subj)
;     else
         (if (denied (list subj) action obj)
            #f
         (allowed (list subj) action obj)
)  )  )  )
```

*Figure 11: NT Security Policy*

## 5.9 Centralized Policy-based Authorization

Having described the abstractions necessary to implement coherent security, it is now possible to map the authentication, authorization and accountability requirements onto an implementation. Figure 12 contains elements of the disparate information systems that are deployed in enterprises today. These include file and web servers, along with peer-to-peer systems. This is clearly not an exhaustive list. Authorization is provided by the introduction of lightweight agents on the devices where information access occurs: a Realm Controller, a Virtual Realm database and Policy persistent storage. Authentication is provided by the Authentication Server, which may be domain-based or providing capabilities such as single sign on. Accountability is provided by the Audit Log.

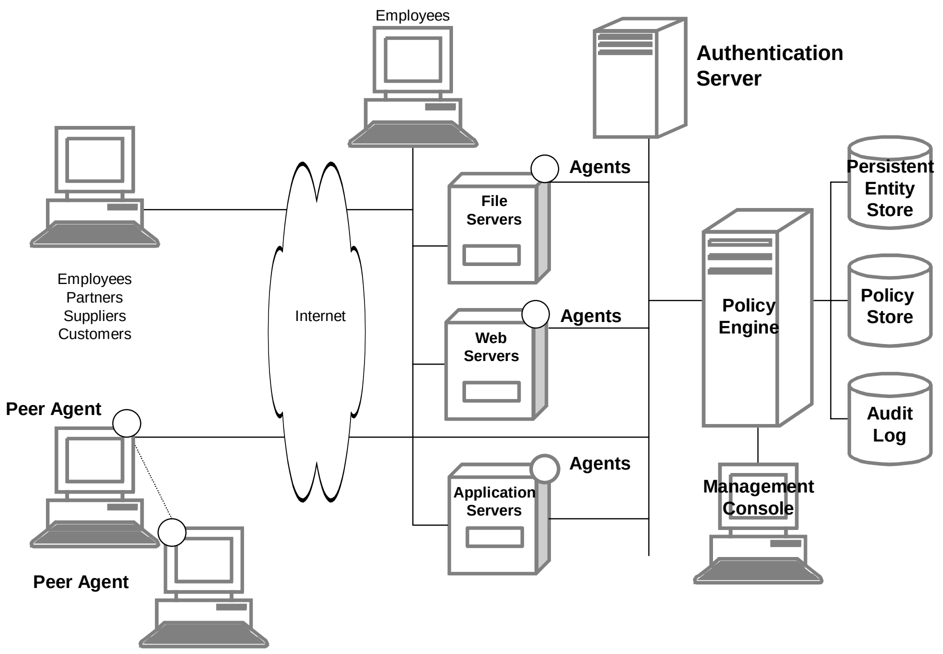

*Figure 12: Policy Engine Architecture*

The following sections describe essential functions of these components.

### 5.9.1 Communicating With a Policy Engine

A policy engine is incomplete unless there is some mechanism by which an external application can communicate with the engine itself and have the engine mediate access requests on behalf of the application. This is best performed by means of either a well defined API or well defined communications protocol based on some standard, such as XML, whereby queries are handled in a service architecture. Either is acceptable although more and more of the industry is moving towards an XML services model.

It is interesting to note that nothing precludes taking an existing policy language and embedding it into XML thereby providing both an API and service delivery model.

### 5.9.2 Persistent Storage of Entities

The persistent entity store is a database of all of the entities being protected. The store is organized around two basic structures: the entity and binary relationships between entities. Examples of binary relationships are:

- *parent(a,b)*, where *a* represents a directory and *b* a file within it;

- *read(x,y)*, where *x* is a user and *y* a file; and
- *execute(u,v)*, where *u* is a user and *v* an executable application.

This permits complex relationships to be created relatively easily while retaining a security perspective. It is crucial that the policy engine require that all interactions come via well-defined APIs so as to manage the full lifecycle of an entity; i.e. creation, utilization, modification, and deletion. The audit responsibility falls with the domain of responsibility of the policy engine and not the outlying agent or application that is calling the policy engine via the API.

### 5.9.3 Policy Engine Proper

The policy engine provides authorization services for client applications or agents. The policy engine answers the mediation request by interacting with the persistent store. A typical scenario works as follows:

1. Agent A sends request, "Can Alice read Design.doc?" The agent can also pass environmental information such as permissions that Alice has to access Design.doc.
2. Lookup entity for Alice.
3. Lookup entity for Design.doc.
4. If resource-based access, retrieve policy for Design.doc.
5. If user-based access, retrieve policy for Alice.
6. Evaluate policy retrieved, store result.
7. If result is false and we are using resource-based access, retrieve access violation policy for Design.doc.
8. If result if false and we are using user-based access, retrieve access violation policy for Alice.
9. If result is false, evaluate retrieved access violation policy.
10. Store mediation request and its result in the Audit Log.
11. Return result to requesting agent.

In order to achieve reasonable performance it would be beneficial that when a policy is loaded, it is compiled and cached. Although not strictly necessary, highly interactive systems such as Web servers or file servers would benefit enormously from such compilation, as subsequent evaluations would result in the retrieval and compilation step to be bypassed. Whenever a policy is modified within the Policy Store, the policy engine would be notified and the cached copy flushed.

The policy engine would also have to deal with entity lifecycle events. Whenever a new file is created or destroyed, the policy engine would have to be notified in order to update the policy store. This ensures that old entities do not litter the policy store, resulting in potential security problems.

### 5.9.4 Policy Store

Policy Store is a repository for all policies that may be associated with entities. It is responsible for lifecycle maintenance of policies. The repository maintains an association between the policy name and its actual implementation. The Policy Store is also responsible for notifying the policy engine whenever a policy changes in order that cached, compiled policies may be flushed.

### 5.9.5 Audit Log

The Audit Log is a repository for all mediation requests. Whenever a mediation request occurs, the request and its result are logged. Should an access violation policy be invoked, it too is logged. The Audit Log insulates the policy engine and policy writers from knowing anything about the log details; i.e., whether it is a simple flat file or a relational database.

### 5.9.6 Management Console

Although not crucial, it is beneficial if there is some mechanism in place to permit administrators and policy writers access to the various features of the policy engine and policy store. Today such an interface would typically be created in Java or as a Web-based front end to the actual policy engine.

# 6. Components of a Policy Engine

## 6.1 Introduction

This section describes the components that constitute a policy engine as well as a set of requirements that can be used to determine how flexible a given security policy engine or language actually is. No effort is expended examining the nuances of developing policies or utilizing any interfaces as that is beyond the scope of this survey. However, as will be seen, a majority of the policy engines and languages utilize a standard programming paradigm requiring developer coding of policies within standard text editors.

- Specifically, this section examines the following components of a policy engine:
- Policy Engine Overview;
- Reference Monitor;
- The Entity;
- Language Requirements;
- Functionality;
- Engine Particulars; and
- Generic Requirements.

## 6.2 Policy Engine Overview

A policy engine allows programmers to create and generate valid security policies from high-level, verifiable semantic descriptions of a security model. There must be a one-to-one mapping between the entities being protected and the security information maintained. The purpose of a policy engine is to provide sufficient security to applications so that they require only minor security specific instructions to meet even the highest security requirements.

A policy engine's primary data structure is typically called an entity or an object. An entity contains all pertinent security data. Security policies, associated with each entity, determine the behaviour of access attempts to a given entity. The various components required to develop a policy engine are discussed at some length in the following sections, as is the structure of an entity, its data elements, the facilities required to manipulate entities and policies, the base functionality of a policy language, and overall security functionality of a policy engine.

## 6.3 Reference Monitor

In 1972 Anderson [Anderson 1972] introduced a concept that continues to form the foundation of all trusted information systems:

> *The function of the Reference Monitor is to validate all references (e.g., references to programs, data, and peripherals) made by programs in*

*execution against those authorized for the subject (e.g., the user). The Reference Monitor not only is responsible for assuring that the references to access shared resource objects are authorized but also to assure that the reference is the right kind (i.e., read, or read or write, etc.).*

Typically the Reference Monitor model requires a combination of hardware, software, and firmware; this combination is called the *Reference Validation Mechanism*. Anderson proposed three guiding principles for the definition of any reference monitor:

- the Reference Validation Mechanism must be tamperproof;
- the Reference Validation Mechanism must always be invoked; and
- the Reference Validation Mechanism must be small enough to be subjected to analysis and tests to ensure that it is correct.

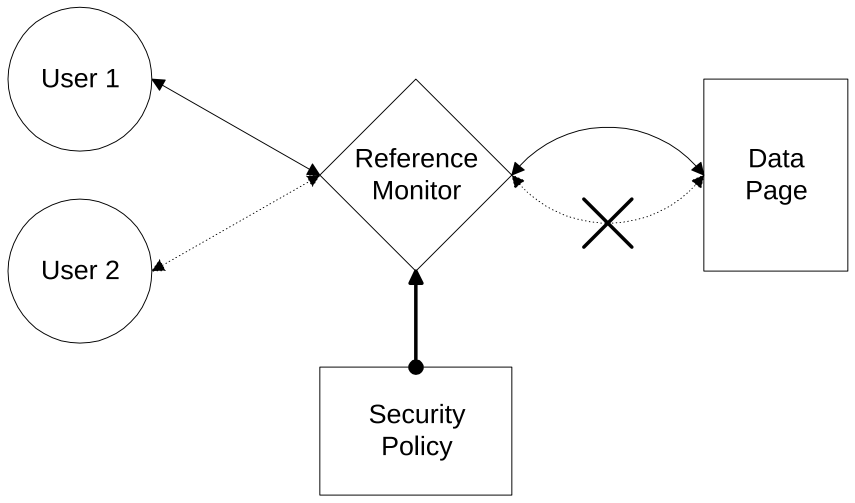

*Figure 13: Reference Monitor Model*

The operational characteristics of the reference monitor can be succinctly illustrated in Figure 13 which shows a reference monitor utilizing a security policy to allow or deny access requests from users based on a predefined set of rules associated with the policy. Trusted systems are designed to always invoke the reference monitor and to ensure it is not bypassable, thereby ensuring information flows as illustrated in Figure 13. It is within the reference monitor that all requests to access information are resolved by means of the security policy.

For our purposes we can imagine the policy engine to a programmable reference monitor.

## 6.4 The Entity

An *entity* is the unified security object against which all security related functions are performed. Both the base security application and any ancillary security relevant applications utilize entities in order to manage and maintain the security attributes of the applications. An entity is typically subdivided into six basic components: unique identifier, authentication information, audit, access control violation policy, state, and security policy.

## 6.5 Language Requirements

A true policy engine requires a policy language that provides system programmers with an expressive tool capable of elegantly defining security specific functionality[4]. By also ensuring the language is as simple as possible to program and has a small footprint, while retaining sufficient expressive power, we realize the following advantages:

1. we meet the requirements of the various security criteria;
2. we provide an easy to learn system, which is less likely to be bypassed in favour of direct low-level coding due to complexity issues;
3. we provide a unified means of providing authorization services across a variety of platforms; and
4. we minimize the footprint thereby limiting any performance impact.

## 6.6 Functionality

### 6.6.1 Subjects, Objects, and Processes

The functioning of a system can be viewed as successive operations on entities. Entities are all those things controlled and managed by the system. They can be files, peripherals, devices, physical memory, virtual memory, network packets, users, or programs – anything that can hold data. Entities have historically been defined as being in one of two states: active or passive. Active entities, called *subjects*, can access or manipulate passive entities. Passive entities, known as *objects*, are those being acted upon. Within most systems a subject is a process, job, or task operating on behalf of (i.e., a surrogate for) a user. Within this nomenclature, peripherals and daemons are treated as either subjects or objects depending on the observer's point of view. In order to alleviate confusion, the term entity (at times passive entity) shall be used to denote what is commonly referred to in computer security as an object. Subjects will be referred to as active entities. Processes will refer to those actions an active entity attempts against a passive entity.

In order to identify entities properly within the system, all must maintain unique identifiers to maintain security.

[4] Business managers and other non-technical staff typically have a requirement to create policies. Unfortunately, most tools today are written so that non-computer literate individuals, especially those without a good understanding of logic, cannot create logical and coherent policies. There are ongoing research efforts to address this via simplified GUIs, artificial intelligence, and natural language techniques. However, codified logic remains the clearest and most precise mechanism available to create consistent and logical policies for computer security purposes.

### 6.6.2 Identification & Authentication

Mechanisms must be put in place that provide for unique identification of every individual utilizing a system, with the proviso that the system require all individuals to authenticate themselves to the engine. The most common mechanism currently utilized, but by no means the only viable mechanism, is the standard login name-password pair.

### 6.6.3 Accountability

Mechanisms must be in place to uniquely identify and authenticate users and to dynamically track their actions within an application or system. Security controls, regardless of type, are not foolproof and to ensure a system of recourse after a security breach a non-circumventable, unalterable, continuous audit mechanism must be in place and operational. A base set of auditable events must exist which can never be disabled. The routines must be non-circumventable to ensure that neither accidental nor intentional modification of the audit system occurs.

The unique identification and authentication of every user ensures that individuals utilizing the system can be held accountable for their actions. The mechanics behind identification and authentication can vary greatly and as such the core elements should be customizable to suit the threat assessment and the particular architecture. The guarantee of a minimal auditable event list is vital to maximize the trustworthiness of an application.

### 6.6.4 Audit Logs

No matter the effort expended, it is never possible to declare a system 100% error free nor 100% secure. The small amounts of doubt that remain after a system is complete must be removed in order to ensure that unforeseen attacks will be recorded, enabling system personnel to correct the breach and perhaps to apprehend the culprit. Each time an auditable event is triggered, the engine writes relevant information, such as the entity performing the action or request, the date and time, the status of the request and other pertinent information to a log file, typically stored and protected separate from the system’s regular files. This log file must be regularly examined to determine the security state of the system being protected by the engine and ensure that no breaches have occurred.

### 6.6.5 Access Control

Information flow must be controlled so that access to and manipulation of information within the policy engine is restricted to a specific set of interfaces. As such there must be mechanisms provided to provide access to the capabilities of the policy engine. This would typically be done via a well-defined application programming interface (API) or via a published service capability. Attempting to access policies, entities, or any other data structure within the policy engine must be done with by and through APIs or services. This will guarantee a level of security and consistency and preclude manipulation of data structures except through well-defined means. In other words, it is necessary to heavily utilize data types to protect vital information while providing APIs that allow programmers to utilize the security provided by the policy engine.

### 6.6.6 Security Policy

By utilizing a programmable policy language, additional security policies can be modelled that enhance, complement, or override the default security policy implemented within each entity. Security policies can therefore be defined to reflect the requirements of the applications reflecting the policies and procedures pertaining to the information flow and control of a specific organization. Default security policies, those applicable to all entities, can be provided to simplify security policy maintenance. Furthermore, compatible and cooperative security policies can be written to represent the nuances particular to a given department or section.

## 6.7 Engine Particulars

### 6.7.1 Virtual Machine

Although not entirely necessary, a virtual machine architecture does create a clean implementation that allows for simpler validation of proper functioning than does a monolithic implementation of a policy engine. Virtual machines maximize portability and simplify the core and primitive elements of the language. Virtual machines have been shown to be small, efficient, and relatively simple to port (see Smalltalk, Scheme, and Java).

### 6.7.2 The Matter of Persistence

The controlled entities must be readily available between invocations. In order to do this there must be a database in place that ensures that all entities are readily available to the policy engine and to any policies implemented in the policy language. Manipulation, storage, and retrieval of entities should be transparent to the policy writer, much as garbage collection is transparent to users of modern functional and object oriented languages.

### 6.7.3 Trusted Systems

Users may be trusted to protect data but the same does not necessarily hold true for the computer programs that they run. We thus face the dichotomy of trusting the users but not trusting their tools, namely the various computer applications. This dichotomy is resolved by understanding the basic edict of any secure system: *By default, access to information shall be denied unless explicitly granted*. Therefore, for us to allow a user to properly access information we must categorize the programs they utilize into one of three categories:

| | |
|---|---|
| *Trusted:* | Software that is fundamental to the correct operation of the system. This software is typically responsible for enforcing security and must operate flawlessly. |
| *Benign:* | Software that is not responsible for any aspect of the system's security but is of known origin, perhaps supplied with the system being utilized. Being of known origin implies that it must be trusted (by the user) to manipulate protected information within the confines of the system's security. Any |

flaws are unintentional due to programmer error rather than malicious act and should not impact security.

*Malicious:* Software of unknown origin which, from a security perspective, must be treated as malicious and likely to attempt to breach the system's security

This implies that normal binary logic may not be adequate to define whether or not a particular action should be allowed or not. This means that the language should be fairly complete and sufficiently powerful to meet the varied needs of security policies as they are currently understood and as they may be understood.

## 6.8 Generic Requirements

In order to develop a policy engine a few fundamental requirements must be satisfied in order for the engine to be truly generic – i.e., able to define any policy that the end-user requires.

*Table 3: Requirements for a Policy Engine*

| # | Requirement | Description |
|---|---|---|
| **1** | ***Persistence*** | It must be possible to define and store any policy that may be associated with an entity. |
| **2** | ***Expressive*** | It must be possible to express policies that apply to intentionally defined associations (or groupings) of entities. There should not be a limit, logically or otherwise, as to the number and types of associations. |
| **3** | ***Assignment*** | It must be possible to define which policy (out of a collection of policies) is associated with a particular entity. It should be possible to dynamically change which policy covers which entity through simple means. |
| **4** | ***Natural Language*** | It would be desirable if there were the means by which to automate the natural language specification of a security policy into a logical one understood by the Policy Engine. In other words, taking Business Rules and converting them directly into logic understood by a Policy Engine. |
| **5** | ***Provable*** | The policy must be expressible and checkable in a well-defined and understandable logic/language. |
| **6** | ***Commutative*** | It must be possible for the engine to utilize either the passive and active entities' policies and identifying information in order to determine whether a particular request is actionable. |
| **7** | ***Mediatative*** | The language must be able to define and determine what is to occur upon a denied or allowed action. Furthermore, a violation of a policy must result in the (possible) invocation of a "violation" policy that will perform specific actions based on the failed logic of the originating policy. |
| **8** | ***Associative*** | It must be possible to determine to which entities a given policy is associated and, conversely, which policy is associated with a given entity. |
| **9** | ***Enforcement*** | Each policy must be well-defined and utilized to enforce access by one entity against another. |
| **10** | ***Labelled*** | Each entity controlled by the security policy must be tagged with a label. This label indicates information relevant to the security policy in |

| | | |
|---|---|---|
| | | determining how information can be accessed. In the case of the Bell-LaPadula model, the labels would indicate the clearance of the active entity and the sensitivity label of the passive entity. |
| **11** | ***Unique*** | All entities must be uniquely identified, authenticated, and assigned relevant information pertaining to which security policies are associated with the given entity. |
| **12** | ***Accountable*** | Security relevant events such as entity accesses, entity modification, process instantiation, and access of one entity by another must be recorded in an audit trail. The purpose is to detect and diagnose malicious or non-malicious actions that lead to a non-secure state. |
| **13** | ***Independent*** | The engine allows for policies independent of any underlying architecture. This permits the creation of individual policies that are pertinent to the problem at hand and not restricted to a single “type” of policy, such as Bell-LaPadula or Role-based. It also permits the mixing and matching of policies within the policy space (i.e., any combination of policy types, Lattice, RBAC, etc.). |
| **14** | ***Assured*** | The software must be designed and implemented in such a way that all the preceding requirements are enforced. The security aspects of a product must be of sufficiently simple organization and complexity to be subjected to and pass a wide spectrum of analyses and tests. |

# 7. Academic Research Alternatives

## 7.1 Overview

Research into security policies and security policy languages of all types has sped up as of 2002. One of the best summations of security and management policy specification was done by Morris Sloman and Emil Lupu of Imperial College [Sloman]. Their paper in 2002 provides an excellent summation of the state of the art in 2002 and the direction in which policy research was headed at that time.

Ironically, the paper comments that no reported deployment of a policy-based system on a large-scale existed. They also indicated that it was too early to judge whether policy-based systems would materialize. As this survey shows there was one policy-based system deployed on a large scale: SecureRealms. *It appears, as of the writing of this report, that this is still the only such system to ever be deployed.*

Further, Sloman and Lupu's paper indicates that there is considerable interest from government, standardization bodies, industry, and academia in policy research. The following sections outline some of the research that has transpired in the academic arena in policy-related research.

Specifically, this section examines the following academic research areas:

- Authorization Specification Language (ASL);
- Generic Policy Engine;
- LaSCO;
- MeSMO (Meta Security Model) & OASIS;
- PAMINA;
- PolicyMaker/KeyNote;
- Ponder;
- REFEREE;
- Sandhu Policy Research;
- SDSI/SPKI;
- Security Policy Language;
- XML-based Initiatives; and
- Other Policy Languages in the Literature.

## 7.2 Authorization Specification Language (ASL)

The Authorization Specification Language (ASL) as defined by Jajodia et al [Jajodia 1997] is composed of a set of nine (9) rules. These rules provide simple rules by which to perform complex tasks. The rules can be anded together to create complex policies. However, due to the high-level nature of each rule it is difficult to determine the complexity required to create a policy as well-known as Bell-LaPadula. Jajodia et al indicate that a graphical interface will be forthcoming to facilitate policy creation but it is impossible to determine whether this will be the case without actually seeing the final product.

The rules provided by ASL are:

*Table 4: ASL Rule Set*

| Rule Name | Syntax | Parameter description |
|---|---|---|
| Direct Membership | **dirin** ($s_1$, $s_2$) | |
| Indirect Membership | **in** ($s_1$, $s_2$) | |
| Object Grouping | **typeof** (o, T) | T: Object type |
| Done | **done** (o, u, R, a, t) | t: Integer (time) |
| Authorization | **cando** (o, s, <sign>a) ← $L_1$ & … & $L_n$ | $L_1$, …, $L_n$: *in*, *dirin*, or *typeof* literal |
| Derivation | **dercando** (o, s, <sign>a) ← $L_1$ & … & $L_n$ | $L_1$, …, $L_n$: *in*, *dirin*, or *typeof*, *done*, or *cando* literal |
| Resolution | **do** (o, s, <sign>a) ← $L_1$ & … & $L_n$ | $L_1$, …, $L_n$: *in*, *dirin*, *typeof*, *done*, *cando*, or *dercando* literal |
| Access Control | **grant** (o, s, <sign>a)← $L_1$ & … & $L_n$ | $L_1$, …, $L_n$: *in*, *dirin*, or *typeof*, *done*, *cando*, *dercando*, or *do* literal |
| Integrity | **error** (o, s, <sign>a)← $L_1$ & … & $L_n$ | $L_1$, …, $L_n$: *in*, *dirin*, or *typeof*, *done*, or *cando*, *dercando*, *do*, or *grant* literal |

| | | | |
|---|---|---|---|
| *Legend:* | *u: User* | *R: Role* | *s: Subject (User, Group, or Role)* |
| | *o: Object* | *a: Action* | *<sign>: + or -* |

No publicly available implementation has been discovered even after extensively perusing the various documents describing this academic policy engine. Hence, this engine cannot be included in the list of possible options for DRDC's efforts. It does not appear that ASL has been actively researched since the original publication of this report in 2003.

## 7.3 Generic Policy Engine

The Generic Policy Engine [Bacic 1998], further described by Tony White and Eugen Bacic [Bacic 2002a, Bacic 2002b, White 2003], and the precursor of the Texar SecureRealms policy engine, contained a programmable policy engine capable of implementing and enforcing any policy that could be defined programmatically. It utilized a Scheme-based interpreter to allow for complex security policies which were attached to "entities", effectively any resource within a system. Each entity could then be protected by a unique policy specific to the needs of the object being protected.

The policy engine provided authorization services via an exposed API that allowed external programs to request mediation between entities as well as a storage local for policies that were associated with a given entity.

Because all policies were written in a Scheme-like dialect all policies were full fledged functions written in a Turing complete language. The structure of the Generic Policy Engine was such that there was no limit to how many policies could co-exist within the local database and be associated with a given entity.

One of the reasons Bacic selected a dialect of Scheme is that Scheme has well-defined denotational semantics that can be utilized to prove the correctness of a given policy or set of policies.

The design goals of the Generic Policy Engine from its inception as the Master's thesis by Bacic were:

- integration with existing products;
- uniform, ubiquitous security across the enterprise;
- controlled trust between intranet and extranet systems;
- centralized and uniform controls; and,
- programmable policies that reflect the business rules.

The Generic Policy Engine became SecureRealms, a product of Texar Corporation, in 1999. Texar ceased operations in 2003 and development of SecureRealms was halted. The source code was abandoned at the same time that development stopped. Those interested in SecureRealms can find the basic principles in Bacic's thesis [Bacic 1998] and other Bacic papers [Bacic 1989, Bacic 1990a, Bacic-Kuchta, Bacic 2002, Bacic 2002a, Bacic 2002b, White 2003].

## 7.4 LaSCO

LaSCO [Hoagland] is a graphical approach for specifying security constraints on objects, in which a policy consists of two parts: the domain (assumptions about the system) and the requirement (what is allowed assuming the domain is satisfied). Policies defined in LaSCO have the appearance of conditional statements used to express authorizations between objects in the system and are stated as policy graphs. A policy graph is an annotated directed graph where the annotations are domain and requirement predicates. Nodes in the policy graph represent the sorts of objects described by the associated domain predicate. Collectively, the nodes, edges, and domain predicates form the domain of a policy graph. The domain describes when the policy is in effect, i.e. when it applies. The other part of the policy graph is the requirement, which consists of the requirement predicates on each of the nodes and edges. A node requirement predicate is an expression that must be met on the object and constitutes an authorization policy. LaSCO cannot specify any form of obligation policies, and there is no way of composing policies or specifying policies for groups of objects apart from those defined for classes of objects. This makes the scope of this approach very limited to satisfy the requirements of security management. In addition, graphs are often used in conjunction with a textual version to specify details not easily expressed in the graphical format. In LaSCO this is lacking, making the language difficult to use and further restricting its expressiveness. Note however, that a graphical approach to specifying policies is attractive for human users, and is thus an interesting future research direction.

At present it appears that LaSCO [Hoagland] is a language for stating and enforcing authorization policies for Java programs. LaSCO provides a policy compiler that adds policy checks into Java source code as wrappers on method invocations. LaSCO can express only what events must be present, but not what events must be absent, for the policy to be applicable. LaSCO cannot state system liveness properties such as obligations. Also, LaSCO cannot state policies relating to objects that are in more than one distinct state.

For this reason it is felt that LaSCO is not a suitable candidate for consideration by DRDC's policy engine requirements. It also appears as though LaSCO is no longer actively supported or pursued at UC Davis.

## 7.5 MeSMO (Meta Security Model) & OASIS

MeSMO is a meta security model for OASIS, an Open Architecture Security for Information Systems. OASIS is a system described by Essmayr et al [Essmayr] whose purpose was to create an enterprise-wide system for administrating and enforcing security in distributed heterogeneous systems. The system generally is designed similarly to a reference monitor with a centralized policy engine being queried with respect to particular access requests. Based on the result of the mediation the action is allowed or denied.

MeSMO was designed to handle general security abstractions and, from the literature, appears to be in line with what a true generic policy engine would be. MeSMo utilizes high-level abstractions to describe heterogeneous security concepts such as subjects, objects, and the various relationships among them as well as the access types. Although the literature indicates that several prototypes were developed no further information on MeSMO can be discovered and it is believed that it is a dead project. It does provide validation of the architecture proposed for a policy engine outlined in this survey and was shown to work against a file system as well as a database. What the actual security policies looked like and the language of implementation are not described.

As no additional information has been found on MeSMO and it appears not to have been pursued, it must remain a curiosity and validation of the policy engine concept.

## 7.6 PAMINA

PAMINA (Privilege Administration and Management INfrAstructure) [Nochta] is an authorization system based on authorization certificates working in conjunction with X.509 or similar directories. The purpose of this research was to show that authorization certificates in an I-CVT (Improved Certification Verification Trees) [Gassko] structure could be developed that would result in a provably correct authorization infrastructure. Initial work created a working version in Java. This work was published in late 2002 and no subsequent reports have been discovered. However, the authors did specify that additional R&D was being pursued and for this reason it would be of interest to track developments with PAMINA to see where the authors take their authorization solution that links PKI and PMI, especially considering DRDC's interest in such technology

## 7.7 PolicyMaker/KeyNote

PolicyMaker [Blaze 1996] and its successor, KeyNote [Blaze 1999], are verifiers which give yes/no answers to questions of the form 'is the proposed operation safe to execute in terms of the defined security policy?' (written as key1, key2, key3 REQUEST actionstring). The benefit of PolicyMaker/KeyNote is that policies (specifically, security policies) are defined outside of the application code, rather than hard-coded into applications, and can therefore be altered by responsible users [Blaze 1999].

In [Blaze 1996] and [Blaze 1999], two trust management applications are presented: PolicyMaker and its successor KeyNote. Both of these applications are used to answer signed queries of the form "does a set of requested actions r, supported by credential set C, comply with policy P?", where the credentials can be public key certificates with anonymous identity. Both policies and credentials are predicates specified as simple C-like regular expressions. In this context a policy is a trust assertion that is made by the local system and is unconditionally trusted by the system. Although trust management systems provide an interesting framework for reasoning about trust between unknown parties, assigning authorizations to keys may result in authorizations that are difficult to manage [Samarati]. In addition, providing a common solution to both authentication and access control makes the system more complex.

The source code to KeyNote as well as sample applications were made available by the University of Pennsylvania at their KeyNote page (http://www.cis.upenn.edu/~keynote/). For this reason PolicyMaker/KeyNote will be examined as a possible option for consideration by DRDC's survey of available security policy engine technology. However, it does not appear that any active research is being pursued and no papers have been written concerning PolicyMaker or KeyNote since 2000.

## 7.8 Ponder

The Ponder language for specifying Management and Security policies [Damianou] evolved from work on policy management at Imperial College over a period of about 10 years. Ponder is a declarative, object-oriented language that can be used to specify both security and management policies. Ponder authorization policies can be implemented using various access control mechanisms for firewalls, operating systems, databases and Java [Corradi]. It supports obligation policies that are event triggered condition-action rules for policy-based management of networks and distributed systems. Ponder can also be used for security management activities such as registration of users or logging and auditing events for dealing with access to critical resources or security violations. Key concepts of the language include domains to group the objects to which policies apply, roles to group policies relating to a position in an organization [Lupu 1997], relationships to define interactions between roles, and management structures to define a configuration of roles and relationships pertaining to an organizational unit such as a department.

Ponder [Damianou] appears to be the most complete and fully tested representation of a policy engine framework. It can express authorization and obligation policies as well as basic descriptive policies through an associated domain service. Ponder is intended to improve system flexibility by not hard-coding policies into components but removing them out of the core applications and system kernels into separate, well-protected domains.  Sloman [Sloman 1994, Sloman 1995] argues for the need to specify, represent, and manipulate policy information independent of management components to enable dynamic change of policies and reuse of components within different policies. A policy service provides the means for storing policy specifications, determining the objects to which they apply, performing analysis to detect conflicts, and disseminating policies to managers for interpretation [Sloman 1995]. The capability to have both positive and negative authorization policies allows a very rich set of access control specifications to be created. Obligation policy support enabling event-triggered actions adds to the richness available. Furthermore, Ponder utilizes path-based domain scoping whereby it becomes possible to specify which sets of objects a policy applies. In other words, Ponder allows every entity to have its own associated security policy. Even though Ponder introduces its own language syntax it is well worth examining in more detail as a viable option for DRDC's policy engine survey.

Unfortunately, as of the writing of this survey update, Ponder is no longer supported nor available for download. The documentation remains available at the Ponder site (http://www-dse.doc.ic.ac.uk/Research/policies/ponder.shtml) but the source code has been removed.

## 7.9 REFEREE

REFEREE (Rule-controlled Environment for Evaluation of Rules, and Everything Else) is a result of collaboration among researchers from AT&T and the W3C. It was designed to be a general-purpose execution environment for all Web applications requiring trust. REFEREE evaluates user policies in response to a host application's request for actions modelled after a reference monitor. Policies are treated as programs in REFEREE and appear in a LISP-like syntax. For a given request, REFEREE invokes the appropriate user policy and interpreter module and returns to the host application a response as to whether the requested action complies with the policy.

Because REFEREE is a true rule-based system that regards policy as coded logic, it resembles to a fair degree the Texar SecureRealms main policy engine. Although SecureRealms utilizes a Turing Complete language at its core, REFEREE allows for the inclusion of more complex policy languages so long as the application programmer desires such. This flexibility is also one of REFEREE's drawbacks, because if the provided PicsRULZ policy language is insufficient, the programmer will have to extend it for his or her particular application. A full API is provided to interact with REFEREE and current efforts have focused on utilizing REFEREE as a solution to Web-based security policy issues and problems.

A sample policy from REFEREE may illustrate its power, and the fact that policies are very LISP-like in nature:

```
(invoke “load-PICS-label” STATEMENT-LISP http://software-security
          URL (EMBEDDED “http://label-bureau))
(invoke “check-signature” (match (* (* (for URL) *))
(match    ((“load-PICS-label”)
          ((version “PICS-1.1”)
           (service http://software-security)
           (by “virus-buster”)
           (ratings (RESTRICT > confidence 8))
        STATEMENT-LIST)
```

*Figure 14: Sample REFEREE Policy*

This sample REFEREE policy says to allow downloading the code pointed to by this URL if and only if “virus-buster” allows it. Obviously, it is not the most readable of policies. Almost all the work done on REFEREE up to and including 1997 was with respect to its applicability to protecting the Web and Web Servers.

REFEREE has the advantages of being part of the W3C Open Source initiative for Trust Management. Although not much effort has been put into REFEREE since 1997 it is available and may be an excellent candidate for use as a base for future development at DRDC, especially R&D efforts that focus on the Web rather than on generic authorization in a distributed environment.

As of the writing of this survey update, it appears that work on REFEREE has terminated as it no longer appears on the AT&T web site of active research projects. Furthermore, the W3C site has not been updated since 1997 with the third, and apparently final, release of the source code to the sample REFEREE implementation.

## 7.10 Sandhu Policy Research

Due to the amount of research into computer security in general and policy engines and languages specifically by Ravi Sandhu and his team at George Mason, the following describes Sandhu et al’s ongoing work in the field.

### 7.10.1 Schematic Protection Model

This was the original work done by Ravi Sandhu in 1985 [Sandhu 1985, Sandhu 1988] that led to much of his work on RBAC and security policy modelling [Sandhu 1996, Sandhu 1998, Sandhu 2000]. It is included here for completeness and for historic record. It is of primary interest in that it introduces and solidifies the notions surrounding the use of types for security objects and subjects and their relationships with one another.

Sandhu and his colleagues at George Mason University have continued their research into security policies and policy languages as can be seen in the following section. The Schematic Protection Model has been abandoned.

### 7.10.2 Constraint Specifications

Sandhu et al. at George Mason University have been generally busy in the area of policy enforced solutions to information security. Along with the Schematic Protection Model above and the Usage Control model presented below, Sandhu has also spent time examining variations on RBAC. One of the more interesting efforts is the work done during 1999 and 2000 on a Role-based Separation Language, RSL.

This work was done by Gail-Joon Ahn and Ravi Sandhu [Ahn-Sandhu 1999, Ahn 2000]. The research focused on constraints as an important aspect of role-based access control. Although this research focused on modelling a constraint-based language, Ahn and Sandhu did provide soundness and completeness proofs for the RSL99 and RCL 2000 versions of the constraint language they defined. A grammar and architectural diagram are provided within the papers, as well as sample algorithms for specific constraints.

Discussion was given towards future directions for both RSL99 and RCL 2000 but work seems to have stopped as Sandhu began pursuing the Usage Control Model, a much more flexible policy language not tightly coupled to role-based solutions nor to a constraint language.

Additional work on constraint-based languages has been done by Joshi et al. [Joshi 2003]. Joshi et al. leverage RBAC by extending it temporally to create a Generalized Temporal Role-based Access Control (GTRBAC) model that captures temporal constraints for access control. GTRBAC allows for the application of temporal constraints against roles, user-role assignments, and role-permission assignments. Joshi et al. then couple that with the ability to apply time constraints to control flow and separation of duty to create temporally-based dynamically changing access control. Bhatti's thesis [Bhatti] provides an XML-based extension to GTRBAC, however no implementation is currently publicly available.

### 7.10.3 Usage Control Model

In 2004 Park and Sandhu introduced the $\text{UCON}_{\text{ABC}}$ Usage Control Model [Park 2004, Park 2002, Zhang]. This model integrated authorization, obligation, and conditions into a form of access control. The purpose behind this new research was to define a more generic approach to security policies that went beyond the traditional confidentiality, mandatory, integrity, discretionary, and role-based solutions that typically predated this work.

By defining a *unified framework* Park and Sandhu attempt to unify access control, trust management, and digital rights management (DRM). They accomplish this via a well thought out language to define the usage controls while ensuring privacy concerns are addressed. This fusion of access control, trust management, and digital rights management is unique. As a superset it actually allows the creator of a UCON policy to select how much of each element they desire within their policy. UCON can provide pure DRM or pure mandatory controls or a hybrid of almost anything a security policy author desires.

The work done during 2002 and 2003 were paper models. The 2004 paper [Park 2004] presents a much more detailed report on the progress of the research, including an excellent comparison to other efforts such as XrML, ODRL, and other rights languages as well as comparisons to classic access control solutions.

The 2004 work also provides more detailed information on the architecture of UCON which closely resembles the Generic Policy Engine as presented in [Bacic 98]. The primary difference is in how security attributes are associated with a given object. In the Generic Policy Engine attributes can be associated with any entity, which can be any resource within a system. Within UCON the security attributes are actually the objects, which are a set of entities, upon which subjects hold rights. Thus, objects contain security labels and other information that can be examined by an object.

This is similar to the Generic Policy notion of entities with attributes, but within the Generic Policy Engine *everything* is deemed an entity so that what differentiates an object from a subject is the fact that an object is passive while a subject is active. A subject may act upon another subject, something that appears not to be possible with the UCON model as an object is considered to be “derivative” of an original work and resource and thus passive.

As of the writing of this survey no implementations of UCON have been made publicly available, though UCON remains actively researched by Sandhu’s team at George Mason University.

## 7.11 SDSI/SPKI

The Simple Distributed Security Infrastructure (SDSI) [SDSI] developed by Ron Rivest, Carl Ellison, and Butler Lampson at MIT combines a public-key infrastructure design with a means of defining groups and issuing group-membership certificates. These groups provide a mechanism and a clear terminology for defining access-control lists and security policies. SDSI's design emphasizes an association of linked local name spaces rather than a hierarchical global name space, ensuring that any type of relationship that exists in computer security can be adequately modelled. Furthermore, these associations provide a simple way by which authority can be delegated.

SDSI is yet another policy language built using a LISP-like notation, in this case S-expressions. The implementation utilizes S-expressions that not only simplify parsing but also provide a description of the security data and an executable format all in one. In SDSI, certificates allow one user to give authority to another user, for example permission to read a file. Java implementations of SDSI exist and are available freely on the Internet. They provide the means to implement the fundamental SDSI objects, such as certificates and public keys that can be used to build application tools to work with an SDSI-enabled infrastructure.

Since SDSI utilizes S-expressions, fundamental data elements become code fragments as well, as illustrated in the “executable” access control list shown in Figure 15 below.

```
(acl:  (read: associates))
(acl:  (read: newsweek-subscribers))
(acl:  (read: verisign))
(acl:  (write: (or: bob bob-assistant)))
(acl:  (read: (or: bob bob-friends mit-faculty))
       (write: ron))
```

*Figure 15: Sample SDSI ACLs*

As can be seen in the above example, the S-expression “(acl: …)” illustrates a data element containing specifics as to whether an individual can read or write to the associated resource. However, it is possible to execute the definition and have the S-expression act as logic returning to the calling routine who has or doesn’t have access to a resource.

SDSI is of particular interest to DRDC considering the purpose of the Proof-of-Concept in linking PKI and PMI and the requirement to facilitate the development of policies. The only issue is whether complex policies can be developed or whether the general architecture of SDSI precludes any policy other than access control-based ones.

SDSI seems to be only partially supported. Many of the original links are no longer functional, although John Pritchard seems to have created a web page dedicated to SDSI/SPKI (http://www.syntelos.com/spki/) though it hasn’t been updated since 2001. There does not appear to be much advancement with respect to SDSI/SPKI since 2002 and it appears to have been abandoned.

## 7.12 Security Policy Language

The Security Policy Language (SPL) [Ribeiro 1999, Ribeiro 2001a], used for database transaction control, is designed to express policies about the acceptability of events. An event-driven policy language, SPL supports access control, history-based, and obligation-based policies. SPL is implemented by an event monitor that determines, for each event, whether to allow, disallow, or ignore said event. Events in SPL are synonymous with action calls on target objects, and can be queried to determine: the subject who initiated the event, the target on which the event is called, the attribute values of the subject and target, and the event itself. SPL supports two types of sets to group the objects on which policies apply: groups and categories. Groups are sets defined by explicit insertion and removal of their elements, and categories are sets defined by classification of entities according to their properties. The building blocks of policies in SPL are constraint rules which can be composed using a specific tri-value algebra with three logic operators: and, or, and not. A simple constraint rule is comprised of two logical binary expressions, one to establish the domain of applicability and another to decide on the acceptability of the event.

SPL defines two abstract sets called *PastEvents* and *FutureEvents* to specify history-based policies and a restricted form of obligation policy. The type of obligation supported by SPL is a conditional form of obligation, which is triggered by a pre-condition event. SPL obligations are thus additional constraints on the access control system, which can be enforced by security monitors [Ribeiro 2001b], and not obligations for managers or agents to execute specific actions on the occurrence of system events, independent of the access control system.

The notion of a policy is used in SPL to group set definitions and rules together to specify security policies that can be parameterized; policies are defined as classes that allow parameterized instantiation. Instantiation of a policy in SPL also means activation of the policy instance, so no control over the policy life cycle is provided. Further re-use of specifications is supported through inheritance between policies. A policy can inherit the specifications of another policy and override certain rules or sets. Policy constructs can also be used to model roles, in which case sets in the policy specify the users allowed to play the role and those playing the role. Rules or other nested policies inside a role policy specify the access rights associated with the role. SPL provides the ability to compose policies hierarchically by instantiating them inside other policies, thus enabling the specification of libraries of common security policies that can be used as building blocks for more complex policies. The authors claim that this hierarchical composition also helps restrict the scope of conflicts between policies, however this is not clear, as there may be conflicts across policy hierarchies. Note that SPL does not cater for specification of delegation of access rights between subjects, and there is no explicit support for specifying roles.

SPL appears to have been abandoned around 2002 as no additional information or papers have been published by the original SPL team nor any other researchers. The source code to SPL was never released.

## 7.13 XML-based Initiatives

### 7.13.1 Semantic Web Languages

Tonti et al. [Tonti] provide an excellent comparison of semantic web languages for policy representation, with a particular focus on KAoS, Rei, and Ponder. Although not a policy language paper per se, it does provide an excellent summation of what constitutes a semantic web language and how the selected languages compare. With an eye to policies, which constrain system component behaviour, Tonti et al seek to determine how well semantic web languages address the notion of dynamic adjustability of applications and data within the World Wide Web. Tonti et al also focus on the management issue surrounding the semantic web and how well these languages facilitate the management of the semantic web and its ultimate goals as outlined by the W3C.

### 7.13.2 SWRL: Semantic Web Rule Language

Li et al [Li 20 05] presented their new Semantic Web Rule Language (SWRL) which is a Horn Clause rule extension to OWL. This rule language uses Prolog-like language [Clocksin] to define antecedent-consequent clauses that define rules that are human readable:

$$\text{file (?a) \& (fileOwner (?a, ?b)} \wedge \text{memberGroup (?b, ?c))} \Rightarrow \text{permitAccess(?a).}$$

Some of this effort has been formalized into the RuleML work (http://www.ruleml.org/) which is attempting to create a Horne Clause-based policy language. Li et al. are continuing their efforts on SWRL and have been working with the W3C RuleML and SWRL effort (http://www.w3.org/Submission/SWRL/) to formalize their approach.

### 7.13.3 χ-Sec Credential Specification Language

χ-Sec is a credential-based language for specifying subject credentials and security policies and for providing a mechanism by which they can be organized into profiles and policy bases. The overall solution proposed by Bertino [Bertino 2001] is subscription-based and focuses on the notion that attempts to access web resources is done in a heterogeneous way. They moreover focus on the fact that not only are the access attempts performed by a heterogeneous user community but that the community is characterized by differing skills and needs.

χ-Sec attempts to address the issues surrounding web documents and the fact that the web is highly dynamic and that the credentials required to access a resource may not reside on the same machine that houses the resource.

χ-Sec provides a number of "credential types" such as *business manager*, *customer*, and *carrier*. These are then used in a manner similar to roles to determine access. Although χ-Sec appears to be fairly flexible, most of the examples provided by Bertino et al are based on e-commerce and web-based document access case studies. One of the more interesting aspects of χ-Sec is that it appears to contain logic built into its policy language specification for proper and complex work flow. There exists the capability to define the flow of a transaction, including delegation flows. All through the flows χ-Sec ensures that the entity attempting access is properly accredited and that their credentials are valid.

It is unknown whether χ-Sec is currently being actively pursued or not. The only paper on χ-Sec was published in 2001.

## 7.14 Other Policy Languages in the Literature

There are a good many policy languages and efforts underway at universities and research labs worldwide. What is provided in this section are those policy systems and languages that show some level of promise and may be worth tracking from an R&D perspective. They are presented in alphabetical order.

### 7.14.1 Event-Trigger-Rules

The University of Florida has worked on a specification scheme similar to the ECA rule-based approaches with some key differences [Su]. The development of their approach, called Event-Trigger-Rule (ETR) paradigm, is motivated by the need for rule-based processing capabilities in the distributed environment of electronic commerce enterprises [Su]. Included here for completeness, the ETR paradigm is a generalization of the ECA approach where the event specification and conditions and actions of the rule are specified as separate entities. Specifying a trigger then associates the event and rule together into a policy. This is in contrast to the ECA rule specification approach, where the event specification, associated conditions and actions be combined into a single rule.

Even though ETR is not of particular use to the requirements of this survey, it does show that a number of orthogonal efforts are underway that are effectively re-inventing the policy engine paradigm for their own purposes.

### 7.14.2 Open Distributed Programming Reference Model (ODP-RM)

The group working on the International Organization for Standardization (ISO) Open Distributed Programming Reference Model (ODP-RM) define an enterprise language as part of the RM-ODP

Enterprise Viewpoint [ISO/IEC 1999], which incorporates concepts such as policies and roles within a community. A community in RM-ODP terminology is defined as a configuration of objects formed to meet an objective. The objective is expressed as a contract, which specifies how the objective can be met, and a configuration, which is a collection of objects with defined relationships between them. The community is defined in terms of the following elements:

- the enterprise objects comprising the community;
- the roles fulfilled by each of those objects and the relationships between them;
- the policies governing the interactions between enterprise objects fulfilling roles;
- the policies governing the creation, usage and deletion of resources;
- the policies governing the configuration of enterprise objects and assignment of roles to enterprise objects; and
- the policies relating to the environment contract governing the system.

Policies constrain the behaviour of enterprise objects that fulfil actor roles in communities and are designed to meet the objective of the community. Policy specifications define what behaviour is allowed or not allowed and often contain prescriptions of what to do when a rule is violated. Policies in the ODP enterprise language thus cover the concepts of obligation, permission and prohibition.

The ODP enterprise language is really a set of abstract concepts rather than a language that can be used to specify enterprise policies and roles. Recently, there have been a number of attempts to define precise languages that implement the abstract concepts of the enterprise language. These approaches concentrate on using UML to depict the static structure of the enterprise viewpoint language graphically, as exemplified by [Steen 2000], and on languages to express policies based on those UML models. Steen et al. [Steen 1999, Steen 2000] propose a language to support the enterprise viewpoint where policy statements are specified using the grammar shown below. Each statement applies to a role, the subject of the policy, and represents a permission, an obligation or a prohibition for that role. The grammar of the language is concise, however it does not allow composition of policies or constraints for groups of policies. Constraints cannot be specified to restrict the activation and deactivation of roles, nor the assignment of users and permissions in roles.

For these reasons it is felt that the ODP enterprise language is not a viable candidate for inclusion in the viable option list for the DRDC policy engine survey; although it remains an interesting technology to watch.

### 7.14.3 Rei

In 2002 Kagal [Kagal 2003] specified and created the general policy language *Rei* including an associated policy engine that interprets security policies written in *Rei* while working at HP. In 2003, while at the University of Maryland, Kagal further explored *Rei* [Kagal 2003]. Kagal focuses her efforts with *Rei* on distributed and pervasive environments via a prototype system.

Although not much has changed in the overall language as compared to her work while at HP (see HP *Rei*, below) she has slightly refocused her attention away from networks in general and more towards pervasive systems and networks in particular.

The language continues to be modelled on deontic logic, including the notions of rights, prohibitions, obligations and dispensations. Furthermore, *Rei* allows actions and conditions to be external to the system and assumes that the meaning of actions or conditions are domain dependent and that their complete processing is outside the policy. The notion is that by using these actions and conditions a policy author is able to create policy objects.

There are four kinds of policy objects: rights, obligations, prohibitions, and dispensations. In order to associate these policy objects with users, the policy author uses the has predicate, creating rights, obligations etc. for the users. *Rei* models four speech acts that can be used within the system to modify policies dynamically: delegate, revoke, cancel and request. In order to make correct policy decisions, it expects all relevant speech acts about the resources and users in its policies to be input to it. The *Rei* policy language contains meta-policy specifications for conflict resolution. Associated with the policy language, is a policy engine that interprets and reasons over the policies and speech acts to make decisions about users rights and obligations.

*Rei* assumes that there can be more than one applicable policy within a domain. This may lead to policy conflicts and the solution proffered is to utilize the meta-policies to resolve conflicts by setting applicable precedence and other rules to govern how rules trigger.

The 2003 paper provides details of the Virgil System, which uses *Rei* as a crucial element. A fully distributed network of nodes, Virgil provides the testbed for *Rei* that allows Kagal to garner better results in terms of the functionality and capabilities *Rei* can offer in terms of distributed computer security policies.

*Rei* derives some of its ideas from Ponder, including the ability for every entity to have its own policy rule. However, it does not create its own language syntax, instead relying on Prolog as the basis for its security language, and allowing for policies to be specified in RDF. Unfortunately, *Rei* is, at the moment, an internal to HP project with no public source available and only a single document written describing its functioning. No detailed description exists nor working models and therefore, although fascinating, it must be excluded from consideration as a viable option for DRDC's policy engine survey.

Most of Kagal's recent work is in regard to the semantic web (http://www.rei.umbc.edu) and *Rei* has been repurposed as a policy language based on OWL-Lite as opposed to the general purpose language it was originally framed as while Kagal was at HP. Work is ongoing on *Rei* as of the publication of this survey. Work is being sponsored through the Univeristy of Maryland's ebiquity Group.

### 7.14.4 SULTAN

SULTAN (Simple Universal Logic-oriented Trust Analysis Notation) is an abstract, logic-oriented framework designed to facilitate the specification, analysis and management of trust relationships. It is the Ph.D. thesis of Tyrone Grandison [Grandison] at Imperial College. The initial primary focus is on specification and analysis. It has been designed and tailored for use by management and system administrators who have a global view of the system resources and needs of an organization. SULTAN assists in the creation of organization-specific solutions to problems of trust in an interconnected environment.

As a logic-oriented framework, SULTAN's logical evaluations (constraints, in fact) evaluate to true or false. The SULTAN documentation stresses that "logic-oriented" does not mean logical or logic-based.

What you can specify with SULTAN are trust relationships and recommendations, something akin to if-then clauses. It accomplishes this through its five main components:

- **Trust establishment** defines the protocols by which parties wishing to form a trust relationship can negotiate and exchange evidence and credentials, which can be provided to an evaluation service.
- **Trust evaluation** service gathers and evaluates the evidence for defining a trust relationship. This could be based on a risk analysis of the context for the relationship, experience from past interactions with the trustee or recommendations from other entities, such as colleagues, who have experience of interacting with the entity within a similar context. In addition, a trusted third party evaluation service such as a consumer organization, which evaluates goods or services, can be used for recommendations.
- **Trust specification** defines trust relationships in terms of the parties involved, and the context of the interaction. Multiple trustors or trustees to be involved in a relationship and a trustor/trustee can be involved in multiple different relationships. The context is defined as a set of actions with a trust level applying to all the actions, and a set of constraints, which must be evaluated for the trust relationship to apply. At the moment most of the trust specifications developed are aimed at authentication or access control methods. Efforts are underway to create more abstract, high-level policies.
- **Trust monitoring** updates experience and risk information.
- **Trust analysis** examines trust relationship specifications to identify unwanted implicit relationships and possible conflicts of relationships.

SULTAN can interoperate with Ponder, mentioned elsewhere in this survey. Briefly, Ponder is a language for specifying security and management policies, while SULTAN is a language for specifying and analysing trust relationships. The two frameworks can be connected either by using SULTAN in Ponder policies, or by using Ponder as the target for the refinement of SULTAN rules.

SULTAN remains a Ph.D. thesis. No source is available. It does appear to be an interesting concept and is probably worth watching.

### 7.14.5 Trust Policy Language (TPL)

The Trust Policy Language (TPL) by IBM [Herzberg] provides a fairly clear separation between the authentication of subjects based on certificates and the assignment of authorizations to those subjects that have been successfully authenticated. With TPL, the credentials result in a client being assigned to a role which specifies what the client is permitted to do, where a role is a group of entities that can represent specific organizational units (e.g., employees, managers, auditors). TPL relies heavily on X.509v3 certificates and defines the language in XML, which makes the syntax rather verbose. Viable implementations are also wanting.

This, too, is not a viable candidate for DRDC's policy engine requirements.

### 7.14.6 Miscellaneous Role Specification Languages

The following languages all fall under the "role specification" umbrella. None are capable of addressing the requirements of DRDC's need for a policy engine, however each presents interesting information on RBAC extensions that address some of the shortcomings of RBAC as it was originally defined. Neither of the two languages discussed have actual implementations listed as having been developed or available.

#### 7.14.6.1 Role Definition Language (RDL)

The Role Definition Language RDL [Hayton] is based on Horn clauses and was developed as part of the Cambridge University Oasis architecture for secure interworking services. RDL is based on sets of rules that indicate the conditions under which a client may obtain a name or role, where a role is synonymous to a named group. The conditions for entry to a role are described in terms of credentials that establish a client's suitability to enter the role, together with constraints on the parameters of those credentials. The work on RDL also falls into the category of certificate-based access control.

#### 7.14.6.2 Role Specification Language (RSL)

RSL [Ahn-Sandhu 1999] is another role specification language which extends the ideas introduced in [Chen] and can be used for specifying separation of duty properties in role-based systems. The language covers both static and dynamic separation of duty constraints, and its grammar is simple, although the expressions are rather complicated and inelegant.

# 8. Standards & Open Source Alternatives

## 8.1 Overview

This section discusses standards and Open Source efforts in access control policy expression, knowledge representation, and DRM that DRDC may leverage in building and/or integrating an access control policy engine.

## 8.2 Access Control Policy Efforts

This section discusses germane standards and Open Source alternatives geared to the expression and enforcement of access control policy.

### 8.2.1 IETF/DMTF Policy Core Information Model

Network policy specification, as per [RFC 3060], provides an interesting comparison in order to determine what additional technologies should be surveyed for this report. Network policy is the set of the rules that define the relationship between clients using network resources and the network elements that provide those resources, so the similarity to security policies is obvious. The main interest in network policies is to manage and control the quality of service (QoS) experienced by networked applications and users, by configuring network elements using policy rules. The most notable work in this area is the IETF's policy model, which considers policies as rules that specify actions to be performed in response to defined conditions:

*if <condition(s)> then <action(s)>*

which looks suspiciously like security policy event-action triggers. The condition-part of the rule can be a simple or compound expression specified in either conjunctive or disjunctive normal form. The action-part of the rule can be a set of actions that must be executed when the conditions are true. Although this type of policy rule prescribes similar semantics to an obligation of the form event-condition-action, there is no explicit event specification to trigger the execution of the actions.

Instead it is assumed that an implicit event such as a particular traffic flow, or a user request will trigger the policy rule. The IETF approach does not have an explicit specification of authorization policy, but simple admission control policies can be specified by using an action to either allow or deny a message or request to be forwarded if the condition of the policy rule is satisfied.

Figure 16 presents a simple example the types of rules an administrator can specify. In this example, the administrator has given high priority for particular multicast traffic for the corporate management sub-network on Monday nights from 6:00pm to 11:00pm:

```
if ((sourceIPSubnet = 224.0.0.0/240.0.0.0) AND
   (timeOfDay = 1800-2300) AND
   (dayofweek = Monday))
then
   set Priority := 5
```

*Figure 16: Sample IETF Policy*

The policies as specified do not allow for complex nesting or call-outs but do provide the rudimentary aspects of a policy language. It remains a good idea to examine the existing IETF literature on an on-going basis to realize what does and does not work from a policy presentation perspective.

The IETF does not define a specific language to express network policies but rather a generic object-oriented information model for representing policy information following the rule-based approach described above. Early attempts at defining a language [Strassner] have been abandoned. The IETF are defining an information model to represent policies that administer, manage, and control access to network QoS resources for integrated and differentiated services [Snir]. The philosophy of the IETF is that business policies expressed in high-level languages, combined with the network topology and the QoS methodology to be followed, will be refined to the policy information model, which can then be mapped to a number of different network device configurations. Vendors following the IETF approach are using graphical tools to specify policy in a tabular format.

Other approaches to network policy specification try to extend the IETF rule-based approach to specify traffic control using a concrete language. An example is the path-based policy language (PPL) from the Naval postgraduate school described in [Stone]. The language is designed to support both the differentiated as well as the integrated services model and is based on the idea of providing better control over the traffic in a network by constraining the path (i.e., the links) the traffic must take.

Since the 2003 survey, refinements and extensions to the IETF/DMTF Policy Core have followed, but the principal aim of such efforts has been to specify bindings for use with LDAP servers and SNMP MIBs [RFC 3460, 3703, 4011, 4104]. Currently, there is no standard means of expressing IETF/DMTF Policy Core items in XML, although it would appear a straightforward exercise.

### 8.2.2 GSS API / GAA API

The Generic Authorization and Access-control API (GAA API) [Ryutov-Neuman] provides fine-grained access control and application-level intrusion detection capabilities that are accessible to applications through a simple API. As an IETF initiative and part of the Open Source movement, code has been published and is available the Internet at the following address: (http://gost.isi.edu/info/gaaapi/distribution/gaa-announce.html). Applications can use this framework to delegate access control and application-level intrusion detection to the GAA API. The GAA API was developed to address the requirements of most applications so as to allow developers to focus on the tasks at hand and not the authorization mechanisms. Furthermore, the designers' goal was to create a standardized authorization mechanism. Baseline implementations exist, and are in use, for Apache and IPsec.

The GSS API provides the developer with:

- fast integration;
- extensibility and portability;
- a simple policy language called the **Extended Access Control Lists (EACL)**;
- **conditions** associated with granted (or denied) access rights; and
- a **three-phase policy enforcement scheme** to evaluate conditions.

The associated Generic Security Service Application Program Interface (GSS API) presented by Linn at the 1990 USENIX Security Workshop [Linn 1990] presents an API that addresses specific concerns regarding the definition of security services. Much like the GAA API, the GSS API assumes and addresses several basic goals, including:

- **mechanism independence:** the GSSAPI is independent of the particular underlying mechanisms used to implement any given portion of the API;
- **protocol environment independence:** no particular communications protocol is assumed; and
- **suitability to a range of implementation placements:** no restrictions are placed upon the GSSAPI clients to reside within a given trusted computing base (TCB) perimeter.

The GSS API is network security specific and attempts to generalize the security requirements prevalent within networks available today. The GSS API provides peer entity authentication, per-message data origin authentication, and data integrity protection. A certain amount of data confidentiality is also provided; however, the GSS API is primarily a network message authenticator.

Although the GSS API is an interesting additional security service for the GAA API, it is the GAA API that is of real interest. Although not as powerful as some policy engines, its tight integration with Web-based applications make it particularly interesting for the PMI efforts of DRDC.

### 8.2.3 XACML

XACML (eXtensible Access Control Markup Language) [OASIS] is an XML specification for expressing policies for information access over the Internet. It is a standard defined by the Organization for the Advancement of Structured Information Standards (OASIS) technical committee. The language permits access control rules to be defined for securely browsing XML documents that can update individual document elements. Similar to other existing policy languages, XACML is used to specify a subject-target-action-condition oriented policy in the context of a particular XML document. The notion of subject comprises identity, group, and role and the granularity of target objects is as fine as single elements within the document. The language supports roles, which are the same as groups, and are defined as collections of attributes relevant to a principal. XACML includes conditional authorization policies, as well as policies with external post-conditions to specify actions that must be executed prior to permitting an access.

Note that XACML is intended to be used in conjunction with SAML (Security Assertion Mark-up Language) assertions and messages, and can thus also be applied to certificate-based authorizations. Also of interest is the fact that Sun Microsystems released an Open Source implementation of XACML. The goal is that XACML replace proprietary access control mechanisms and be integrated into products from file servers to Web services. Sun's XACML code is available for download at (http://sunxacml.sourceforge.net/).

XACML describes both an access control policy language and a request/response language. The policy language is used to express access control policies. The request/response language expresses queries about whether a particular access should be allowed (requests) and describes answers to those queries (responses).

In a typical XACML usage scenario, a subject (e.g. human user, workstation) wants to take some action on a particular resource. The subject submits its query to the entity protecting the resource (e.g. file system, web server). This entity is called a Policy Enforcement Point (PEP). The PEP forms a request (using the XACML request language) based on the attributes of the subject, action, resource, and other relevant information. The PEP then sends this request to a Policy Decision Point (PDP), which examines the request, retrieves policies (written in the XACML policy language) that are applicable to this request, and determines whether access should be granted according to the XACML rules for evaluating policies. That answer (expressed in the XACML response language) is returned to the PEP, which can then allow or deny access to the requester.

The developers of XACML believe it is sufficiently flexible to define most access control policies and adequately extensible to support new requirements. Although preliminary examination reveals that XACML is not a sufficiently rich language to define complex policies, it does provide the ability for one policy to refer to another. The unknown question that remains is whether XACML provides the means by which to create complex rules that reflect the real world. An interesting R&D project would be to attempt to define standard policies such as Bell-LaPadula, Clark-Wilson, Biba, Lee as well as OS-style policies such as those found in Windows 2000, Unix, etc.

With the release of Sun’s Open Source toolkit, XACML it becomes a viable candidate to fulfil DRDC’s policy engine requirements.

Since the 2003 Survey, OASIS has standardized XACML v2.0, incorporating the Core Specification [Moses 2005a] and new extension profiles to accommodate:

- Hierarchical RBAC [Anderson 2005a];
- Hierarchical Resource Specification [Anderson 2005b];
- Multiple Resource Specification (per request) [Anderson 2005c];
- Privacy [Moses 2005b];
- Digital Signature [Anderson 2005d]; and
- SAML v2.0 assertion and protocol mechanisms [Anderson-Lockhart 2005].

To anticipate LDAP distribution of policy elements, the XACML v2.0 specification includes corresponding LDAP structural object classes and defines the following policy behaviours accordingly:

- policy posting;
- policy retrieval;
- policy validation; and
- policy combination.

XACML v2.0 provides PEP profiles implementing three biases:

- base PEP;
- deny-biased PEP; and
- permit-biased PEP.

The new specification refactors some clumsy XML syntax, and standardizes several new functions, data types, and algorithms. It remains a viable candidate to fulfill DRDC's policy engine requirements.

#### 8.2.4 EPAL

IBM developed the Enterprise Privacy Authorization Language (EPAL) and submitted it to the W3C. The XML-based language has been a W3C submission [Ashley 2003] in v1.2 since 2003, but the W3C has not progressed it toward recognition as a standard.

EPAL is a formal XML-based language to exchange privacy policy in a structured format between applications or enterprises, allowing organizations to:

- comply demonstrably with their stated policies;
- reduce costs in enforcing data handling policies; and
- employ existing standards and technologies in the effort.

EPAL allows fine-grained specification of policies for positive and negative authorization rights, with obligations for both cases, in order of precedence. EPAL policies define lists of hierarchies of:

- data-categories – hierarchical categories of collected data, differentiated in terms of privacy;
- user-categories - hierarchical categories of user that use collected data;
- purposes - intended services for which data is used; and
- sets of:
    - actions – how the data is used;
    - obligations – actions to be taken by the environment of EPAL; and

- conditions – necessary conditions on context, as transmitted in parameterized containers.

EPAL vocabularies name elements of authorization policies. However, EPAL does not allow for nesting of policies, nor named policy references.

EPAL does not set predefined data-categories, user-categories, purposes, actions, obligations, or conditions. Rather, the language offers a framework for an enterprise to specify such as befits its own authorization profile. The effort does incorporate rulings of allow, deny, and not-applicable into the language. However, EPAL does not include any language-defined means of combining results where multiple policies apply to any given request. Furthermore, the language does not allow for multiple responses where a request refers to a hierarchical structure of resources [Anderson 2005f].

EPAL requires the system using it to determine and enforce applicable policies on all protected resources. Furthermore, EPAL relies on policy enforcement engines to interpret and enforce obligations on positive and negative authorization decisions: they are opaque. As such, there is no guard against mutually exclusive obligations. EPAL does not define policy-directed handling of error conditions or missing attributes.

EPAL can accommodate dynamic roles through careful specification of policies and dynamic assignment of user categories at run-time. However, EPAL's structure does not lend itself to hierarchical role engineering as per [ANSI 359], since it builds from user-categories rather than permissions in defining roles.

EPAL is an IBM-proprietary specification offering a subset of the W3C standard XACML's amenity [Anderson 2005f]; furthermore, there are no publicly available implementations. EPAL therefore does not represent a likely candidate for a DRDC policy engine.

### 8.2.5 NIST RBAC Efforts and Adoption

The (U.S.) National Institute of Standards and Technology (NIST) supports the American National Standards Institute / International Committee for Information Technology Standards (ANSI/INCITS) efforts in standardizing RBAC. This section discusses such advances and their acceptance.

#### 8.2.5.1 ANSI/INCITS 359-2004

ANSI published ANSI/INCITS 359 [ANSI 359] in 2004. This specification can be regarded as the de facto RBAC standard. It defines a consensus model of RBAC that can be used in a full range of applications. The standard itself is copyright material and not freely available, but the NIST overview [Ferraiolo 2001] provides formal logical definitions for Core RBAC, Hierarchal RBAC, and Constrained RBAC (including Separation of Duty, both Static and Dynamic). The discussion provides an overview of the functional specification for compliance to each of the above models, and then motivates eight profiles:

- Core RBAC, plus zero or more of:

- Hierarchical RBAC;
- Static Separation of Duty (SSD) Relations;
- Dynamic Separation of Duty (DSD) Relations.

Note that these specifications employ formal logic, not XML-based language.

The (U.S.) National Institute of Standards and Technology (NIST) has released its first draft for RBAC implementation [NIST 2006]. The standard defines terms and establishes requirements for a design to comply with [ANSI 359]. Again, the specification employs formal logic, not XML-based language.

#### 8.2.5.2 Biometric Standards

INCITS Working Group M1 is developing a set of biometric standards that refer to RBAC, including [ANSI 359]. This early work is still in progress.

#### 8.2.5.3 Healthcare Initiatives

HIPAA ((U.S.) Health Insurance Portability and Accountability Act, 1996) advises RBAC to protect patient information, and cites HP's model integrating policy-driven RBAC with the common data security architecture [Lin 1999]. This effort extends the HP Common Data Security Architecture (CDSA) with constraint-based RBAC, using a Prolog-based policy description language and a policy enforcement mechanism.

The Health Level Seven (HL7) is an ANSI-accredited Standards Developing Organization for the exchange, management, and integration of electronic health care information. Its activities advance application-level systems employing the services defined in the general purpose RBAC standards [Healthcare 2003].

#### 8.2.5.4 Veterans Affairs

The (U.S.) Department of Veterans Affairs is leading ongoing Role Engineering efforts in particular [VA 2006], in compliance with [ANSI 359].

#### 8.2.5.5 Industrial Control Systems

The Instrumentation, Systems, and Automation Society (ISA) working group SP99 (Manufacturing and Control Systems Security) has initiated the adoption of RBAC into emerging standards to secure networks and applications in process control systems such as power plants and manufacturing facilities. These early efforts [Mintchell 2004] continue to develop.

## 8.3 Knowledge Representation Effort: RuleML

Rule Markup Language (RuleML) is a Semantic Web initiative to express rules in XML for inferential and transformational tasks. The goal of the Rule Markup initiative is to develop

RuleML as the canonical Web language for rules using XML markup, formal semantics, and efficient implementations. The initiative maintains a web site at http://ruleml.org/ to publish its progress.

### 8.3.1 Design Methodology

RuleML is a rule markup (XML) language, with associated Knowledge Representation (KR) semantics, suitable as a standard for interchange of rules (and facts) in XML over the Web between heterogeneous applications using heterogeneous rule systems. Its developers have built the language on a foundation of formal logic and inference engines, gearing it to the general expression and evaluation of rules.

The core technical approach of RuleML is based on declarative logic programs (LP). RuleML includes additional expressive extensions including:

- URI's as predicates/constructors;
- prioritized conflict handling;
- procedural attachments; and
- named argument roles.

RuleML's scope ranges from derivation rules to transformation rules to reaction rules. It is not geared specifically to the expression of access control policy.

RuleML is currently in draft v0.9. The standard is geared to expressing facts and rules of inference. DRDC would need to define XML schema using RuleML to satisfy its Policy Engine's requirements before proceeding. RuleML standards are Open Source and available.

### 8.3.2 Inclusions

As an expression of Knowledge Representation (KR), RuleML presents an intuitive fit for policy based decision-making. The language includes formalism for expressing premises, conclusions, known facts, and epistemological norms that, for any given set of premises, formally defines an associated set of sanctioned conclusions. Access control policies represent well as rules; policies can then evaluate through a rule inference engine.

Logic Programs express knowledge representation. RuleML can mark any parameter as a known constant or as an unknown variable. Therefore, RuleML could be made to craft queries for result sets satisfying a given set of constraints. For example, an Attribute Query could request all known attributes that a given user possesses. This could allow more distributed intelligence throughout the network, as opposed to centralizing access control decisions at a designated PDP.

RuleML consists of a galaxy of independently-developed modules, many of which overlap in scope, and some of which adopt clashing naming conventions arising from the name churn that the language has suffered between v0.8 and v0.9. Some of these modules define constructs for prioritized conflict handling, procedural attachments, and rule labelling.

### 8.3.3 Exclusions

RuleML is not geared specifically to rule-based access control. It does not assume a PMI network architecture, and does not currently admit tokens for specific query or statement types.

RuleML does not at present formally integrate the prevailing XML standards for resource location, encryption, and digital signature. Neither does the language exclude extensions to incorporate the possibility; relying parties may cite or codify their own extensions and work out where to identify them in any pertinent XML documents. Indeed, many have done so independently and at cross-purposes.

RuleML does not identify algorithms, such as X.500 Distinguished Name matching or RFC822-name matching, to match query argument content. Neither does the language exclude extensions to incorporate the possibility; relying parties may cite or codify their own matching relations and work out where to identify them in any pertinent XML documents.

### 8.3.4 Adoption

The RuleML initiative has achieved large mindshare in the Semantic Web community, who are principally academics at this point. At present, the technology is still embryonic and has been in relatively rapid state of churn, although at v0.9 it appears that several aspects of the language are beginning to stabilize.

The W3C SWRL submission [Horrocks 2004] adopts RuleML in an effort to express knowledge representation in a manner consistent with W3C OWL.

Any developers intending to use RuleML to build a Policy Engine should be familiar with Knowledge Representation, formal ontology, Expert System design, PMI concepts, and current XML initiatives.

Selecting a profile of RuleML and extending it could serve as a basis for a DRDC Policy Engine. Doing so would represent a significant custom design effort because RuleML has not tracked ongoing XML developments. As such, some of the work would first need to align the profile with current prevailing trends in XML.

## 8.4 DRM Efforts

DRM refers to technology to enforce policies for controlling access to digital data (such as software, music, movies) and hardware. In more technical terms, DRM handles the description, layering, analysis, valuation, trading, monitoring and enforcement of usage restrictions that accompany a specific instance of a digital work.

This section explores emerging DRM standards with a view to seeding a Policy Engine.

### 8.4.1 ODRL

The Open Digital Rights Language (ODRL) version 1.1 is currently a W3C submission [ODRL 1.1] at www.w3.org/TR/odrl. The W3C has not progressed ODRL on a standards track since its submission in 2002, but an ODRL working group continues to collaborate on v2.0. ODRL is available publicly as an Open Source initiative.

The ODRL model for rights information consists of:

- Assets:
  - any uniquely identifiable content; and
  - may include encryption information for secure delivery;
- Rights:
  - Permissions (actual usage allowed over an Asset);
  - Constraints (limits to Permissions);
  - Requirements (obligations needed to exercise the Permission); and
  - Conditions (exceptions to control Permissions);
- Parties:
  - End Users;
  - Roles; and
  - Rights Holders (who may assert some form of ownership over the Asset and its Permissions);
- Offers:
  - Proposals from Rights Holders for specific Rights over Assets; and
- Agreements:
  - when Parties enter contracts or deals with specific Offers.

ODRL supports a set of limits to Permissions including constraints on space and time. ODRL supports Requirements for Permissions granted, but not for Permissions refused.

ODRL uses two XML schemas: one for its Rights Expression Language (REL) and one for its Data Dictionary elements; in particular:

- permission types;
- requirement types;
- constraint types;
- condition types;
- context types;and
- rights holder types.

The Data Dictionary is extensible, within these predefined types.

ODRL makes no assumptions on permissions not explicitly mentioned (i.e., silence = denial), and provides no means of resolution on permissions in conflict.

ODRL's scope is limited to the specification of framework for metadata elements describing digital rights and conditions for exercising them. The effort does not include a protocol for authorization decision queries and responses; nor does it specify a mechanism to enforce permissions.

The language could represent a candidate to address a subset of DRDC's needs in creating an access control policy engine.

#### 8.4.2 ISO-IEC 21000 / MPEG 21 / IPMP / XrML

The Moving Picture Experts Group (MPEG) refers to DRM as Intellectual Property Management and Protection (IPMP). The group's aim in this area is to provide a framework enabling all users to express their rights and interests in, and agreements related to, Digital Items and to have assurance that such will be managed and protected across a wide range of networks and devices. To this end, the MPEG-21 effort includes the (Section 5) Rights Expression Language (REL) and the (Section 6) Rights Data Dictionary (RDD). The result is an extensible language geared to expressing user permissions, with support for conditions on (positive) permissions.

The MPEG-21 specifications are standards of ISO-IEC, numbered 21000. The standards are proprietary, but available to use for a fee.

ContentGuard's Extensible Rights Management Language (XrML 2.0) forms the basis for the MPEG-21 Section 5 REL standard [Rightscom 2003]. MPEG-21 Section 5 is essentially the same language as XrML 2.0, as per Section 9.3. The framework still does not anticipate policy obligations on access request refusals.

The MPEG-21 Section 6 Rights Data Dictionary (RDD) offers a methodology to define and catalogue terms necessary for the description of rights, fostering interoperability and thus enhancing the value of the REL.

The MPEG-21 (Section 5) REL is based upon the proprietary XrML specification owned by ContentGuard – itself a private company principally owned by Time Warner Inc. and Microsoft Corporation. It is ContentGuard's position that "products or systems that implement either of the [XrML or ISO MPEG REL] specifications may infringe one or more of ContentGuard's patents and will need a patent license from us."

Licensing issues aside, MPEG-21 could constitute a candidate for use in developing a comprehensive DRDC Policy Engine, once any actual implementations appear. At present, no such offering exists.

### 8.4.3 Open Media Commons / Project DReaM

The Open Media Commons' Project DReaM [Fernando 2005] is a Sun Labs initiative to develop a DRM solution centred on open-standards-based solutions, but capable of integrating with proprietary solutions.

The DReaM project considers DRM based on three independent variables:

- number of content producers (few vs. many);
- binding of licences (to devices, individuals, roles, family, etc.); and
- dynamicity of licences (ability to change rights assignments in real time).

The project replaces the notion of an REL with a Mother-May-I (MMI) model, whereby clients negotiate with the licence service for terms of use and then translate such into functional licence service locally. Effectively, this protocol acts as a session-based yet-another-REL with corresponding requests and responses, but the architecture effectively employs the MMI as a Policy Decision Point (PDP). The protocol is based on HTTP (not XML), but provides guidelines for extension in the likely event that neither of its profiles (for documents, multimedia, or applications) suffice.

This effort is mainly an attempt to bridge competing Open Source and proprietary DRM schemes. It offers no additional functionality in specifying or enforcing complex access control policy. It is an inappropriate choice as the basis of a comprehensive DRDC Policy Engine.

### 8.4.4 OpenIPMP

OpenIPMP [OpenIPMP 2003] is a content management and distribution framework to secure digital assets. Key concepts include:

- user management;
- content identification and management; and
- rights management.

OpenIPMP allows for the expression of licences using MPEG-REL or ODRL. Each MPEG-REL or ODRL XML stream (“licence”) is signed by the OpenIPMP (X.509-based) Certificate Authority for authenticity, and is cryptographically specific to its intended recipient.

OpenIPMP is geared primarily to restricting access to digital content. The effort is mainly an attempt to bridge competing Open Source and proprietary DRM schemes. It offers no additional functionality in specifying or enforcing complex access control policy. It represents an inappropriate choice of basis for a comprehensive DRDC Policy Engine.

### 8.4.5 Authena

The Authena effort [McGucken 2001] uses an entirely Open Source (e.g., PGP and not X.509) approach and Dublin Core metadata nomenclature to specify licences, costs, and ratings for content. Authena identifies three profiles of content for licence:

- pristine (original length and quality);
- thumbnailed (abbreviated/degraded/excerpt/video clip for browsing); and
- watermarked (abbreviated/degraded/lower quality for browsing).

This approach is not sufficiently generic to serve as a basis for a policy engine. This survey includes it only for completeness.

# 9. Research Laboratory & Industry Consortium Alternatives

## 9.1 Overview

This section examines research laboratory and industry consortium alternatives in the area of policy expression. Specifically, this section examines the following initiatives:

- Bell Lab's Policy Description Language (PDL);
- ContentGuard's eXtensible Rights Markup Language (XrML);
- HP's Rei;
- Secure European System for Applications in a Multi-Vendor Environment (SESAME);
- Liberty Alliance; and
- Web Services Interoperability Organization (WS-I).

## 9.2 Bell Labs' Policy Description Language (PDL)

The Policy Description Language (PDL) is an event-based language from Bell-Labs [Lobo] in which an event-condition-action rule paradigm of active databases is used to define a policy as a function that maps a series of events into a set of actions. PDL's syntax is simple and policies are described by collections of two types of expressions: policy rules and policy-defined event propositions. Policy rules are expressions of the form:

*event causes action if condition*

where the *action* is executed if the *event* occurs under the provided *condition*. Policy defined event propositions are expressions of the form:

*event triggers policy-defined-event if condition*

where the *policy-defined-event* is triggered if the *event* occurs under the provided *condition*.

Events can be primitive or complex, and there are two types of primitive events: policy-defined-events, which are only generated by policy defined event propositions, and system events, which are generated by the environment. Primitive event classes can define attributes, and instances of the classes take actual values for those attributes that can be referenced by other events, actions or conditions within the same rule. Primitive events can be composed to form complex events that enable policies to be enforced under any of the following situations:

- If two events e1 and e2 occur simultaneously;
- If an event e does not occur;
- If an event e2 immediately follows an event e1;

- If an event e2 occurs after an event e1.

Despite its expressiveness, PDL does not support access control policies, nor does it support the composition of policy rules into roles, or other grouping structures. The language has clearly defined semantics and an architecture has been specified for enforcing PDL policies. Work on conflict resolution for policies written in PDL is described in [Chomicki], and extensions to the language to specify workflows for network management can be found in [Kohli]. The language has been used by Lucent to program their switching products [Virmani] and has been shown to be powerful in a variety of network operations and management scenarios. Unfortunately, it does not appear to be sufficiently powerful to work as a security policy engine for the purposes of this DRDC survey.

## 9.3 ContentGuard XrML

ContentGuard's eXensible rights Markup Language (XrML) is a general-purpose language in XML used to describe the rights and conditions for using digital resources (such as content, services, or software applications). Using XrML, anyone owning or distributing digital resources can identify the parties allowed to use those resources, the rights available to those parties, and the terms and conditions under which those rights may be exercised.

ContentGuard sells the XrML Software Development Kit (SDK) that enables developers to focus in their area of expertise and interface with XrML.

Due to its flexibility, XrML is one policy engine option that DRDC should consider.

XrML is a digital rights language used to specify rights, terms, and conditions. ContentGuard claims that XrML is the most advanced and mature rights language available and that it is:

- **Comprehensive**, providing a framework to express rights at different stages of a workflow or life cycle;
- **Generic**, defining a large body of format and business neutral terms that you use to specify rights for any digital content or service;
- **Precise**, using a grammar and processing rules to ensure unique interpretation of the language; and
- **Extensible**, allowing any third party to define elements to meet specific business needs.

XrML exploits XML technologies including its syntax and grammar as well as its schema. XrML is therefore tied tightly to XML, which can be viewed as a plus or a minus depending on the application being considered. As illustrated in Figure 17, the XrML data model for a rights expression consists of four basic entities and the relationship among those entities. This basic relationship is defined by the XrML assertion "grant". Structurally, an XrML grant consists of:

- the principal to whom the grant is issued;
- the right that the grant specifies;
- the resource to which the right in the grant applies; and

- the condition that must be met before the right can be exercised.

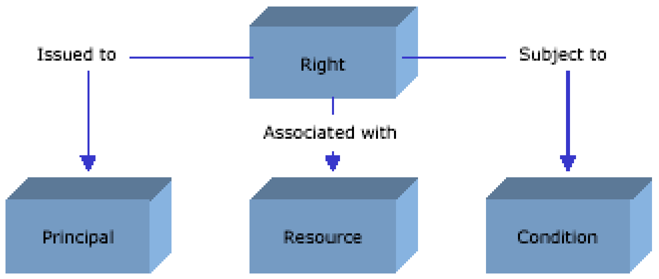

*Figure 17: Diagram illustrating basic ContentGuard SDK & XrML*

### 9.3.1 Principals

A principal encapsulates the identification of those users (entities) to whom rights have been granted. Each principal uniquely identifies one user. XrML allows for the creation of groups of principals, if desired. Principals are authenticated with an associated authentication mechanism whereby the principal proves its identity.

### 9.3.2 Rights

A right is the "verb" that a principal can be granted to exercise against some resource under some condition. Typically, a right specifies an action or class of actions that can be performed on a particular resource. Specific rights are typically predefined, such as the commonly used rights: *issue*, *revoke*, and *obtain*. Extensions to XrML allow for the definition of additional rights according to need and requirement. For example, it would be possible to create rights pertaining to the actions *print* or *decrypt*.

### 9.3.3 Resources

A resource is the entity to which a principal can be granted a right. A resource can be anything that can be uniquely defined to the system. This includes such things as files, applications, network connections, etc. Each resource is typically owned by a principal.

### 9.3.4 Conditions

A condition specifies the terms, conditions, and obligations under which rights can be exercised. A simple condition is a time interval within which a right can be exercised. A slightly complicated condition is to require the existence of a valid, prerequisite right that has been issued to some principal. Using this mechanism, the eligibility to exercise one right can become dependent on the eligibility to exercise other rights. It appears that the complexity this provides may be sufficient to create even extremely complex policies. In fact, XrML allows for the definition of what could be termed "functions", effectively a grouping of conditions to create complex behaviour.

### 9.3.5 Grants

A grant conveys to a particular principal the sanction to exercise an identified right against an identified resource, possibly subject to first fulfilling some conditions. It appears that a principal can delegate grants either on a time basis, a one-time basis, or any other conditional basis. This is a fairly scarce capability amongst the various policy engines surveyed.

## 9.4 HP's Rei

[Kagal 2003] defines work performed to specify and create the general policy language *Rei* including an associated policy engine that interprets security policies written in *Rei*. The language, modelled on deontic logic, includes notions of rights, prohibitions, obligations and dispensations. Furthermore, *Rei* allows actions and conditions to be external to the system and assumes that the meaning of actions or conditions are domain dependent and that their complete processing is outside the policy. The notion is that by using these actions and conditions a policy author is able to create policy objects.

There are four kinds of policy objects: rights, obligations, prohibitions, and dispensations. In order to associate these policy objects with users, the policy author uses the has predicate to create rights, obligations etc. for the users. *Rei* models four speech acts that can be used within the system to modify policies dynamically: delegate, revoke, cancel and request. In order to make correct policy decisions, it expects all relevant speech acts about the resources and users in its policies to be input to it. The *Rei* policy language contains meta-policy specifications for conflict resolution. Associated with the policy language, is a policy engine that interprets and reasons over the policies and speech acts to make decisions about users rights and obligations.

*Rei* is seen by its creators as the core policy engine within a larger Policy Server. The Policy Server would retrieve policies associated with domains, map the domain specific names into unique names for the Policy Server, and insert the policies into *Rei*. Every time an agent requests a certain action, the *Rei* policy engine would be queried. *Rei* would check if the requester has the right, check for prohibitions and revocations, and utilize a meta-policy associated with the agent and the requested action to resolve any conflicts. If the agent has the right, *Rei* would inform the owner of the action and allow the owner to interpret the action and handle its execution.

*Rei* also assumes that there can be more than one applicable policy within a domain. This may lead to policy conflicts: the solution proffered is to use meta-policies to resolve conflicts by setting applicable precedence, and other rules to govern how rules trigger.

*Rei* derives some of its ideas from Ponder including the ability for every entity to have its own policy rule. It does not create its own language syntax, instead relying on Prolog as the basis for its security language, and allowing for policies to be specified in RDF. Unfortunately, *Rei* is, at the moment, an internal to HP project with no public source available and only a single document written describing its functioning. Neither detailed descriptions nor working models exist and therefore, although fascinating, *Rei* must be excluded from consideration as a viable option in DRDC's policy engine survey.

## 9.5 Secure European System For Applications In A Multi-Vendor Environment (SESAME)

SESAME is a European research and development project integrating single sign-on with added distributed access control features and cryptographic data protection [Ashley 2000].

At sign-on, the client authenticates to an Authentication Server to receive a secure identity token. The user presents the token to a Privilege Attribute Server to obtain a set of access rights, encoded in a Privilege Attribute Certificate (PAC). The user then pushes the PAC to any target application. SESAME supports delegation of rights, and includes mechanisms to prevent inappropriate delegation.

SESAME PACs support group-memberships, roles and administration-defined local types. SESAME admits GSS-API bindings. Its single sign-on components are accessible through the Kerberos protocol [RFC 1510]., SESAME leaves each implementation free to interpret the semantics of administration-defined local types: the standard provides no data dictionary.

SESAME's authorization model is geared to access control, not to policy specification. The framework does not provide any constructs for the incorporation of tertiary computations in access control policy. Ultimately, SESAME provides ACL enforcement for files only, albeit for both users and roles. Worse for a policy engine, in accordance with POSIX v6 principles, user permissions necessarily override role permissions. As such, SESAME would be of limited use in building a comprehensive DRDC policy engine.

## 9.6 Liberty Alliance

The Liberty Alliance comprises a consortium of over 150 organizations (including Sun, IBM, and HP) to address technical, business, and policy challenges around identity and identity-based Web services.

The Liberty Alliance efforts have produced XML standards specifying:

- the Identity Federation Framework (ID-FF), enabling identity federation and management through features such as:
  - identity/account linkage;
  - single sign-on; and
  - basic session management;
- the Identity Web Services Framework (ID-WSF), providing the framework for building:
  - interoperable identity services;
  - permission-based attribute sharing; and
  - identity service description and discovery, and associated security profiles; and
- Identity Services Interface Specifications (ID-SIS), providing interoperable identity services, including technology such as:

- a personal identity service;
- a contact book services;
- a geo-location service; and
- a presence service.

The framework specifies SAML v2.0 to transport attributes.

Liberty toolkits are available for the Java and .NET frameworks to support single sign-on capabilities and consolidation of enterprise authentication schemes.

With respect to authorization policy, the effort assumes familiar Access Control network entities (e.g., Policy Decision Points, Policy Enforcement Points), but any federated identity technology is ultimately complementary to an Access Control Policy Engine.

## 9.7 Web Services Interoperability Organization (WS-I)

WS-I is an industry organization that creates, promotes and supports generic protocols for the interoperable exchange of messages between Web services. WS-I members include Microsoft, BEA, IBM, RSA, and Verisign.

WS-I has developed several standards to foster interoperable, secure Web services. The standards adopt the most recent advances in XML technology, but are limited in scope to  Web Services, as opposed to more general efforts such as XACML or RuleML.

Some standards are company-proprietary and others have been submitted individually to OASIS or to the W3C. The subsections below introduce the WS-Federation, WS-Trust, WS-Security, WS-SecurityPolicy standards.

DRDC is advised to continue to track the rapid evolution of this set of specifications. Once mature, it could provide a basis for a comprehensive policy engine.

### 9.7.1 WS-Federation

WS-Federation is a (proprietary standard) XML-based markup language specifying tags and attributes for secure exchange of information across a federated network. It allows for the exchange of metadata and the use of pseudonyms. [Bajaj 2003] WS-Federation is geared to specifying identity, not specifically to addressing concerns of authorization: for this purpose, the WS-Federation anticipates the use of WS-Trust.

### 9.7.2 WS-Security

WS-Security is a formal OASIS XML standard in v1.1, since early 2006. The WS-Security [Nadalin] specification includes the Core standard implementing content integrity and confidentiality, and extension profiles for:

- username tokens;

- X.509 tokens;
- SAML tokens
- Kerberos tokens; and
- ISO/IEC 21000-5 Rights Expression Language (REL) tokens.

### 9.7.3 WS-Trust

The WS-Trust standard [Anderson 2005e] is an extension to WS-Security allowing for:

- requesting and obtaining security tokens;
- managing trusts and establishing trust relationships; and
- establishing and assessing trust relationships.

The WS-Trust model incorporates Web Services, their clients ('Requestors'), and Security Token Services, which themselves may be or request Web Services. Figure 18 illustrates this.

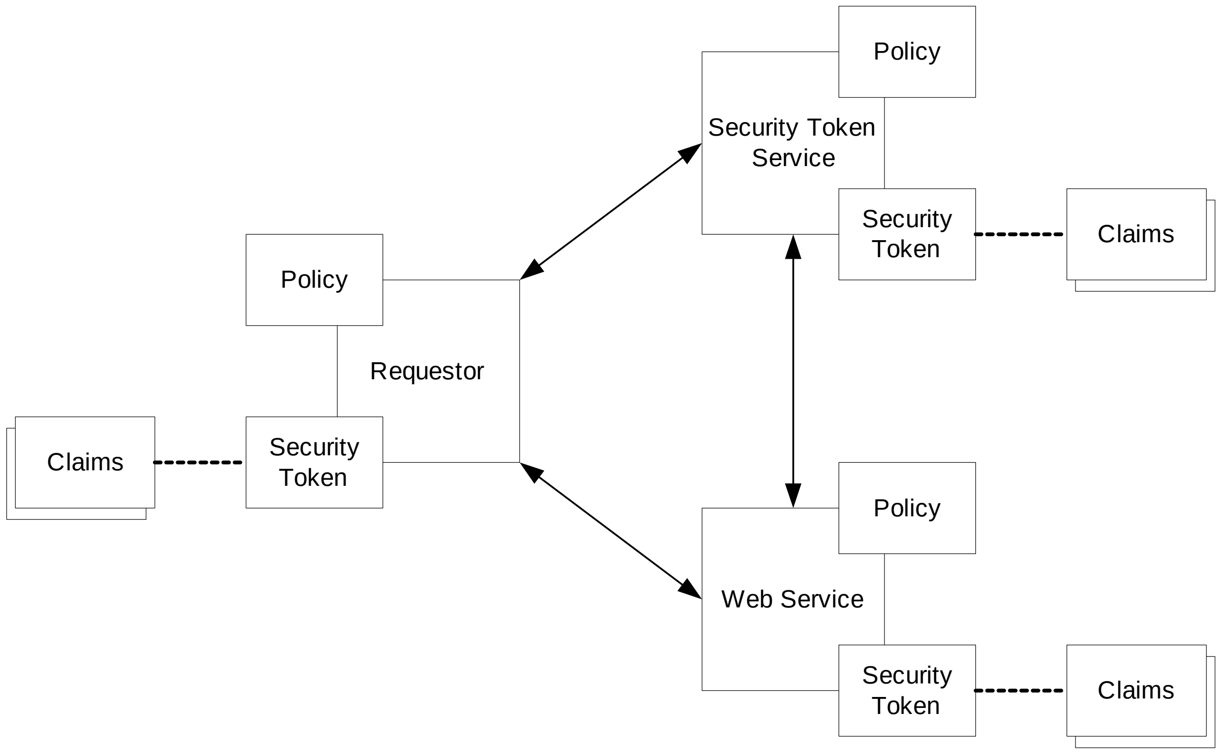

*Figure 18: WS-Trust Model (adapted from [Anderson 2005e])*

The Web Service has a policy applied to it, receives a message from a requestor that may include security tokens, and may have protection applied to it using WS-Security mechanisms. To authorize the request, the Web Service's trust engine must:

- verify the claims in the token to suffice to comply with the policy;
- verify the signatures in the token to prove the legitimacy of the attributes of the requestor, based on policy; and
- verify the issuers of the security tokens as trustworthy to issue the claims in the token, potentially by making requests from a further security token service.

WS-Security specifies cryptosecurity elements of the token. The framework uses WS-SecurityPolicy to specify security policy. The WS-Trust specification was contributed to the OASIS WS-SX Technical Committee in 2005.

### 9.7.4 WS-SecurityPolicy

WS-SecurityPolicy [Della-Libera] defines an XML framework for allowing Web Services to express constraints and requirements. The model and assumes the use of the WS-Policy specification. It allows for:

- nested policy assertions and semantics;
- policy subjects (e.g., message, operation, endpoint);
- integrity and confidentiality assertions;
- required message elements;
- token assertions (inclusion and properties of tokens protecting or binding tokens and claims to the message);
- security binding properties and assertions; and
- supporting token binding.

The model does not consider obligations stemming from positive or negative satisfaction of policy.

The WS-SecurityPolicy specification was contributed to the OASIS WS-SX Technical Committee in 2005.

### 9.7.5 WS-Policy

WS-Policy [Bajaj 2006] provides a general purpose model and extensible XML syntax to describe policies for a Web Service. It sets standards for policy expression in terms of:

- policy normalization;
- policy identification;
- compact expression; and
- policy intersection.

The framework leaves resolution of intersecting policies to domain-specific processing. The model does not consider obligations stemming from positive or negative satisfaction of policy.

WS-Policy is a draft release owned by the WS-I consortium, but submitted to the W3C in 2006.

#### 9.7.6 WS-Authorization

WS-Authorization is a pending WS-I standard to specify policy parameters for Web Services to authorize users. This specification will provide a schema to describe a

# 10. Industrial Alternatives

## 10.1 Overview

The market for industrial solutions for a policy server is quite confusing and, as a result, consumers are confronted with a wide array of products each purporting to support authorization policies and consequently policy-based access control. However, given that the overall objective is to move away from the heterogeneity of today's systems and networks, and from the variety of mechanisms implemented to handle access control and authorization, this section concentrates on industrial alternatives focused exclusively on the use of authorization policies and policy-based access control to protect sensitive information resources. As a result, this survey excludes products that embed authorization policies and policy-based access control as part of their overall product offering (e.g., databases, gateways, firewalls, web servers, application servers, security compliance tools).

Even within this relatively narrow segment of the information security market, consumers encounter a variety of technologies and terms. This was not always the case. When this market segment was in its infancy, the term PMI achieved some degree of prominence. Unfortunately, over time the term PMI became inexorably linked with PKI. Given the inherent challenges of implementing PKI, this effectively signalled the useful end of this term within industry. However, the term lives on in academia and is now inexorably linked with both attribute certificates and PKI. However, industry-acceptable variations on the term PMI have emerged to refer to various technologies, each focused on the use of authorization policies and policy-based access control to protect sensitive information resources.

This section provides a detailed examination of each of these technologies. This detailed examination will consist of a description and assessment of the technology, along with a high-level overview of the various industrial alternatives within each of the technologies. The four technologies examined include the following:

- Access Management;
- Enterprise Rights Management (ERM);
- Extrusion Prevention; and
- Policy-based Security.

## 10.2 Access Management

Access management, also referred to as extranet access management, web access management and portal security management, is a means of mediating user access primarily to web-based applications and content. Although access management is a mature security technology in its own right, it has recently been included as a core component of a larger identity and access management offering.

In many ways, the access management architecture, as seen in Figure 19, resembles that of the reference monitor model. However, in the case of the access management architecture the reference monitor is split into two distinct components; the policy server and enforcement modules.

While the policy server is also responsible for authenticating users and auditing access attempts, its primary role is mediating user access to protected content. Within the context of an access management solution, a policy server typically consists of the three following services:

- **Authentication Service –** The authentication service is capable of authenticating users through a variety of means (e.g., username & password, SAML assertions, tokens, X.509v3 certificates). It is also capable of incorporating environmental considerations (e.g., physical location, time of day, day of week) into the authentication process. Furthermore, the authentication service typically provides Single Sign-On (SSO) across web-based applications and supports federated identity to facilitate access by external users such as partners and customers;
- **Authorization Service –** The authorization service is used to provide fine-grained access to web-based resources according to the relevant security policy. The authorization service can also be used to personalize web content in order to enhance overall user experience; and
- **Audit Service –** The audit service is used to log all access attempts, whether successful or not. The log file is typically digitally signed or hashed in order to detect tampering.

Enforcement modules, which are co-located with web or application servers, intercept the access attempt, query the policy server and enforce the access response. In the case of web servers, the enforcement module is used to protect web-content (universal resource locators (URLs)). In the case of application servers, it is used to provide fine-grained access control to objects hosted on the application server (e.g., servlets, JavaServer Pages, Enterprise JavaBeans (EJB) components).

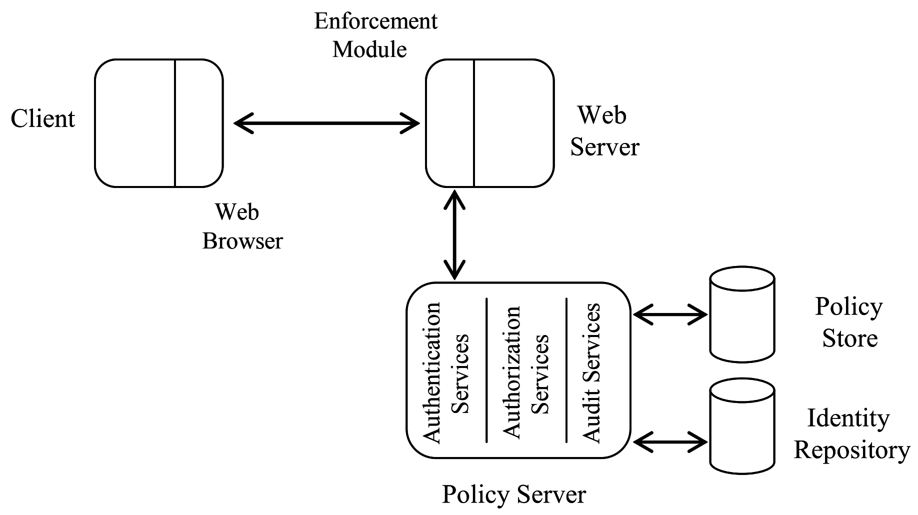

*Figure 19: Access Management Reference Architecture*

Access management solutions have achieved considerable success in securing access to both intranets and extranets. This is due to the fact that these solutions have been engineered specifically for these web-based environments and the relatively simple authorization policies supported are sufficient for most organizations' purposes. However, access management solutions are ill-suited to the role of central policy engine for all authorization policies and policy-based access control within an organization[5]. The majority of access management solutions are limited in that they only support web-based resources that can be described using a URL. As a result, non web-based resources are difficult to support and invariably require the development of custom enforcement modules. Furthermore, while access management solutions may support a programmable policy language capable of expressing a variety of policies, they tend to be severely restricted by a Graphical User Interface (GUI) that emphasizes ease of use over functionality. As a result, access management solutions typically lack a rich set of logical expressions and only support extremely limited security policies consisting of *and*, *or*, and *not* rules.

This section examines the following access management solutions:

- BMC Software Access Management;
- Computer Associates eTrust SiteMinder;
- Entegrity Solutions AssureAccess;
- Entrust GetAccess;
- Evidian Secure Access Manager (SAM);
- Hewlett Packard (HP) OpenView Select Access;
- IBM Tivoli Access Manager;
- Novell iChain;
- Oracle COREid Access;
- RSA Access Manager; and
- Sun Java System Access Manager.

### 10.2.1 BMC Software Access Management

BMC Software acquired OpenNetwork Technologies in March 2005. Access Management is a core component of the BMC Identity Management suite. Access Management supports a two-tier architecture in which the policy server component is co-located with the enforcement modules. The enforcement modules, known as BMC Enforcement Agents, reside on proxy servers or directly on web servers. Access Management uses an LDAP directory as both its identity and policy store.

Within Access Management, access to resources is determined using a combination of resources, users, and roles. Web resources, as identified by URL, are assigned authorization levels required for access. Users uniquely identify themselves to the system using one of a variety of supported credentials. Their roles, and the permissions associated with these roles, are also factored into the

[5] This was the conclusion reached during SAMPOC II, in which an Access Management solution served as the central policy engine for the prototype. The results from this POC are documented in [Magar 4a] and [Magar 4b].

authorization decision. Access Management also supports dynamic roles in which roles are dynamically associated with the user based on the user's profile at the time of access.

### 10.2.2 Computer Associates eTrust SiteMinder

Computer Associates acquired Netegrity in November 2004. eTrust SiteMinder is an access management product consisting of two primary components: a policy server component, the eTrust SiteMinder Policy Server, and enforcement modules, eTrust SiteMinder Agents. The Policy Server runs four primary services: authentication, authorization, administration and auditing. Agents, which enforce policy-based authentication and access control, include web agents, application server agents, SAML affiliate agents, enterprise application agents and custom agents.

The Policy Server provides fine grain access control over resources (e.g., file, page or object level) through a combination of security policies and eTelligent Rules. Security policies apply to protected resources and specify who (e.g., user, group, role) has access to the resource and under what conditions. The security policy also specifies what happens if a user is denied access to a resource. eTelligent Rules extend security policies by incorporating dynamic data such as time, IP address, or API calls into access decisions.

### 10.2.3 Entegrity Solutions AssureAccess

Entegrity AssureAccess is an access management solution that allows application developers to incorporate authentication, authorization, policy administration, audit and SSO into their web-based applications. It consists of four primary components: Application Server Adapters, Audit Server, Authentication Server and Management Server. The Application Server Adapters, which are available for web and J2EE servers, are used for authentication and authorization. For authorization the adapters serve to both mediate and enforce policy. The Audit Server collects audit events from other AssureAccess components and saves them to disk. The Authentication Server uses attribute certificates, containing user attribute information from authoritative sources, for SSO and authorization decisions. The Management Server provides administrators with a web interface with which to manage AssureAccess components.

Dynamic policies are used to control the four key processes within AssureAccess; authentication, authorization, audit, and administration. Policies consist of rules which can be combined using logical operators, can be negated, and can be evaluated in strict order. Policies can also be used to test for user attributes, the value of an attribute, the time and date, the IP address, the strength of authentication, and the level of encryption. Not only does AssureAccess support custom rules, and consequently custom policies, but it supports the ability to use policies within rules.

### 10.2.4 Entrust GetAccess

Entrust acquired enCommerce in May 2000. GetAccess, which is the core component of the Entrust Secure Web Portal Solution, consists of the GetAccess Server and the GetAccess Runtime Service. The GetAccess Server consists of a number of key services (e.g., Access Service, Identification Service, Logging Service), including the Entitlements Service. The Entitlements Service is a CORBA service that mediates access attempts according to policy. The Runtime

Service is an enforcement module for web servers that intercepts incoming access requests and redirects them to the Entitlements Service for mediation.

The current version of GetAccess, version 7.0, includes support for SAML and XACML. SAML is used within GetAccess to facilitate identity federation and to exchange authentication and authorization information between the Runtime Service and the GetAccess Server. XACML is used within GetAccess to provide rules- and roles-based access control. When a user attempts to access a protected resource, GetAccess first determines the roles associated with the user and then processes the rules associated with the protected resource. Entrust GetAccess is capable of supporting rules that take into account the authentication method used, the time and date, and IP addresses. GetAccess also includes an open callback mechanism to facilitate support for custom policies.

### 10.2.5 Evidian SAM

Evidian, formerly the BullSoft software division, became a separate company under Bull majority ownership in June 2000. SAM consists of three separate products which together comprise the access management component of Evidian's identity and access management suite; AccessMaster NG (New Generation). The three separate products are SAM-Standard Edition, SAM-Web Edition, and SAM-J2EE. They provide authentication and authorization services respectively for multiple client environments, web applications, and multi-tier environments. Each of the products consists of enforcement modules and a central policy server.

SAM provides fine-grained authorization, in which access to applications and resources can be controlled based on specific criteria such as an amount, the type of customer or the type of access.

### 10.2.6 HP OpenView Select Access

HP acquired SelectAccess from Baltimore Technologies in July 2003. Baltimore's SelectAccess was in turn acquired by Baltimore in 2000 from Nevex, a Toronto-based security company. SelectAccess, which is a core component of the HP OpenView identity management suite, consists of Policy Builder, Policy Validator, and Enforcer plug-ins. Policy Builder is the interface through which administrators define authorization policies. Policy Validator, which is HP's equivalent of the policy server component, evaluates user access requests using policy information stored in the Policy Store (directory). Enforcer plug-ins, which are enforcement modules in the SelectAccess architecture, are located on web/application servers and are used to enforce access decisions.

Select Access uses a Policy Matrix to apply access policies to all possible user/group and resource combinations. For each combination the access policy has one of three values: deny, allow, or conditional. While the first two are self-explanatory, conditional means that a user is permitted to access the resource provided he satisfies certain criteria, as defined in rules. Rules can be used to evaluate the user's network address or domain name, the time of day, the user's encryption level, the user's attributes, the user's authentication credentials, and the port the user is attempting to access.

### 10.2.7 IBM Tivoli Access Manager

IBM acquired Dascom in September of 1999. IBM Tivoli Access Manager is actually three separate products consisting of a central policy server component and an enforcement module or resource manager. The three products are Access Manager for e-Business (formerly Tivoli Policy Director), Access Manager for Business Integration, and Access Manager for Operating Systems. Access Manager for e-Business is designed to manage authorization policies for web-based resources. Access Manager for Business Integration is used to define security policies for IBM WebSphere MQ (Message Queuing) queues and messages in these queues. Access Manager for Operating Systems can be used to define access control policies for a range of Unix-based operating systems (e.g., Solaris, HP-UX, Linux).

Security policies within Access Manager are defined through the use of ACLs, Protected Object Policies (POPs) and authorization rules. Authorization rules are specified in eXtensible Style Language (XSL). For situations where Access Manager is incapable of expressing all of the conditions required by an organization's security policy, Access Manager provides an external authorization capability.

### 10.2.8 Novell iChain

Novell iChain is an access management solution consisting of the iChain Proxy Server, iChain Authorization Server, and Novell Modular Authentication Server (NMAS) RADIUS Server. Unlike most access management solutions that utilize enforcement modules, iChain uses a reverse proxy server.

Access to resources is controlled by setting up static or dynamic access control rules. Dynamic access control rules are capable of querying user attributes.

### 10.2.9 Oracle COREid Access

Oracle acquired Oblix in March of 2005. Oracle COREid Access consists of Access Manager, Access Server, WebGates and AccessGates. Access Manager is a graphical tool with which to administrate policies, assign policies to resources and test policy functionality. Access Server provides authentication, authorization, and auditing services in order to protect web and non-web resources. WebGates are basically web server enforcement modules, while AccessGates are custom policy enforcement modules built from an Access SDK.

COREid Access provides policy-based authorization for web and J2EE resources. It supports security policies that can restrict access to specific resources based on user, role, group membership, time of day, day of week, and IP address. COREid Access also supports custom authorization plug-ins in order to incorporate authorization logic not supported by the product. These custom authorization plug-ins can be built using the Authorization API provided.

### 10.2.10 RSA Access Manager

RSA Security acquired Securant in September of 2001. RSA Access Manager (formerly ClearTrust) is comprised of a central server and agents co-located with protected resources. The central server consists of three components; the dispatcher, the authorization service and the entitlements service. The dispatcher serves to maintain session keys for communications between components. The authorization service mediates the access attempt and the entitlements service retrieves information from the identity store.

RSA Access Manager rules consist of static attributes, dynamic attributes, and boolean constructs. Rules that incorporate dynamic user attributes are referred to as Smart Rules within RSA Access Manager.

### 10.2.11 Sun Java System Access Manager

Sun Java System Access Manager is a Java-based access management solution consisting of a policy server (Access Manager Server) and enforcement modules (Access Manager Policy Agents). The Access Manager Server is comprised of a number of services, including the Authentication Service, User Session Management, SAML Service, Identity Federation Service, Logging, and Policy Service.

The Policy Service supports normal policies and referral policies. Normal policies consist of policy rules, subjects, conditions, and response providers. Policy rules specify a resource, an action (or several actions), and the corresponding permission (e.g., allow or deny). Subject plug-ins supported by Access Manager include Access Manager Identity Subject, Access Manager Roles, Authenticated Users, LDAP Groups, LDAP Roles, LDAP Users, Organization Web, and Services Clients. Condition plug-ins supported by Access Manager include Authentication Level, Authentication Scheme, IP Address, LE (Less than or Equal to) Authentication Level, Session, Session Property, and Time. Response providers are used to personalize application pages. Referral policies enable administrators to delegate policy creation to a different Access Manager Server or a third-party product. Access Manager supports the development of customized plug-ins for subjects, conditions, response providers, and referrals.

## 10.3 Enterprise Rights Management (ERM)

DRM is a technology for controlling the distribution of digital media (e.g., e-books, movies, music, software). It originated in the mid-1990s in response to commercial piracy of digital media and is used by both the entertainment industry and software publishers to ensure that only authorized users (i.e., paying customers) have access to digital content. In addition, DRM can be used to control how the digital content can be used. The term DRM now encompasses a wide range of technologies for protecting digital content. These technologies include products that generate end-user licence keys, hardware-based licensing and protection schemes, encryption and key revocation systems that prevent the copying of optical discs and DVDs, etc. DRM technologies of note include Apple Fairplay, Macrovision (eMeta) eRights Suite, Microsoft Windows Media DRM, RealNetworks Helix DRM, and Sony’s infamous XCP Content Protection Software.

ERM, sometimes referred to as Enterprise DRM, uses DRM technology and principles but applies them to information resources such as documents and e-mail. ERM emerged as a distinct market segment in approximately 2003. The primary differences that distinguish ERM from DRM are as follows:

- **Digital Content –** As mentioned previously, DRM is applied to digital media such as e-books, movies, music and software, while ERM is applied to information resources such as documents and email;
- **Authentication –** ERM must be capable of supporting the wide variety of authentication techniques found in most organizations. In contrast, DRM need only support the relatively limited number of consumer authentication mechanisms; and
- **Authorization Policies –** DRM must support flexible authorization policies capable of representing a variety of usage models (e.g., one-time pay-per-view, periodic access, timed licence, permanent licence). In contrast, ERM is more concerned with being able to modify or revoke authorization policies even after the information resource has been distributed.

The ERM architecture, which can be seen in Figure 20, consists of three primary components:

- **Content Provider –** ERM-protected content consists of the encrypted information resource and a publishing licence. The content provider, which can be either an information server (e.g., file server, web server) or an actual person, is responsible for distributing the ERM-protected content to the client;
- **Rights Management Plug-in –** When the client attempts to perform some function involving ERM-protected content, the rights management plug-in sends the client's credentials and the publishing licence to the licence server. The licence server validates the request and issues a use licence. The rights management plug-in enforces the rights as specified in the use licence. The use licence, which includes the decryption key, ultimately determines what the client is allowed to do with the ERM-protected content; and
- **Licence Server –** In addition to validating use licence requests, the licence server is responsible for signing publishing licences and storing decryption keys for ERM-protected content.

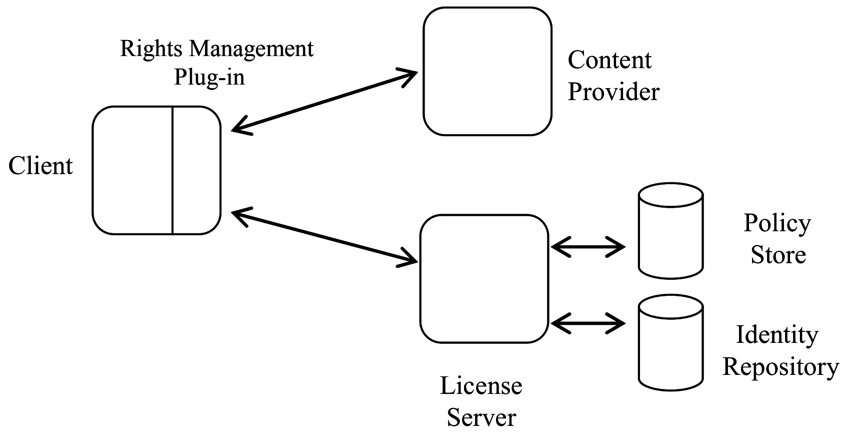

*Figure 20: ERM Reference Architecture*

While DRM solutions are in widespread use in consumer applications, ERM solutions have yet to achieve the same level of adoption. This may be due in part to the fact that ERM solutions are a relatively new technology that is in the process of maturing. However, it may also be the result of the impact that this technology has on an organization's IT infrastructure. ERM solutions typically necessitate the deployment of a client-side rights management plug-in to enable users to consume rights-protected content. Likewise, a number of ERM solutions force the organization to adopt a redundant cryptographic infrastructure rather than leveraging the organization's existing one[6]. In any case, ERM solutions are ill-suited to the role of central policy engines for all authorization policies and policy-based access control within an organization. Although ERM solutions can support a wide range of information resources, they are severely limited in terms of policy expression by both the GUI and the policy language. Not only does the GUI provide limited access to policy constructs but the policy languages used in ERM solutions tend to be better suited for expressing rights and conditions rather than complex policies.

This section examines the following ERM solutions:

- Adobe LiveCycle Policy Server;
- Avoco Secure secure2trust;
- EMC Corporation (Authentica) Active Rights Management Platform;
- Liquid Machines Document Control/Email Control;
- LockLizard Protector/Safeguard;
- Microsoft Windows Rights Management Services; and
- SealedMedia.

### 10.3.1 Adobe LiveCycle Policy Server

Adobe acquired the FileLine DRM division of Navisware in January 2006. This acquisition has expanded the scope of Adobe LiveCycle Policy Server to encompass more than just Adobe Acrobat documents. Adobe LiveCycle Policy Server is now capable of supporting Microsoft Office documents and Computer Aided Design (CAD) files.

Adobe LiveCycle Policy Server enables content creators to specify through policy who has access to protected content, the level of encryption used, whether printing is permitted, whether a user can make changes, and the expiration date of the document. Furthermore, changes to policy can be made dynamically. In other words, policies can be updated long after a protected document has been distributed.

### 10.3.2 Avoco Secure secure2trust

Avoco Secure secure2trust is capable of protecting Adobe, CAD, Microsoft Office (including the new Microsoft Office Open XML format), Microsoft Project, Microsoft Visio and XML documents.

Security policies can be assigned to documents through the use of policy templates based on typical classifications. These policy templates dictate what recipients are capable of doing with

[6] This was the conclusion reached during SAMPOC II, in which the ERM solution was not capable of leveraging the existing PKI. The results from this POC are documented in [Magar 4a] and [Magar 4b].

the document in terms of opening the document, editing the document, and using third party screen capture.

### 10.3.3 EMC Corporation (Authentica) Active Rights Management Platform

EMC Corporation acquired Authentica in February 2006. The Active Rights Management Platform is an ERM solution that can be used to protect Microsoft Outlook, Lotus Notes, and BlackBerry email messages, as well as Microsoft Office and Adobe Acrobat documents. The solution consists of two primary products, Secure Mail and Secure Documents. Secure Mail consists of the Authentica Policy Server and a desktop email plug-in for Microsoft Outlook, Lotus Notes, or BlackBerry devices. Together they serve to control user actions, such as reading, printing, copying and forwarding email messages, through policy. Secure Documents comprises the Authentica Policy Server and a desktop plug-in for Adobe Acrobat or Microsoft Office. Not only do they serve to control what the user can do with the document, but they provide continuous tracking and auditing of document activity, and policy-based watermarking.

Policies are typically applied to sensitive information through policy templates Furthermore, Authentica Active Rights Management Platform supports dynamic policy controls thereby providing control over the document long after it has been distributed.

### 10.3.4 Liquid Machines Document Control / Email Control

Liquid Machines is an ERM solution that is capable of protecting content across more than 65 applications and file formats including Microsoft Office, Visio, Sharepoint, and Adobe Acrobat. Furthermore, through Liquid Machines for Microsoft RMS, the software is capable of RMS-enabling additional applications and formats not supported by Microsoft. The two primary products in the Liquid Machines suite are Document Control and Email Control.

Policies consist of administrator defined roles and rights such as read, write, print, assign and transfer. Although users can assign preconfigured policies to documents, they cannot create custom policies.

### 10.3.5 LockLizard Protector / Safeguard

LockLizard is an ERM solution consisting of Lizard Protector and Lizard Safeguard. Lizard Protector is used to protect web content, while Lizard Safeguard is used to protect Adobe Acrobat documents. Of interest is the fact that Lizard Safeguard protects Adobe Acrobat documents without the use of an Adobe plug-in, which it considers insecure.

Lizard Protector and Lizard Safeguard control respectively who can view (and for how long), print, save, and copy Web content and Adobe Acrobat documents. Furthermore, Lizard Safeguard supports dynamic revocation of Adobe Acrobat documents.

#### 10.3.6 Microsoft Windows Rights Management Services

Microsoft Windows RMS is an ERM solution for protecting Microsoft Office documents and HTML content. However, APIs are provided that can be used to extend this functionality to other applications and file formats.

When assigning policy to sensitive information, users can either choose to create a custom policy or utilize a policy template created by the system administrator. In either case, the policy information is stored in XrML-based licences.

#### 10.3.7 SealedMedia

SealedMedia is an ERM solution capable of supporting a wide-range of applications and file formats. Supported applications include Microsoft Office, Microsoft Outlook, Lotus Notes, Novell GroupWise, BlackBerry, Adobe Acrobat, Microsoft Internet Explorer, Apple QuickTime, and Informative Graphics Brava! Reader.

Policies can be used to control document sealing, opening, searching, printing, editing, change tracking, copying, reclassifying, screen capturing, annotating, measuring, and sealing. Furthermore, SealedMedia supports dynamic policy control in order to effect post-delivery policy changes.

## 10.4 Extrusion Prevention

Extrusion Prevention is a relatively new technology targeted at preventing sensitive information from leaving an organization in an unauthorized manner. It accomplishes this in one of two ways: by discovering and protecting sensitive information located on servers and desktops within the organization, or by monitoring outbound network communications for sensitive information. The first approach is referred to as host-based extrusion prevention, while the second approach is referred to as network-based extrusion prevention. Extrusion prevention technology is also referred to by any number of combinations of the words content, monitoring and filtering (e.g., content monitoring & filtering, Internet content filtering, web content monitoring, e-mail content security management, content monitoring, web content security, content filtering, Internet content monitoring, web content filtering, and email content security management), as well as secure content management, information leak detection and prevention, and content protection for the network.

The extrusion prevention architecture, as

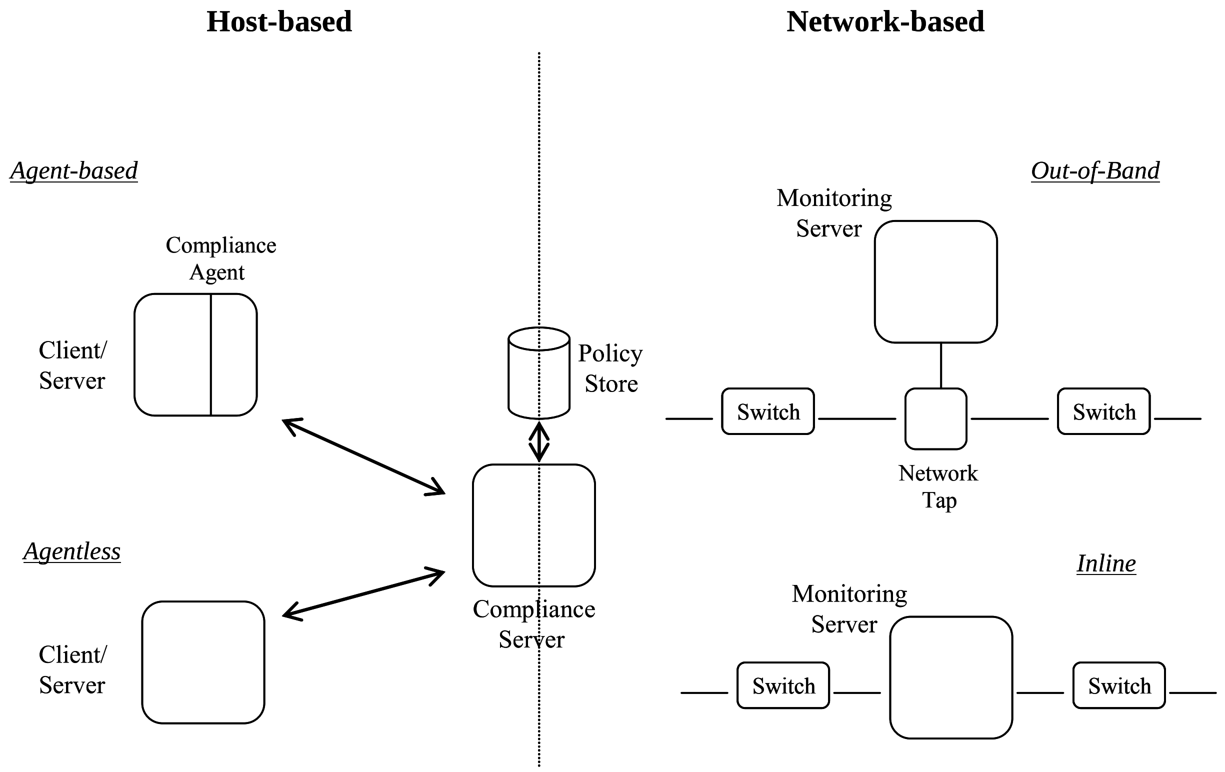

Figure 21 illustrates, depicts both host-based and network-based solutions. It consists of three primary components:

- **Compliance Server –** The Compliance Server is responsible for managing the extrusion prevention infrastructure, including compliance agents and monitoring servers. This includes pushing policy out to these end points;
- **Compliance Agent –** Compliance agents are located on a local host, such as a client desktop or server, and prevent users from performing unauthorized actions on sensitive information. For example, a compliance agent may prevent a user from printing a sensitive document or copying it to a USB device; and
- **Monitoring Server –** The Monitoring Server monitors network communications, specifically multiple channels (e.g., email, instant messaging, FTP, HTTP), for sensitive information. Content is analyzed using linguistic or statistical analysis such as keyword matching, document fingerprinting, Bayesian analysis, and machine learning. The Monitoring Server can either be located out-of-band, in which case it performs a monitoring function only, or inline, in which case it is also capable of preventing information leaks.

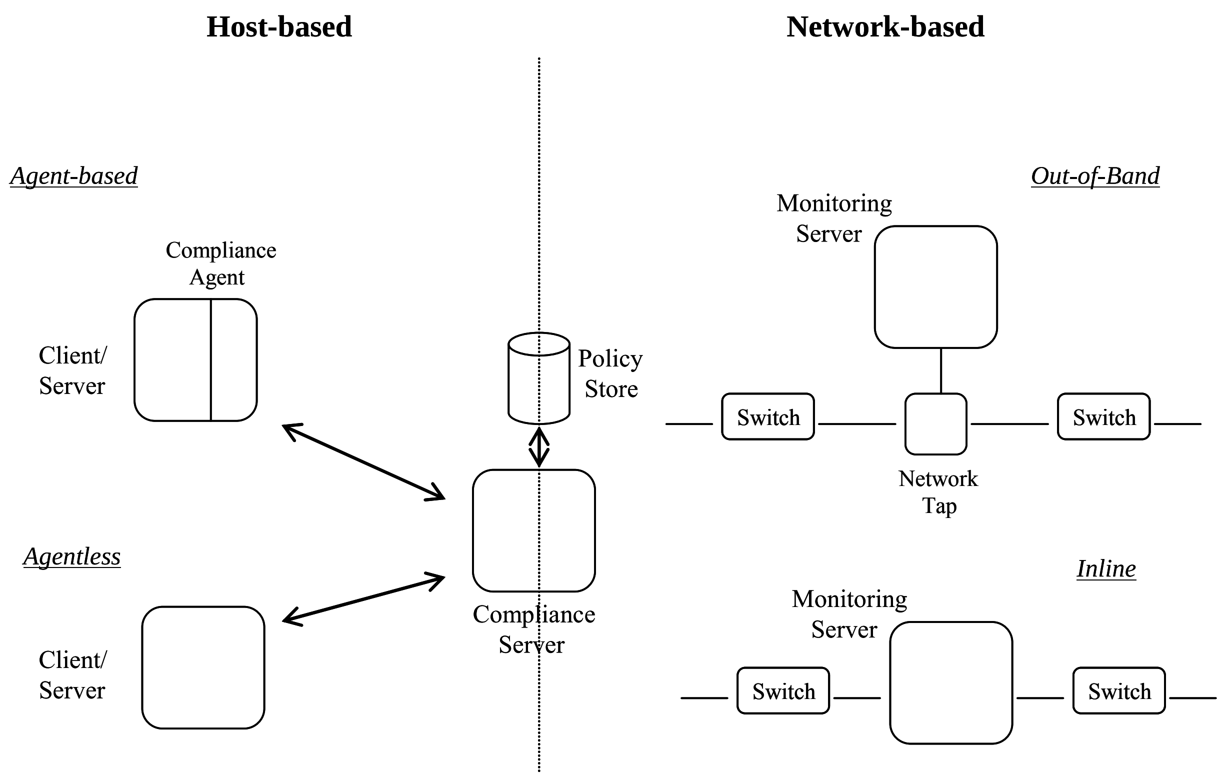

*Figure 21: Extrusion Prevention Reference Architecture*

Extrusion Prevention solutions, both host-based and network-based, are a relatively recent phenomenon (although of the two, network-based solutions are the more mature technology). Consequently, they have not achieved widespread adoption in most organizations. However, extrusion prevention solutions are ill-suited to the role of central policy engines for all authorization policies and policy-based access control within an organization. These solutions tend to ship with policies pre-developed to address particular privacy regulations or information security challenges. While these policies can be customized to some degree, the customization tends to be limited by the functionality accessible through the limited GUI. Furthermore, extrusion prevention solutions do not usually support a full-featured, programmable policy language capable of articulating the complex policies required for the defence environment.

This section examines the following Extrusion Prevention solutions:

- Entrust Entelligence Content Control Server;
- Fidelis Security Systems' DataSafe;
- Intrusion Compliance Commander;
- Oakley Networks' SureView;
- Orchestria Active Policy Management Platform;
- Palisade Systems' PacketSure;
- PortAuthority Technologies;

- Proofpoint Protection Server;
- Reconnex inSight Platform;
- Tablus Solution Suite;
- Verdasys Digital Guardian; and
- Vontu.

### 10.4.1 Entrust Entelligence Content Control Server

Entrust Entelligence Content Control Server is a network-based extrusion prevention solution capable of monitoring a variety of communications protocols (e.g., email, instant messaging, HTTP and FTP) for sensitive information. Not only does the product block traffic that violates policy but it provides tools that automatically educate users committing the violations. Rather than matching words encoded in rule bases, the Content Control Server uses statistical and linguistic algorithms to perform content analysis.

### 10.4.2 Fidelis Security Systems' DataSafe

Fidelis Security Systems' DataSafe is a port and protocol independent network-based extrusion prevention solution. It can be deployed either inline, with all network traffic flowing through the product, or out-of-band via a network tap. Both architectures allow DataSafe to enforce policy on all traffic flowing on the network without necessitating changes to the network or the installation of software at the endpoints. DataSafe consists of the Network Channels Controller, the Enterprise Content Controller and CommandPost. The Network Channels Controller is responsible for ensuring that all traffic on the network flows in approved channels and terminating any session that violates policy. The Enterprise Content Controller is responsible for preventing the unauthorized transfer of sensitive digital assets. It is also capable of terminating network sessions that violate policy. The CommandPost is a web-based interface with which to administer DataSafe.

In addition to providing tools to create custom policies, DataSafe ships with pre-built policies designed to comply with specific government regulations (e.g., HIPAA, DoD Data Classification). These pre-built policies make use of information profiles in order to prevent the unauthorized transfer of sensitive information.

### 10.4.3 Intrusion Compliance Commander

Intrusion Compliance Commander is a network-based extrusion prevention solution. It can be deployed either inline, so that sensitive data must pass through it prior to leaving the organization, or out-of-band via a network tap. In the latter case, Compliance Commander is used for monitoring purposes only. Compliance Commander consists of Sentry, D3 and Provider. Sentry is a required component used to monitor network traffic and enforce policy. It comes in both a software version and a hardware appliance. D3 is a required component that serves as middleware for uploading the required data records from the database to the Sentry appliance. Provider is an optional component that facilitates the management of larger deployments.

### 10.4.4 Oakley Networks’ SureView

Oakley Networks’ SureView is a host-based extrusion prevention solution that uses compliance agents on the user’s workstation to protect sensitive information. The compliance agents on the user’s workstation are capable of monitoring a variety of applications including email, instant messaging, Microsoft Office, clipboard, file system, USB storage, CD and DVD writers, and output to printers.

SureView is capable of creating custom policies specifically tailored to an organization’s security requirements through its Policy Builder component. It also ships with predefined industry and government policies.

### 10.4.5 Orchestria Active Policy Management Platform

The Orchestria APM Platform is a host-based extrusion prevention solution for corporate messaging channels. It consists of Real-Time Prevention, Intelligent Review and Smart Tagging. Real-Time Prevention analyzes messages before they are sent and blocks messages that violate policy. Intelligent Review analyzes messages for compliance after they have been sent and provides workflow to address messages that violate policy. Smart Tagging provides a means to categorize messages prior to the messages being archived.

Security practitioners have the option of custom developing policies or using pre-built policies called PolicyPaks. PolicyPaks are groups of related policies that address specific areas of regulation. In any case, policies are defined based on context, content, and concepts. Context refers to information about a message (e.g., date and time, sender, recipient); content refers to the message contents; concepts define the type of message (e.g., research report, financial model). A configurable action can then be assigned to the policy. Configurable actions include Block, Warn, SmartQuarantine, Inform, Copy, Send, Procedural Support, or Silent with capture.

### 10.4.6 Palisade Systems’ PacketSure

Palisade Systems’ PacketSure is a network-based extrusion prevention solution. PacketSure includes Content Surety and TrendReporter. Content Surety analyzes file contents for specific data such as social security numbers, credit card numbers, etc. TrendReporter monitors network traffic in order to plot network application usage and consequently provide the information required to develop rules to better manage network traffic.

PacketSure can be configured to monitor, block or allow protocols based on more than 140 signature-based rules. These rules can be customized or custom rules can be created from scratch.

### 10.4.7 PortAuthority Technologies

PortAuthority is a network-based extrusion prevention solution that prevents sensitive information from leaking out through mail, web, or networked printing. It consists of two appliances; PortAuthority Enterprise Manager and PortAuthority Monitor. The PortAuthority Enterprise Manager is used to manage multiple Monitor appliances. It is typically installed inside the network, and includes a reporting module and a policy wizard to facilitate the creation of custom policies. The PortAuthority Monitor is used to monitor multiple protocols for the unauthorized transmission of sensitive information. The PortAuthority Monitor is typically deployed out-of-band via a network tap.

### 10.4.8 Proofpoint Protection Server

Proofpoint Protection Server is a network-based extrusion prevention solution that provides a modular architecture to combat spam and viruses, and to ensure regulatory compliance and digital asset security. Key components of its modular architecture include Proofpoint Content Compliance, Proofpoint Digital Asset Security, and Proofpoint Regulatory Compliance. Proofpoint Content Compliance allows organizations to define and enforce acceptable use policies for email content and attachments. Policies can be defined with the help of preconfigured filters that dictate maximum message size, allowable attachment types, maximum number of recipients, and maximum number of attachments. Proofpoint Digital Asset Security analyzes and classifies sensitive information (and non-sensitive information) and then monitors messaging traffic to ensure that this information does not leave the organization. Policies, which can be linked to specific document types or a document similarity score, dictate how sensitive information found leaving the organization should be handled. Options include quarantine, reject, annotate, redirect, reply to sender, discard, etc. Proofpoint Regulatory Compliance ensures that outbound messages comply with privacy regulations. While this module comes with a number of privacy rules defined, these rules can be modified through a point and click interface.

### 10.4.9 Reconnex inSight Platform

Reconnex inSight Platform is a network-based extrusion prevention solution that enables organizations to register sensitive content by emailing or uploading it to the central server or by automatically registering sensitive information stored on file servers or in databases. With Reconnex inSight Platform administrators can define policies, such as setting expiration dates, creating sensitivity levels and defining acceptable transmission methods, to further protect this sensitive information. Reconnex inSight Platform consists of the Reconnex iGuard, Reconnex iController, Reconnex iManager and Reconnex eRisk Modules. Reconnex iGuard monitors, captures and stores all outbound and inbound network content. The stored content can then be analyzed for threats and threat correlation. Reconnex iController is responsible for registering sensitive information within an organization. The iController creates a unique digital fingerprint, consisting of thousands of overlapping digital signatures, for each piece of sensitive information. Reconnex iManager is the administrative interface for managing the Reconnex inSight Platform. Reconnex eRisk Module(s) are industry specific policy engines consisting of pre-defined rules and reporting templates. There are four types of eRisk Modules: Compliance eRisk Module, Competitive Advantage eRisk Module, Corporate Governance eRisk Module, and Critical Information Security eRisk Module. The Compliance eRisk Module enables compliance monitoring and reporting of various privacy regulation violations. The Competitive Advantage eRisk Module defines policy for the protection of intellectual property. The Corporate Governance eRisk Module defines policy for workplace safety issues and appropriate use of corporate resources. The Critical Information Security eRisk Module defines policy for safeguarding sensitive resources against insider threats and social engineering.

### 10.4.10 Tablus Solution Suite

The Tablus solution suite is a network-based and host-based extrusion prevention solution that allows organizations to protect customer and business information from insider threats. The solution suite consists of the Content Sentinel, Content Alarm NW and Content Alarm DT. Content Sentinel scans an organization's infrastructure for sensitive information stored in insecure locations. Sensitive information thus identified is quarantined, deleted, encrypted or moved to a secure server. Content Alarm NW monitors network traffic leaving an organization for sensitive information. Sensitive information found leaving the organization is quarantined or blocked. Content Alarm DT is the desktop policy enforcement component that prevents unauthorized handling of sensitive information. This includes preventing sensitive information from leaving the organization through email, printing, or removable media.

### 10.4.11 Verdasys Digital Guardian

Verdasys Digital Guardian is a host-based extrusion prevention solution that espouses a philosophy of protecting sensitive information at the point of use through the use of distributed agents. These agents ensure that end-user use of applications and data is in compliance with the organization's security policy. Components of the Digital Guardian solution include the Digital Guardian Server, Agents, Management Console, and Solution Paks. The Digital Guardian Server is responsible for deploying and monitoring the agents distributed throughout the organization. The Digital Guardian Agents are compliance agents that are deployed to all desktops, laptops, and servers within an organization. The remote device agents operate at the kernel level of the operating system in order to monitor, log, and mediate (according to pre-defined rules) actions by end-entities. The Digital Guardian Management Console is a web-based interface to the Digital Guardian Server. It is through this interface that the management of agent installation, security settings, rule definition, reporting and auditing is accomplished. The Digital Guardian Solution Paks are policies, consisting of pre-written rules, that have been developed to address a particular regulation or information security challenge. They include Consumer Data Protection, Intellectual Property Protection, Outsourced Data Protection and Compliance/Privacy Assurance.

### 10.4.12 Vontu

Vontu is a host and network-based extrusion prevention solution emphasizing data loss prevention. Vontu is an agentless two-tier architecture consisting of the following components: Discover, Protect, Monitor, Prevent and Enforce. Vontu Discover scans servers and desktops within an organization for unsecured sensitive information. Vontu Protect enforces access control and encryption policies in order to secure sensitive information located on servers and desktops within the organization. Vontu Monitor scans network communications for sensitive information. Vontu Prevent blocks, quarantines or reroutes network communications containing sensitive information. Vontu Enforce is the policy management platform for the other Vontu components. It provides 70 pre-built compliance templates but also supports custom policies that dictate response rules, automatic notification, remediation, blocking, quarantine, and encryption.

## 10.5 Policy-based Security

Unlike Access Management, ERM and Extrusion Prevention, Policy-based Security is not a recognized market segment. Policy-based Security protects sensitive information, regardless of whether it is stored locally or on a server, and mediates access according to policy. In actuality, sensitive information is encrypted using symmetric cryptography. The symmetric keys are encrypted by the policy server using its public encryption key. Once access has been approved, the policy server decrypts the symmetric key using its private decryption key and re-encrypts it using the user's public encryption key. The user is then able to decrypt the document using his/her private decryption key and consequently open the document.

While there are a number of similarities with ERM, there are two key differences that differentiate Policy-based Security from ERM. First, Policy-based Security solutions tend to use more full-featured policy languages that are capable of supporting more complex policies. ERM solutions, in contrast, make use of rights expression languages that are quite constrained in terms of articulating policy. Second, while both solutions use cryptography to protect sensitive information, ERM goes one step further by enforcing proper handling as well.

This section examines the following Policy-based Security solution:

- Entrust Authority Key Release Server.

### 10.5.1 Entrust Authority Key Release Server

Entrust Authority Key Release Server (KRS) protects sensitive information within an organization using a combination of encryption and policy-based access control. Sensitive information is encrypted using the Key Release Server's public encryption key. When a user attempts to access this protected information, the Key Release Server mediates the access attempt and, provided that access is permitted, releases the appropriate decryption key to the user.

Entrust Authority Key Release Server, which is illustrated in Figure 22, consists of the following components:

- **Key Release Server –** Unlike traditional policy architectures, the Key Release Server hosts the following policy components:
    - **PEP –** The PEP intercepts key release requests, formulates policy decision requests for the PDP, and enforces the PDP's response;
    - **PDP –** The PDP mediates the access attempt according to the relevant policy and returns the response to the PEP;
    - **PIP –** The PIP provides the mechanism for retrieving identity and object attributes from external repositories such as directories and databases; and
- **KRS-aware Client –** The KRS-aware client is not an actual product. It is a custom developed application, built using the KRS Toolkit, that is capable of interacting with the Key Release Server.

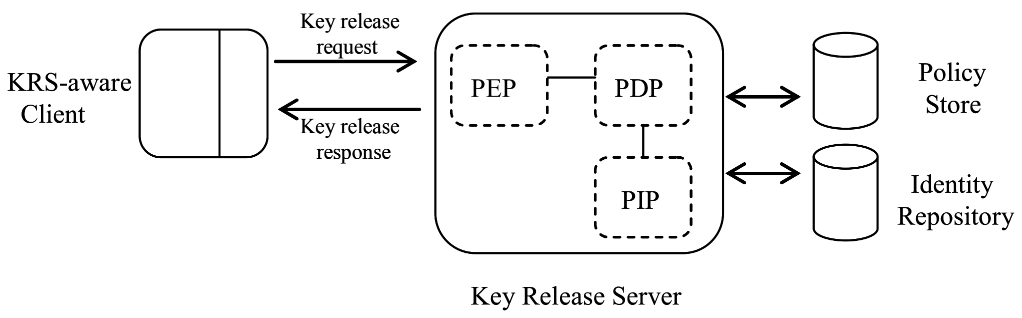

*Figure 22: Key Release Server Architecture*[7]

Key Release Server security policies, and consequently policy rules, are created using XACML 1.1. In fact the Key Release Server supports all mandatory features of XACML 1.0 and 1.1, along with a number of optional schema elements, identifiers, and features. To mediate an access attempt the PDP requires a number of inputs for the policy. These include the subject(s) (along with subject attributes such as role, location, title, etc.), the resource(s) (along with resource attributes such as sensitivity, caveats, etc.), the action(s), and the environmental variables (geographical location, time of day, etc.). Once the access attempt has been mediated, the PDP returns a response to the PEP. Valid responses include permit, deny, not applicable (relevant policy cannot be found), and indeterminate (error condition). Both not applicable and indeterminate responses are treated the same as a deny response in terms of denying the key release request. The response from the PDP will consist of the decision as well as any obligations. Obligations can be used to further constrain what the user can do with the information in the event of a permit decision.

Unlike many industrial alternatives the Key Release Server does not seem to be constrained by the PAP GUI interface. The primary reason is that it does not have one. Rather, rules, and consequently policies, are defined using an XML file. Given that the Key Release Server is 100% compliant with XACML 1.0 and 1.1, its ability to support the complex policies required by the defence environment can be considered the equivalent to XACML (refer to Section 8.2.3). Unfortunately, the Key Release Server has a number of limitations. The first is that although the KRS Toolkit can be used to develop KRS-aware clients, there are currently no existing products that provide this functionality. Second, the KRS does not seem to be actively developed within Entrust at this time. This raises some doubts as to its long-term viability.

[7] Figure 22 is based on the figure on page 16 of [Entrust 2005]

# 11. The “Design and Implement” Alternative

## 11.1 Overview

The alternatives available under a design and develop scenario are:

9. utilize an existing Open Source policy language as the basis for a DRDC-specific policy engine; or
10. create a policy engine from scratch based on existing research and an understanding of what constitutes a properly defined and implemented policy language and engine.

Under the first alternative there are only a limited number of existing policy languages that can be utilized, most notably Ponder. Under the latter alternative, all existing research and available policy languages can be used as a base to determine the workings of a policy engine and the issues to avoid in its development.

The remainder of this section covers what is required in order to develop a custom policy solution relative to each alternative. Specifically, it will address the following aspects of the “design and implement” alternative:

- Architectural Model;
- Policy Language;
- Policy Mediation;
- Virtualized Domain;
- Policy Store;
- Audit Log;
- Management Console;
- Application Programming Interface (API) & 3rd Party Interoperability; and
- Summary.

## 11.2 Architectural Model

The simplest model and the most prevalent for policy engines is the classic Reference Monitor Model. Figure 23 illustrates this classic model. As can be seen the Reference Monitor is situated between the subject and the resource being accessed. In most systems the Security Policy is embedded within the Reference Monitor, as it is in most modern operating systems and applications. In a policy engine the security policy is stored externally and the reference monitor becomes more of an interpreter capable of loading a policy, passing information about the subject and resource to the policy, executing the policy, and then enforcing the response, typically either an allow or deny.

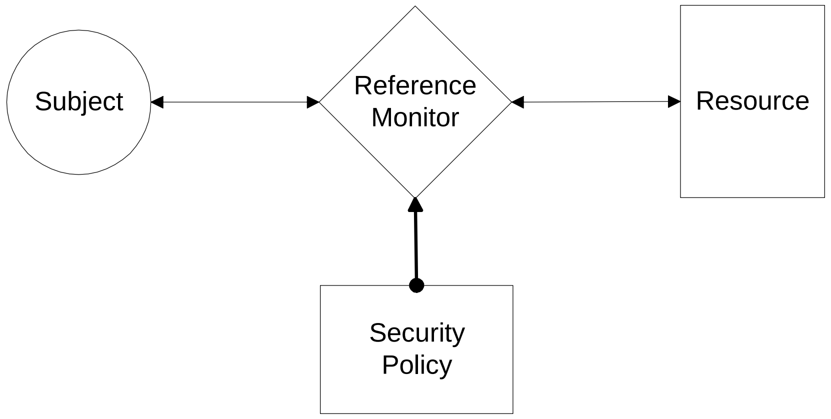

*Figure 23: Reference Monitor Model*

With the Reference Monitor handling the actual mediation requests it becomes the actual policy engine. This then requires that the policy mediation model become more generic, able to run on various platforms, and be able to execute a policy language.

## 11.3 Policy Language

The requirements for a policy language are that it be capable of reacting to, controlling and monitoring the system access event stream. Clearly, this goes beyond saying "yes" or "no" to a request to access a file. A policy language should not merely be passive, but should be capable of interacting with the system being secured to modify it; e.g. reconfiguring a web server if the policy determines that the server is under attack. Most security-related policy languages seem to restrict themselves to the control problem. We believe that this limits their utility.

Much of the work in computer security has delved into verifiable code and mathematical models of policy, often relying on variations of lambda calculus [Church].

The primary requirements for a policy language are: verifiability, small size, efficient execution, expressive functionality, dynamicity and portability. The dynamicity requirement is important as it must be possible to update a policy in real time without taking the policy engine offline. The expressiveness of the language is also important – policies should be concise.

The ability of security policies being able to fetch pertinent information from external (to the policy) information stores such as LDAP, X.500, Certificate Authorities, operating systems, web servers, databases, etc., allows the policy to properly determine whether to grant access or not and what to do about the access event. Clearly, standards-based interfaces available through application programming interfaces (APIs) are required. The information retrieved need not be security related; it could be information on whether an individual is on vacation, what their physical location is, whether another individual is also logged on, and so on. In order to access certain entities it may be necessary to provide additional authentication and this, too, can be provided for in the policy via codified rules requesting additional authentication information.

## 11.4 Policy Mediation

Having defined an entity and policy, mediation need be no more complicated than evaluating:

*Mediate (subject action object)*

where the *subject* is the entity requesting access, as represented by the *action* upon the *object*. It is the job of the mediation authority to determine whether or not such a relationship is allowed within the confines of the defined environment.

The importance of the mediation request is two-fold. First, it matches existing security policy modelling nomenclature regarding subject-object interaction and policy modelling. Second, it is can easily define iterative and reflective definitions. For example, if a user, George, wishes to access a particular application which in turn wishes to access an information resource, the agents would request two separate mediations:

*Mediate (George execute Application env)*

*Mediate (Application open Resource env).*

If state information is required in order to efficiently or more safely process any mediation request then it can be stored in the environment for later use. Using the above example, the environment would remember that it was George that initiated the application and that it was operating on his behalf. Therefore, if any access restrictions were placed on the resource as to which application and which user could access it, the information would be contained within the environment. Since mediations can be logically chained there is no restriction in the number or complexity of the mediation requests that can be handled.

## 11.5 Virtualized Domain

The Virtualized Domain is a database of all of the entities being protected. This is organized around two basic structures: the entity and binary relationships between entities. Examples of binary relationships are:

- *parent(a,b)*, where *a* represents a directory and *b* a file within it;
- *read(x,y)*, where *x* is a user and *y* a file; and
- *execute(u,v)*, where *u* is a user and *v* an executable application.

External programs interact with the virtualized domain through well defined APIs provided as interface mechanisms by the reference monitor. Included with these APIs are full lifecycle management APIs for an entity; i.e. creation, utilization, modification, and deletion.

## 11.6 Policy Store

The Policy Store is a repository for all policies that may be associated with entities. It is responsible for lifecycle maintenance of policies. The repository maintains an association between the policy name and its actual implementation. The actual policy is not stored with the entity; its name is. Both security and access violation policies are stored within the repository. Obviously the Policy Store and the Virtual Domain can both be stored in the same database.

## 11.7 Audit Log

The Audit Log is a repository for all mediation requests. Whenever a mediation request occurs, the request and its result are logged. Should an access violation policy be invoked, it too is logged. The Audit Log insulates the Realm Controller from knowing anything about the log details; i.e. whether it is a simple flat file or a relational database.

## 11.8 Management Console

In order to easily manipulate the reference monitor and the various data stores for entities and policies it is crucial to have some form of Management Console. The Management Console allows an implementer of a policy engine to hide most of the detail and provide only the necessary components to associate policies with entities and entities with one another.

## 11.9 Application Programming Interface & 3rd Party Interoperability

As mentioned previously, it is crucial to have APIs so that the functionality of the engine is available to programmers and policy writers. The interfaces typically required are:

- Policy API;
- I&A API;
- Management/Housekeeping API; and
- Audit API.

## 11.10 Summary

A policy engine is a type of server providing a mediation service based on policy. It provides the means by which a request can be submitted and acted upon and a response will be returned. The response determines the flow of information according to a set of predefined rules as simple or as complex as a developer sees fit.

The engine itself can be either a single-purpose entity or a generic policy engine. A generic engine would use a portable language such as Java as its base and utilize an independently created policy language to describe the policies. The language must be sufficiently rich so as to properly describe a plethora of policies easily and succinctly. Although the authors of Ponder selected a home-grown language, others opted to derive their security policy languages from LISP and LISP-like systems. In order to meet the requirements of creating a highly secure policy engine as well as a verifiable language it behoves anyone attempting to develop an engine to examine the literature and to select a grammar that has a well-defined denotational semantics or other logic so as to facilitate higher-level assurance.

The anticipated effort to create a policy engine based on existing technology is probably in a four to six month timeframe for a medium sized team of six skilled developers. Building the entire system from scratch, and based on prior experience, it is estimated to take nine to twelve months for a medium sized team of six skilled developers. Estimating cost is difficult but assuming an average cost of $1,500 per day based on software and hardware requirements plus labour results in a cost of $1,080,000 for a policy engine based on existing technology and $2,160,000 for one based on technology built from scratch. This is in line with costs experienced by the author during the initial development of SecureRealms at Texar.

# 12. Critique

The requirements outlined above, and summarized in Table 5 below for reference, will be utilized to determine which of the most promising of the policy engines best meet the requirements for a true security policy engine. Note that a determination as to long-term viability of a corporation or technology plays no part in the determination ensuring a purely objective examination.

*Table 5: Requirements for a Policy Engine*

| # | Requirement | Description |
|---|---|---|
| 1 | ***Persistence*** | It must be possible to define and store any policy that may be associated with an entity. |
| 2 | ***Expressive*** | It must be possible to express policies that apply to intentionally defined associations (or groupings) of entities. There should not be a limit, logically or otherwise, as to the number and types of associations. |
| 3 | ***Assignment*** | It must be possible to define which policy (out of a collection of policies) is associated with a particular entity. It should be possible to dynamically change which policy covers which entity through simple means. |
| 4 | ***Natural Language*** | It would be desirable if there were the means by which to automate the natural language specification of a security policy into a logical one understood by the Policy Engine. In other words, taking Business Rules and converting them directly into logic understood by a Policy Engine. |
| 5 | ***Provable*** | The policy must be expressible and checkable in a well-defined and understandable logic/language. |
| 6 | ***Commutative*** | It must be possible for the engine to utilize either the passive and active entities' policies and identifying information in order to determine whether a particular request is actionable. |
| 7 | ***Mediatative*** | The language must be able to define and determine what is to occur upon a denied or allowed action. Furthermore, a violation of a policy must result in the (possible) invocation of a "violation" policy that will perform specific actions based on the failed logic of the originating policy. |
| 8 | ***Associative*** | It must be possible to determine to which entities a given policy is associated and, conversely, which policy is associated with a given entity. |
| 9 | ***Enforcement*** | Each policy must be well-defined and utilized to enforce access by one entity against another. |
| 10 | ***Labelled*** | Each entity controlled by the security policy must be tagged with a label. This label indicates information relevant to the security policy in determining how information can be accessed. In the case of the Bell-LaPadula model, the labels would indicate the clearance of the active entity and the sensitivity label of the passive entity. |
| 11 | ***Unique*** | All entities must be uniquely identified, authenticated, and assigned relevant information pertaining to which security policies are associated with the given entity. |
| 12 | ***Accountable*** | Security relevant events such as entity accesses, entity modification, process instantiation, and access of one entity by another must be recorded in an audit trail. The purpose is to detect and diagnose |

| | | |
|---|---|---|
| | | malicious or non-malicious actions that lead to a non-secure state. |
| **13** | ***Independent*** | The engine allows for policies independent of any underlying architecture. This permits the creation of individual policies that are pertinent to the problem at hand and not restricted to a single "type" of policy, such as Bell-LaPadula or Role-based. It also permits the mixing and matching of policies within the policy space (i.e., any combination of policy types, Lattice, RBAC, etc.). |
| **14** | ***Assured*** | The software must be designed and implemented in such a way that all the preceding requirements are enforced. The security aspects of a product must be of sufficiently simple organization and complexity to be subjected to and pass a wide spectrum of analyses and tests. |

Although the following table provides a completely objective representation as to the viability of a particular solution it is left to the sole discretion of DRDC to determine which, if any, of these solutions will be pursued in future policy-related R&D endeavours.

Table 6 provides a best estimate of whether a particular policy engine solution meets a given requirement. It is highly probable that under a longer survey wherein each policy engine were examined in detail, some of the following values would need to be adjusted.

*Table 6: Comparison of Viable Options vs. Requirements*

| ***Candidate Policy Engines*** | ***Persistence*** | ***Expressive*** | ***Assignment*** | ***Natural Language*** | ***Provable*** | ***Commutative*** | ***Mediatative*** | ***Associative*** | ***Enforcement*** | ***Labelled*** | ***Unique*** | ***Accountable*** | ***Independent*** | ***Assured*** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***POLICIES***[8] | | | | | | | | | | | | | | |
| ***OASIS*** | ✓ | ✓ | | | | | | | | | | | | |
| ***EDAC*** | ✓ | ✓ | ✓ | | | ✓ | ✓ | ✓ | ✓ | ✓ | | | ✓ | |
| ***ACADEMIC RESEARCH ALTERNATIVES***[9] | | | | | | | | | | | | | | |
| ***Authorization Specification Language (ASL)*** | ✓ | ✓ | | | | | ✓ | | ✓ | ✓ | ✓ | | ✓ | ✓ |
| ***Generic Policy Engine*** | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ |
| ***LaSCO*** | ✓ | | ✓ | | | | ✓ | | ✓ | ✓ | | | | |
| ***MeSMO (Meta-Security Model) & OASIS*** | ✓ | ✓ | ✓ | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| ***Pamina*** | ✓ | ✓ | ✓ | | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| ***PolicyMaker/KeyNote*** | ✓ | ✓ | ✓ | | | | ✓ | | ✓ | ✓ | ✓ | ✓ | | |
| ***Ponder***[10] | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ |
| ***REFEREE*** | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | | ✓ | |
| ***Sandhu Policy Research*** | ✓ | ✓ | ✓ | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| ***SDSI/SPKI*** | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | | ✓ | | ✓ | | ✓ | |
| ***Security Policy Language*** | ✓ | ✓ | ✓ | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| ***Semantic Web Rule Language (SWRL)*** | ✓ | ✓ | ✓ | | | | ✓ | | ✓ | ✓ | | | | |
| ***$\chi$-Sec Credential Specification Language*** | ✓ | ✓ | ✓ | | | | ✓ | | ✓ | ✓ | | | ✓ | |
| ***SULTAN*** | | ✓ | | | ✓ | | ✓ | | | | ✓ | ✓ | ✓ | |
| ***STANDARDS & OPEN SOURCE ALTERNATIVES***[11] | | | | | | | | | | | | | | |
| ***GSS API / GAA API*** | ✓ | | ✓ | | | | | | ✓ | | ✓ | | | |

[8] Many of the policies examined were either not applicable or did not have implementations that could be assessed. Consequently, only OASIS and EDAC are analysed.

[9] The academic research alternatives that are hashed out are no longer being actively pursued.

[10] Many of the policy languages, and Ponder is no exception, do not offer a formal Audit facility built into the language. It is assumed that audit will be included as required by the policy writer.

[11] A number of standards and open source alternatives were not assessed for various reasons. IETF/DMTF PCIM is applicable to IP network access control policy, not to digital resources as such, there are no implementations of Project DReaM, and OpenIPMP constitutes a management layer over MPEG-21 and/or ODRL, not a candidate itself for a policy engine.

| *Candidate Policy Engines* | *Persistence* | *Expressive* | *Assignment* | *Natural Language* | *Provable* | *Commutative* | *Mediatative* | *Associative* | *Enforcement* | *Labelled* | *Unique* | *Accountable* | *Independent* | *Assured* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***XACML*** | ✓ | ✓ | ✓ | | | ✓ | ✓ | | ✓ | | ✓ | | ✓ | |
| ***RuleML*** | ✓ | | | | ✓ | | | ✓ | | | | | ✓ | |
| ***EPAL*** | ✓ | ✓ | ✓ | | | ✓ | | ✓ | ✓ | | ✓ | | ✓ | |
| ***ODRL*** | ✓ | ✓ | ✓ | | | ✓ | | ✓ | | ✓ | ✓ | | ✓ | |
| ***MPEG-21*** | ✓ | ✓ | ✓ | | | | | ✓ | | ✓ | ✓ | | ✓ | |
| ***Authena*** | ✓ | | | | | | | ✓ | | | ✓ | | ✓ | |
| ***RESEARCH LABORATORY & INDUSTRY CONSORTIUM ALTERNATIVES*** | | | | | | | | | | | | | | |
| ***PDL*** | ✓ | | | | ✓ | | ✓ | ✓ | | | | | | |
| ***Rei*** | ✓ | ✓ | | | | | | ✓ | | | ✓ | | ✓ | ✓ |
| ***P3P*** | ✓ | | | | | | | | | | | | | |
| ***SESAME*** | ✓ | | | | | | | | | | ✓ | ✓ | | |
| ***Liberty Alliance*** | | | | | | | | | | | | | ✓ | |
| ***WS-I***[12] | ✓ | | | | | | | | | | | | | |
| ***INDUSTRY ALTERNATIVES*** | | | | | | | | | | | | | | |
| ***BMC Software Access Management (AM)*** | ✓ | ✓ | ✓ | | | | | | ✓ | | ✓ | ✓ | | |
| ***Computer Associates eTrust SiteMinder (AM)*** | ✓ | ✓ | ✓ | | | | ✓ | | ✓ | | ✓ | ✓ | | |
| ***Entegrity Solutions AssureAccess (AM)*** | ✓ | ✓ | ✓ | | | | | | ✓ | | ✓ | ✓ | | |
| ***Entrust GetAccess (AM)*** | ✓ | ✓ | ✓ | | | | | | ✓ | | ✓ | ✓ | | |
| ***Evidian Secure Access Manager (AM)*** | ✓ | ✓ | ✓ | | | | | | ✓ | | ✓ | ✓ | | |
| ***HP OpenView Select Access (AM)*** | ✓ | ✓ | ✓ | | | | | | ✓ | | ✓ | ✓ | | |
| ***IBM Tivoli Access Manager (AM)*** | ✓ | ✓ | ✓ | | | | | | ✓ | | ✓ | ✓ | | |
| ***Novell iChain (AM)*** | ✓ | ✓ | ✓ | | | | | | ✓ | | ✓ | ✓ | | |
| ***Oracle COREid Access (AM)*** | ✓ | ✓ | ✓ | | | | | | ✓ | | ✓ | ✓ | | |
| ***RSA Access Manager (AM)*** | ✓ | ✓ | ✓ | | | | | | ✓ | | ✓ | ✓ | | |
| ***Sun Java System Access Manager (AM)*** | ✓ | ✓ | ✓ | | | | | | ✓ | | ✓ | ✓ | | |

[12] The forthcoming WS-Authorization standard may address several criteria currently unsatisfied.

| *Candidate Policy Engines* | *Persistence* | *Expressive* | *Assignment* | *Natural Language* | *Provable* | *Commutative* | *Mediatative* | *Associative* | *Enforcement* | *Labelled* | *Unique* | *Accountable* | *Independent* | *Assured* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Adobe LiveCycle Policy Server (ERM)* | ✓ | ✓ | ✓ | | | | | | ✓ | ✓ | ✓ | ✓ | | |
| *Avoco Secure secure2trust (ERM)* | ✓ | ✓ | ✓ | | | | | | ✓ | ✓ | ✓ | ✓ | | |
| *EMC Active Rights Management Platform (ERM)* | ✓ | ✓ | ✓ | | | | | | ✓ | ✓ | ✓ | ✓ | | |
| *Liquid Machines Document Control/Email Control (ERM)* | ✓ | ✓ | ✓ | | | | | | ✓ | ✓ | ✓ | ✓ | | |
| *LockLizard Protector/Safeguard (ERM)* | ✓ | ✓ | ✓ | | | | | | ✓ | ✓ | ✓ | ✓ | | |
| *Microsoft Windows Rights Management Services (ERM)* | ✓ | ✓ | ✓ | | | | | | ✓ | ✓ | ✓ | ✓ | | |
| *SealedMedia (ERM)* | ✓ | ✓ | ✓ | | | | | | ✓ | ✓ | ✓ | ✓ | | |
| *Entrust Entelligence Content Control Server (EP)* | ✓ | ✓ | ✓ | | | | | | ✓ | | ✓ | ✓ | | |
| *Fidelis Security Systems' DataSafe (EP)* | ✓ | ✓ | ✓ | | | | | | ✓ | | ✓ | ✓ | | |
| *Intrusion Compliance Commander (EP)* | ✓ | ✓ | ✓ | | | | | | ✓ | | ✓ | ✓ | | |
| *Oakley Networks' SureView (EP)* | ✓ | ✓ | ✓ | | | | | | ✓ | | ✓ | ✓ | | |
| *Orchestria Active Policy Management Platform (EP)* | ✓ | ✓ | ✓ | | | | ✓ | | ✓ | ✓ | ✓ | ✓ | | |
| *Palisade Systems' PacketSure(EP)* | ✓ | ✓ | ✓ | | | | | | ✓ | | ✓ | ✓ | | |
| *PortAuthority Technologies (EP)* | ✓ | ✓ | ✓ | | | | | | ✓ | | ✓ | ✓ | | |
| *Proofpoint Protection Server (EP)* | ✓ | ✓ | ✓ | | | | ✓ | | ✓ | ✓ | ✓ | ✓ | | |
| *Reconnex inSight Platform (EP)* | ✓ | ✓ | ✓ | | | | | | ✓ | ✓ | ✓ | ✓ | | |
| *Tablus Solution Suite (EP)* | ✓ | ✓ | ✓ | | | | | | ✓ | | ✓ | ✓ | | |
| *Verdasys Digital Guardian (EP)* | ✓ | ✓ | ✓ | | | | | | ✓ | | ✓ | ✓ | | |
| *Vontu (EP)* | ✓ | ✓ | ✓ | | | | ✓ | | ✓ | | ✓ | ✓ | | |
| *Entrust Authority Key Release Server (P-BS)* | ✓ | ✓ | ✓ | | | | | | ✓ | ✓ | ✓ | ✓ | | |
| ***Design & Implement***[13] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

13 Obviously, with a design and implement solution what requirements were met would be fully at the discretion of the contracting authority. However, nothing precludes the development of solutions to address each and every requirement.

The following sections provide objective analysis of each of the aforementioned solutions coupled with some observations gleaned from examining current directions for the products.

## 12.1 Selection of a Policy Implementation Solutio

The most promising candidates stemming from proof-of-concept policy efforts were OASIS and EDAC. OASIS lacks sufficient dynamicity and granularity. EDAC does have potential but strays from industry standard system architecture, features an overly elaborate system design, and is US Military property. Neither option appears overwhelmingly attractive for DRDC at this time.

## 12.2 Selection of an Academic Solution

There is a renewed interest in policy engines and security in general. Unfortunately not much of this interest has translated into long term support of promising projects. As can be witnessed from the table above most of the academic projects have been abandoned. Many are no more than theses and with some -- Ponder being the prime example -- even the original Open Source code has been withdrawn.

Of the projects that remain active the most promising remains Rei and the work by Sandhu et al. Although Sandhu and his team have abandoned some earlier approaches they continue to learn from those prior experiments as they develop more complex and capable policy engines. Unfortunately these engines are more experimental in nature and are not in any way targeted at production environments.

Even though interest in authorization solutions within computer science remains high, not much effort in commercialization of these endeavours currently exists. This is either, as Sloman has indicated, an indication of a lack of comprehension of issues in the commercial arena in general, or the inability of security experts to impress upon the marketplace the need for a cohesive policy solution.

The final issue facing DRDC in the use of any academic solution is that most – if not all – solutions presented within the academic section are either no longer supported or have no publicly available source code. The only available source code at this time is that based on the RuleML engine which is actively supported within the RuleML.org organization. However, RuleML's battle with XACML remains open to see which "standard" will prevail.

## 12.3 Selection of a Standards-based or Open Source Solution

Open Source efforts continue to develop in computer security. The work of W3C’s XACML Technical Committee is worth examining, especially since the functionality currently provided may be sufficient for DRDC’s immediate needs. Furthermore, becoming an active participant in the various committees and providing source back into the Open Source community is a low cost alternative as compared to developing a solution. XACML is of particular interest because Sun is actively pursuing its development and IBM and others have indicated their willingness to adopt XACML as a go-forward standard. Also, since Sun continues to upgrade and release the source to their Java-based XACML implementation under an Open Source licence and has pledged continued development and maintenance it is an excellent candidate to consider for future R&D work by DRDC in the area of policy.

EPAL is a proprietary IBM specification delivering poorer amenity than XACML on all relevant counts.

DRM solutions such as ODRL and MPEG-21 do not provide enough amenity to constitute a serviceable basis for a DRDC policy engine.

The WS-I efforts continue to develop. In particular, the upcoming WS-Authorization standard may cement a foundation for a DRDC policy engine. This work merits further examination at least, and DRDC could benefit from adopting a leadership role in contributing to its completion. However, all WS-I standards always begin as company-proprietary, and so such options will be limited at least initially.

The Semantic Web efforts such as RuleML and SWRL show promise, but are still early in development. Although DRDC could do well to help advance such Open Source efforts, adoption at present would be tantamount to the development of a custom solution.

## 12.4 Selection of a Research Laboratory Solution

Even though some very interesting R&D is being done in various research laboratories none have released source code or design documents illustrating how their various systems function.

## 12.5 Selection of an Industrial Solution

Four types of industrial alternatives were examined. Three of the alternatives are recognized market segments consisting of a number of different products. None of these industrial alternatives distinguished themselves in meeting the requirements for a true security policy engine. Rather, each of these three industrial alternatives seems to be focused on a particular market segment and includes relatively inflexible policies targeted for that particular use. Consequently, they each lack the flexibility to address the complex policies required by DRDC. The fourth industrial alternative, policy-based security, is not a recognized market segment. It consists of a single product that is no longer undergoing active development, and consequently may or may not be a valid alternative in the future. While this industrial alternative may have some additional policy flexibility due to the fact that it utilizes XACML and is not severely constrained by an administrative interface, these advantages are offset by the fact that the product seems to be more of a prototype than an actual solution.

## 12.6 Design & Develop a Custom Solution

As mentioned earlier, this is a costly venture. It requires that the contracting authority ensure that the team selected has experience in designing and developing quality software as well as a strong background in computer security, and in policy languages and engines. The main advantage is that the software would be built to suit the particular needs of the contracting authority. The major disadvantage is cost and ongoing maintenance. With a minimum cost of just over $1M and an ongoing annual cost of at least $100K it becomes evident that this may not be the most efficient means of addressing DRDC's go-forward policy engine requirements.

## 12.7 Summary

For the sake of comparison, both the original summary (2003) and the updated summary (2006) have been included.

### 12.7.1 Original Summary (2003)

The unfortunate fact is that few of the policy engines discovered during the survey of the literature have actually been implemented. And of those that have, only a few can be considered sufficiently sophisticated to be able to implement complex computer security policies as demanded by the DRDC Proof-of-Concept.

However, there has been work, most notably by the inventors of REFEREE and Ponder, to create useable and stable policy engines. Furthermore, as is evidenced by the volume of available engine research, policy engines will become more common in the near future and it is obvious that existing commercial products will be enhanced with true policy engine capabilities.

The recent release of a Java-based XACML implementation by Sun and IBM's indication of research into XACML with the W3C, it, too, remains a viable candidate for enhanced work on policy engines. In fact, efforts to extend XACML to include generic policy capabilities would be welcomed by Sun, IBM, and the entire W3C community.

### 12.7.2 Original Summary (2006)

In 2002 Sloman observed that there is *considerable interest in policy-based systems from standardization groups, industry, and academia. However, most of the literature relates to proposed systems or small academic prototypes. There is no reported experience of deploying policy-based management in any large-scale system. It is thus too early to judge whether the promised flexibility of policy-based adaptive systems will actually materialize.*

Sloman's observation, now four years old, is just as accurate today given that the ensuing period of time has been a difficult one for policy engine research. While there has been a great deal of activity there has been relatively little progress. In the area of policy research, a significant amount of theoretical research has led to very few actual implementations. Within academia the situation is no better. The original survey identified a number of promising research initiatives, specifically REFEREE and Ponder, within the academic community. Unfortunately, development on these research initiatives has ceased. Lastly, while industrial alternatives abound, they are all point solutions targeted at specific information security applications or a narrow set of problems. These solutions tend to be inflexible, typically implementing a singular policy that is applicable to all information, rather than generic policy engines capable of supporting a variety of complex policies.

On a positive note, all of the activity, however misguided, represents a high level of interest. Furthermore, research continues on promising standards-based and Open Source implementations such as XACML and RuleML. However, while both options represent the most likely candidates for DRDC policy research, they seem to be overly focused on creating access control languages rather than true policy languages that can define any policy an organization wishes to apply to their information. Other promising initiatives that warrant further investigation include the WS-Authorization specification and RAdAC.

# 13. Conclusion

This document has presented an overview of current policy engine technology and policy languages for security, management, and enterprise collaboration as it pertains to the DRDC concept demonstrations of policy-based access control. Broadly speaking, many of the security policy specification languages represent logic-based approaches, whereas management and enterprise collaboration policy specification notations adopt a more informal approach. A common theme across all these specification notations is that they are specialized towards representing either security policies or management policies, not a combination of the two.

Fortunately there appears to be a number of viable candidates that meet the requirements of a policy engine. The availability of Open Source policy engines bodes well to any decision DRDC may take. The true flexibility of each solution can only be determined by actual implementation of a variety of policies in each and testing the efficacy of the implementations. It is also hoped that over time some of the other interesting projects will be released publicly as Open Source so that interested researchers can examine the workings of some modern interpretations of a security policy engine.

Since the original survey was conducted in 2003 there has been a great deal of activity, representing a corresponding level of interest, in policy engine research. Unfortunately, this activity has not resulted in an equivalent level of progress. While a number of viable candidates have progressed marginally, research on other candidates has ceased entirely. Most importantly, all of this research has resulted in very few actual implementations of generic policy engines capable of supporting complex policies.

Furthermore, much of this policy engine work has focused specifically on access control and confidentiality without addressing management policies. A holistic approach should encompass all aspects of information protection and accessibility, rather than just addressing part of the problem. In addition, a great deal of policy engine work attempts to restrict the definition of data rather than providing a solution that is applicable to the entire spectrum of information. This is at odds with the purpose of a uniform security policy system that can apply consistent policies to information regardless of where the information is stored or how it is accessed.

This page intentionally left blank.

# References

[1] [Abrams 1990] Abrams, Marshall D. et al. A Generalized Framework for Access Control: An Informal Description. MITRE Technical Report #MP-90W00043, MITRE, McLean, Virginia. August 1990.

[2] [Abrams 1993] Abrams, M. D. 1993. "Renewed Understanding of Access Control Policies". 16th National Computer Security Conference, Baltimore, Maryland, U.S.A.

[3] [Abrams 1993a] Abrams, Marshall D. and Michael V. Joyce. On TCB Subsets and Trusted Object Management. MITRE Technical Report #92W0000248, MITRE, McLean, Virginia. January 1993.

[4] [Abrams 1993b] Abrams, Marshall D. Perspectives on General TCB Subsets, Supplementary Reading Material for Tutorial #6, 9th Annual Computer Security Applications Conference, December 7, 1993.

[5] [Ahn 2000] G-J. Ahn and R. Sandhu 2000. "Role-based Authorization Constraints Specification". George Mason University.

[6] [Ahn-Sandhu 1999] Ahn, G.-J. and R. Sandhu 1999. "The RSL99 Language for Role-based Separation of Duty Constraints". Fourth ACM Workshop on Role-Based Access Control, Fairfax, Virginia, USA, ACM Press.

[7] [Ahn-Sandhu 2001] Ahn, Gail-Joon and R. Sandhu. "Role-Based Authorization Constraints Specification", ACM Transactions on Information and System Security, Vol. 3, No. 4. November 2000, pgs. 207 – 226.

[8] [Alchourron] Alchourron, C. E. 1971. Normative Systems. New York, Wien: xvii+208.

[9] [Amoroso] Amoroso, Edward G. Fundamentals of Computer Security Technology, Prentice-Hall, Englewood Cliffs, New Jersey, 1994.

[10] [Anderson 1972] Anderson, Jim. Computer Security Technology Planning Study, ESD-TR-73-51, Volume I, AD-758, ESD/AFSC, Hanscom AFB, Bedford, Massachussetts. October 1972.

[11] [Anderson 2005a] A. Anderson (ed.) 2005. "Core and hierarchical role based access control (RBAC) profile of XACML v2.0", OASIS Open.

[12] [Anderson 2005b] A. Anderson (ed.) 2005. "Hierarchical resource profile of XACML v2.0", OASIS Open.

[13] [Anderson 2005c] A. Anderson (ed.) 2005. "Multiple resource profile of XACML v2.0", OASIS Open.

[14] [Anderson 2005d] A. Anderson (ed.) 2005. “XML Digital Signature profile of XACML v2.0”, OASIS Open.

[15] [Anderson 2005e] Steve Anderson et al. 2005. “Web Services Trust Language”; Actional Corporation, BEA Systems Inc., Computer Associates International Inc., International Business Machines Corporation, Layer 7 Technologies, Microsoft Corporation, Oblix Inc., OpenNetwork Technologies Inc., Ping Identity Corporation, Reactivity Inc., RSA Security Inc., VeriSign Inc. (http://specs.xmlsoap.org/ws/2005/02/trust/WS-Trust.pdf).

[16] [Anderson 2005f] A. Anderson. 2005. “A Comparison of Two Privacy Policy Languages: EPAL and XACML”, Sun Labs.

[17] [Anderson-Lockhart 2005] A. Anderson and Hal Lockhart (eds.) 2005. “SAML 2.0 profile of XACML v2.0”, OASIS Open.

[18] [Anderson 1996] Anderson, R. J. 1996. “A Security Policy Model for Clinical Information Systems”. IEEE Symposium on Security and Privacy, Oakland, California, U.S.A.

[19] [ANSI 359] ANSI INCITS 359. 2004. “Role Based Access Control”, ANSI Standard.

[20] [Anton 2000] Antón, A. I., J. H. Dempster, et al. 2000. “Deriving Goals from a Use Case Based Requirements Specification for an Electronic Commerce System”. Sixth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ), Stockholm, Sweden.

[21] [Anton 1996] Antón, A. I. 1996. “Goal-Based Requirements Analysis”. Second IEEE International Conference on Requirements Engineering (ICRE '96), Colorado Springs, Colorado.

[22] [Apt] Apt, K. R., H. A. Blair, et al. 1988. “Towards a Theory of Declarative Knowledge”. Foundations of Deductive Databases. J. Minker. San Mateo, CA, Morgan Kaufmann: 89-148.

[23] [Ashley 2000] P. Ashley et al. 2000, “A Secure European System for Applications in a Multi-vendor Environment”, ISRC (https://www.cosic.esat.kuleuven.ac.be/sesame/)

[24] [Ashley 2003] Paul Ashley et al 2003. “Enterprise Privacy Authorization Language (EPAL 1.2”, International Business Machines Corporation

[25] [Bacic 1989] Bacic, Eugen M., “Process Execution Controls as a Mechanism to Ensure Consistency,” Fifth Annual Computer Security Applications Conference, December, 1989. Tucson, Arizona.

[26] [Bacic 2002] Bacic, E., “Security as Collaborative Relationships”, Collaborative Technologies Symposium 2002, San Antonio, Texas, January 2002.

[27] [Bacic 1990a] Bacic, Eugen M., “Process Execution Controls: Revisited,” Sixth Annual Computer Security Applications Conference, December, 1990. Tucson, Arizona.

[28] [Bacic 1990b] Bacic, Eugen M., “The Canadian Trusted Computer Product Evaluation Criteria,” Sixth Annual Computer Security Applications Conference, December, 1990. Tucson, Arizona.

[29] [Bacic 1998] Bacic, Eugen M., The Generic Policy Engine, Master of Computer Science Thesis, School of Computer Science, Carleton University, Ottawa, Canada. May 1998.

[30] [Bacic 2002a] Bacic, Eugen and T. White, “Authorization as a Service Provided by a Generic Policy Engine”, 2002 International Conference on Security and Management. June 2002. Las Vega, Nevada.

[31] [Bacic 2002b] Bacic, Eugen, “Practices to Successfully Share Sensitive or Time Critical Information”, 14th Annual Canadian Computer Security Symposium, May 2002. Ottawa, Ontario.

[32] [Bacic-Kuchta] Bacic, Eugen M. and Milan S. Kuchta, “Considerations in the Preparation of a Set of Availability Criteria,” Third Annual Canadian Computer Security Symposium, May, 1991. Ottawa, Ontario.

[33] [Bacon 2000] Jean Bacon et al. 2000. “Generic Support for Distributed Applications.” IEEE Computer.

[34] [Bajaj 2003] S. Bajaj et al. 2003. "The Web Services Federation Language (WS-Federation)"; International Business Machines Corporation, Microsoft Corporation, BEA Systems Inc.; RSA Security Inc.; Verisign Inc.

[35] [Bajaj 2006] S. Bajaj et al. 2006. “Web Services Policy 1.2 – Framework (WS-Policy)”. BEA Systems Inc., International Business Machines Inc., Microsoft Corporation Inc., SAP AG, Sonic Software, Verisign Inc., available under W3C Document Licence. (http://www.w3.org/Submission/WS-Policy/).

[36] [Barkely-Beznosov] Barkley, J. F., K. Beznosov, et al. 1999. “Supporting Relationships in Access Control Using Role Based Access Control”. Fourth ACM Workshop on Role-Based Access Control, Fairfax, Virginia, USA.

[37] [Barker 2000] Barker, S. 2000. “Security Policy Specification in Logic”. International Conference on Artificial Intelligence (ICAI00), Las Vegas, Nevada, USA.

[38] [Barker 2001] Barker, S. 2001. Access Control Policies as Logic Programs. London, Imperial College of Science, Technology and Medicine.

[39] [Barker-Rosenthal] Barker, S. and A. Rosenthal 2001. “Flexible Security Policies in SQL”. Fifteenth Annual IFIP WG 11.3 Working Conference on Database and Application Security, Niagara on the Lake, Ontario, Canada.

[40] [Barkley-Kuhn] Barkley, J., R. Kuhn, et al. 1998. “Role-Based Access Control for the Web”. CALS Expo International & 21st Century Commerce 1998: Global Business Solutions for the New Millennium, Long Beach, CA, USA.

[41] [Bell] Bell, D. Elliott. “Concerning ‘Modeling’ of Computer Security,” Proceedings of the 1988 Symposium on Security and Privacy, Oakland, California.

[42] [Bell-LaPadula] Bell, David E. and L.J. LaPadula. Secure Computer Systems: Mathematical Foundations, ESD-TR-73-278, Volumes I, II, and III. The MITRE Corporation, March, May, and December 1973.

[43] [Bertino 2000] Bertino, E., P. Bonatti, et al. 2000. “TRBAC: A Temporal Role-Based Access Control Model”. 5th ACM Workshop of Role-Based Access Control, Berlin, Germany.

[44] [Bertino 2001] Bertino, E., S. Castano and Elena Ferrari. “On Specifying Security Policies for Web Documents with an XML-based Language”, SACMAT ’01, May 3 – 4, 2001, Chantilly, Virginia, pg. 57 – 65.

[45] [Bhatti] Bhatti, R. A. X-GTRBAC: An XML-based Policy Specification Framework and Architecture for Enterprise-wide Access Control. Master’s Thesis, Electrical and Computer Engineering, Purdue University, May 2003.

[46] [Biba] Biba, K.J. Integrity Considertations for Secure Computer Systems, ESD-TR-76-372, MTR-3153, The MITRE Corporation, Bedford Massachussetts. April 1977.

[47] [Blake] Blake, S., D. Black, et al. 1998. An Architecture for Differentiated Services. Network Working Group - RFC2475, http://www.ietf.org/rfc/rfc2475.txt.

[48] [Blaze 1996] Blaze, M., J. Feigenbaum, et al. 1996. “Decentralized Trust Management”. AT&T Research.

[49] [Blaze 1998] Blaze, M., J. Feigenbaum, et al. 1998. “Keynote: Trust Management for Public-Key Infrastructures”. Security Protocols International Workshop, Cambridge, England, Springer-Verlag LNCS.

[50] [Blaze 1999] Blaze, M., J. Feigenbaum, et al. 1999. “The Role of Trust Management in Distributed Systems Security”. Secure Internet Programming: Security Issues for Mobile and Distributed Objects. New York, NY, USA, Springer-Verlag: 185 - 210.

[51] [Bos] Bos, H. 1999. “Application-Specific Policies: Beyond the Domain Boundaries”. Sixth IFIP/IEEE International Symposium on Intergrated Network Management (IM'99), Boston, MA, USA.

[52] [Boswell] Boswell, A. 1995. “Specification and Validation of a Security Policy Model”. IEEE Transacations on Software Engineering 21(2).

[53] [Braghin 2004] Chiara Braghin et al. 2004. “A Distributed Calculus for Role-Based Access Control”. Univ. Ca’ Foscari de Venezia.

[54] [Brandozzi] Brandozzi, M. and D. E. Perry 2001. “Transforming Goal Oriented Requirement Specifications into Architectural Prescriptions”. First International Workshop From Software Requirements to Architectures (STRAW'01), Toronto, Canada.

[55] [Brewer-Nash] Brewer, D. F. C. and M. J. Nash 1989. “The Chinese Wall Security Policy”. IEEE Symposium on Research in Security and Privacy, Oakland, California, USA, IEEE.

[56] [Bull 2002] J.A. Bull, L. Gong, K.R. Sollins 1992. “Towards security in an open systems federation”. European Symposium on Research in Computer Security (ESORICS).

[57] [Burgess 2001a] Burgess, M. 2001. Recent Developments in CfEngine. Unix NL Conference, The Hague.

[58] [Burgess 2001b] Burgess, M. and F. E. Sandnes 2001. “Predictable Configuration Management in a Randomized scheduling Framework”. IEEE/IFIP Workshop on Distributed Systems Operations and Management (DSOM '2001), Nancy, France.

[59] [Burgess 1995] Burgess, M. 1995. “A Site Configuration Engine”. USENIX Computing systems 8(3).

[60] [Casassa] Casassa Mont, M., A. Baldwin, et al. 1999. POWER Prototype: Towards Integrated Policy-Based Management. Bristol, UK, HP Laboratories Bristol.

[61] [Castaneda] Castaneda, H. 1981. The Paradoxes of Deontic Logic. New Studies in Deontic Logic: Norms, Actions and the Foundations of Ethics. Hingham, MA, Reidel Publishing Company: 37-85.

[62] [CC] The Common Criteria. Harmonised Criteria of Canada, the United States, France, Germany, the Netherlands, and the United Kingdom. Version 2, 1999.

[63] [Castano] S. Castano et al. 1995. “Database Security”. Addison-Wesley Publishing Company.

[64] [CGI 2005] White Paper 2005. “Governance-Based Access Control (GBAC): Enabling improved information sharing that meets compliance requirements.” CGI.

[65] [Chandra] Chandra, A. and D. Harel 1985. “Horn Clause Queries and Generalizations”. Journal of Logic Programming 2(1): 1-5.

[66] [Chen] Chen, F. and R. S. Sandhu 1995. “Constraints for Role-Based Access Control”. First ACM/NIST Role Based Access Control Workshop, Gaithersburg, Maryland, USA, ACM Press.

[67] [Cholvy-Cuppens] Cholvy, L. and F. Cuppens 1997. “Analyzing Consistency of Security Policies”. IEEE Symposium on Security and Privacy (S&P97), Oakland, CA, IEEE Press.

[68] [Chomicki] Chomicki, J., J. Lobo, et al. 2000. “A Logic Programming Approach to Conflict Resolution in Policy Management”. Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000), Breckenridge, Colorado, USA, Morgan Kaufmann.

[69] [Choudhary 2005] Dr. A. Rahim Choudhary. 2005. “A Policy Based Architecture for NSA RAdAC Model”. 6th IEEE IA Workshop, USMA, Westpoint.

[70] [Church] Church, A. 1936. “An unsolvable problem of elementary number theory”. American Journal of Mathematics, 58 (1936),

[71] [Clark-Wilson] Clark, David D. and D.R. Wilson. “A Comparison of Commercial and Military Computer Security Policies,” Proceedings of the 1987 Symposium on Security and Privacy.

[72] [Clocksin] Clocksin, W.F. and C.S. Mellish. Programming in Prolog, Springer-Verlag, New York. 1981.

[73] [Conover] Conover, J. 1999. Policy-Based Network Management. Network Computing. http://www.networkcomputing.com/1024/1024f1.html.

[74] [Corradi] Corradi, A., R. Montanari, et al. 2000. “A Flexible Access Control Service for Java Mobile Code”. Annual Computer Security Applications Conference (ACSAC 2000), New Orleans, Louisiana, USA, IEEE Press.

[75] [CTCPEC] Communications Security Establishment, The Canadian Trusted Computer Product Evaluation Criteria. Version 3.0e, January 1993.

[76] [Cuppens-Saurel] Cuppens, F. and C. Saurel 1996.” Specifying a Security Policy: A Case Study”. Ninth IEEE Computer Security Foundations Workshop, Co. Kerry, Ireland, IEEE Press.

[77] [Damianou] Damianou, N., N. Dulay, et al. 2001. “The Ponder Policy Specification Language”. Policy 2001: Workshop on Policies for Distributed Systems and Networks, Bristol, UK, Springer-Verlag.

[78] [Dantsin] Dantsin, E., T. Eiter, et al. 1997. “Complexity and Expressive Power of Logic Programming”. 12th Annual IEEE Conference on Computational Complexity (CCC'97), Ulm, Germany, IEEE Press.

[79] [Darimont 1995] Darimont, R. 1995. Process Support for Requirements Elaboration. Département d'Ingénierie Informatique. Louvain-la-Neuve, Belgium, Université catholique de Louvain.

[80] [Darimont 1996] Darimont, R. and A. Van Lamsweerde 1996. “Formal Refinement Patterns for Goal-Driven Requirements Elaboration”. 4th ACM Symposium on the Foundations of Software Engineering (FSE4): 179-190.

[81] [Della-Libera] Giovanni Della-Libera et al. 2005. “Web Services Security Policy Language (WS-SecurityPolicy)”. International Business Machines Corporation, Microsoft Corporation, RSA Security Inc., VeriSign Inc. (http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf).

[82] [Dimmock 2004] N. Dimmock et al. 2004. “Using Trust and Risk in Role-Based Access Control Policies”. University of Cambridge Computer Laboratory.

[83] [DoD 1985] U.S. Department of Defence. 1985. “Department of Defense Trusted Computer System Evaluation Criteria”. U.S. DoD 5200.28-STD.

[84] [Entrust 2005] White paper. 2005. “Entrust Authority Key Release Server 7.0 Administration Guide, version 1.0”. Entrust Technologies Ltd.

[85] [Essmayr] Essmayr, Wolfgang, Kapsammer, Elisabeth, and Tjoa, A. Min. “Meta-Data for Enterprise-Wide Security Administration.” Third IEEE Meta-Data Conference, April, 1999, Bethesda, Maryland. http://www.computer.org/proceedings/meta/1999/papers/30/wessmayr.html

[86] [Fernandez 2006] Richard Fernandez 2006. “Enterprise Dynamic Access Control, Version 2, Overview”, U.S. Navy.

[87] [Fernando 2005] G. Fernando, et al. 2005. "Project DReaM, An Architectural Overview". Sun Microsystems,

[88] [Ferraiolo 1992] Ferraiolo, D and Kuhn, R. “Role Based Access Controls”. 15th NIST-NCSC National Computer Security Conference, pp. 554-563, October 1992.

[89] [Ferraiolo 1995] Ferraiolo, D., Cugini, J. and Kuhn, R. “Role Based Access Control: Features and Motivations”. Annual Computer Security Applications Conference, IEEE Computer Society Press, 1995 (http://hissa.ncsl.nist.gov/rbac/newpaper/rbac.html).

[90] [Ferraiolo 2001] D. Ferraiolo et al. 2001. “Proposed NIST Standard for Role-Based Access Control”, National Institute of Standards and Technology)

[91] [Gasser] Gasser, Morrie. Building a Secure Computer System, Van Nostrand Reinhold, New York, New York, 1988.

[92] [Gassko] I. Gassko, P. S. Gemmell and P. MacKenzie: “Efficient and Fresh Certification”, Proceedings of the Conference Public Key Cryptography 2000, v. 1751 of LNCS, pp. 342–353, Springer, 2000

[93] [Georgiadis 2001] Georgiadis, Mayridis, Pangalos, Thomas 2001. “Flexible Team-based Access Control Using Contexts”. Proceedings of the sixth ACM symposium on Access control models and technologies.

[94] [Glasgow] Glasgow, J., G. Macewen, et al. 1992. “A logic for reasoning about security”. ACM Transactions on Computer Systems (TOCS) 10(3): 226-264.

[95] [Gligor] Gligor, Virgil, et al. “Design and Implementation of Secure Xenix,” IEEE Transactions on Software Engineering, Volume 13, Number 2, February 1987.

[96] [Grandison] Grandison, T. and M. Sloman 2000. “A Survey of Trust in Internet Applications”. IEEE Communications Surveys and Tutorials 3(4). [See also http://www.doc.ic.ac.uk/~tgrand/]

[97] [Hayton] Hayton, R. J., J. M. Bacon, et al. 1998. “Access Control in an Open Distributed Environment”. IEEE Symposium on Security and Privacy, Oakland, California, U.S.A.

[98] [Healthcare 2003 Healthcare Task Force 2003. “Healthcare RBAC Task Force Charter Version 1.1", The Healthcare RBAC Task Force (http://www.va.gov/RBAC/docs/HealthcareRBACTCharterv1.1.doc)

[99] [Herzberg] Herzberg, A., Y. Mass, et al. 2000. “Access Control Meets Public Key Infrastructure, or: Assigning Roles to Strangers”. IEEE Symposium on Security and Privacy, Oakland, California, USA.

[100] [Hoagland] Hoagland, J. A., R. Pandey, et al. 1998. Security Policy Specification Using a Graphical Approach, UC Davis Computer Science Department.

[101] [Horrocks 2004] Horrocks et al. 2004. “SWRL: A Semantic Web Rule Language Combining OWL and RuleML”. National Research Council of Canada, Network Inference, Stanford University (tp://www.w3.org/Submission/SWRL/).

[102] [Hosmer 1991] Hosmer, Hilary H. “Metapolicies I,” ACM SigSAC Special Workshop on Data Management Security and Privacy Standards, San Antonio, Texas, December 1991.

[103] [Hosmer 1992] Hosmer, Hilary H., “Metapolicies II,” Proceedings of the 15th National Computer Security Conference, October, 1992, Baltimore, Maryland.

[104] [HP] Hewlett-Packard Company. “Proposal for Discretionary Access Control,” IEEE P1003.6, March 14, 1988, Austin, Texas.

[105] [ISO-IEC 1999]ISO/IEC 1999. Information Technology - Open Distributed Processing Reference Model - Enterprise Viewpoint.

[106] [Jager] Jager, G. and R. F. Stark 1993. “The Defining Power of Stratified and Hierarchical Logic Programs”. Journal of Logic Programming 15(1 & 2): 55-77.

[107] [Jajodia 1997] Jajodia, S., P. Samarati, et al. 1997. “A Logical Language for Expressing Authorisations”. IEEE Symposium on Security and Privacy, Oakland, USA, IEEE.

[108] [Jajodia 2000] Jajodia, S., P. Samarati, et al. 2000. “Flexible Support for Multiple Access Control Policies”. ACM Transactions on Database Systems 26(2): 214-260.

[109] [Jones] Jones, A. J. I. and M. J. Sergot 1995. A Formal Characterisation of Institutionalised Power. Logic Journal of the IGPL 4(3): 429-445.

[110] [Joshi 2003] Joshi, J.B.D., E. Bertino, B. Shafiq, and A. Ghafoor. "Dependencies and Separation of Duty Constraints in GTRBAC", SACMAT '03, June 2-3, 2003, Como, Italy, pgs. 51 – 64.

[111] [Kagal 2003] Kagal, L., T. Finin, and A. Joshi. "A Policy Language for a Pervasive Computing Environment", 4th International Workshop on Policies for Distributed Systems and Networks (Policy '03), 2003.

[112] [Katzke] Katzke, Stuart W.; Ruthberg, Zella G., editors. Report of the Invitational Workshop on Integrity Policy in Computer Information Systems (WIPCIS). Washington DC: NIST; January 1989; Special Pub 500-160. SN 003-003-02904-1.

[113] [Kohli] Kohli, M. and J. Lobo 1999. Policy Based Management of Telecommunication Networks. Policy Workshop 1999, HP Labs, Bristol, UK.

[114] [Lampson 1971] Lampson, B. 1971. "Protection". Proc. 5th Princeton Conference on Information Sciences and Systems. Princeton, USA.

[115] [Lee-ES] Lee, E.S et al. Composability of Trusted Systems, Reports 1 - 5. Computer Systems Research Institute, University of Toronto.

[116] [Lee-M] LEE, M. 2000. "Event and Rule Services for Achieving a Web-based Knowledge Network". Computer and Information Science and Engineering, University of Florida.

[117] [Lee-TMP] Lee, Theodore M.P. "Using Mandatory Integrity to Enforce "Commercial" Security," Proceedings of the 1988 Symposium on Security and Privacy, Oakland, California. pp. 140 - 146.

[118] [Li 1989] Li Gong 1989. "A secure identity-based capability system". Proceedings of the IEEE Symposium on Security and Privacy (Los Angeles, USA).

[119] [Li 2002] N. Li et al 2002. "Design of a Role-Based Trust Framework". Proceedings of the IEEE Symposium on Security and Privacy.

[120] [Li 2005] Li, Huiying, X. Zhang, H. Wu, and Y. Qu. "Design and Application of Rule Based Access Control Policies", 4th International Semantic Web Conference, Nov. 6-10, 2005. Galway, Ireland.

[121] [Lin 1999] A. Lin 1999. "Integrating Policy-Driven Role Based Access Control with the Common Data Security Architecture". HP Laboratories Bristol, UK.

[122] [Linn 1990] Linn, J. 1990. "Generic Security Service Application Program Interface". Proc. IEEE Symposium on Research in Security and Privacy, pp. 35-54. Portland, USA.

[123] [Lobo] Lobo, J., R. Bhatia, et al. 1999. “A Policy Description Language”. AAAI, Orlando, Florida.

[124] [Lupu 1997] Lupu, E. C. and M. S. Sloman 1997. “Towards a Role Based Framework for Distributed Systems Management”. Journal of Network and Systems Management 5(1): 5-30.

[125] [Lupu 1999] Lupu, E. C. and M. S. Sloman 1999. “Conflicts in Policy-Based Distributed Systems Management”. In IEEE Transactions on Software Engineering - Special Issue on Inconsistency Management 25(6): 852-869.

[126] [Magar 2001a] Magar, A. “An Initial Investigation of Privilege Management Infrastructure”. DRDC Ottawa CR 2002-058, March 2001, Defence R&D Canada - Ottawa.

[127] [Magar 2001b] Magar, A. “Managing Identity and Access in a Classified Environment”. DRDC Ottawa CR 2001-081, October 2001, Defence R&D Canada - Ottawa.

[128] [Magar 2002] Magar, A. “Plan for a Proof-of-Concept (POC) Demonstration of a Privilege Management Infrastructure (PMI) for the Defence Environment”. DRDC Ottawa CR 2002-065, June 2002, Defence R&D Canada - Ottawa.

[129] [Magar 2003a] Magar, A. “Report on the Privilege Management Infrastructure (PMI) Proof-of-Concept (POC) Demonstration”. DRDC Ottawa 2003-003, January 2003, Defence R&D Canada - Ottawa .

[130] [Magar 2003b] Magar, A. “Enhanced Windows-based Warning Terms Separation Proof-of-Concept (POC) Architecture”. DRDC Ottawa CR 2003-207, December 2003, Defence R&D Canada - Ottawa.

[131] [Magar 2004a] Magar, A. “Report on the Enhanced Windows-based Warning Terms Separation Proof-of-Concept (POC) Demonstrator”. DRDC Ottawa CR 2004-058, April 2004, Defence R&D Canada - Ottawa.

[132] [Magar 2004b] Magar, A. “Report on Secure Access Management Proof–of-Concept (SAMPOC) II with Identity Management”. DRDC Ottawa CR 2004-122, June 2004 Defence R&D Canada - Ottawa.

[133] [Manna] Manna, Z. and A. Pnueli 1992. The Temporal Logic of Reactive and Concurrent Systems, Springer-Verlag.

[134] [Marsh]Dave Marsh. 2005. "Output Content Protection and Windows Vista". Microsoft Inc.

[135] [Martinez] Martinez, P., and Brunner M., et. al. “Using the Script MIB for Policy-based Configuration Management”, IEEE/IFIP Network Operations and Management (NOMS2002), Florence, April 2002.

[136] [McGraw 2004]R. McGraw 2004. “Securing Content in the Department of Defense's Global Information Grid”. National Security Agency.

[137] [McGucken 2001] Dr. Elliot McGucken et al. 2001. “Open Source DRM: The Future of Intellectual Property: Marring Open Source CMS and Open Source DRM”. Harvard Law School & OSCOM.ORG (http://www.authena.org).

[138] [McLaughlin 2004] John McLaughlin 2004. “Information Security Research Challenges”, International Business Machines Corporation.

[139] [Minksy 1991] Minsky, N. H. 1991. “The Imposition of Protocols Over Open Distributed Systems”. IEEE Transactions on software Engineering 17(2): 183-195.

[140] [Minsky 1997] Minsky, N. H. and P.Pal 1997. “Law-Governed Regularities in Object Systems - Part 2: A Concrete Implementation”. Theory and Practice of Object Systems (TAPOS), John Wiley. 2.

[141] [Mintchell 2004] Gary Mintchell 2004, “Standards Body Takes On Security”, Automation World, adapted from the first Technical Report of ISA SP99 Working Group ( http://www.automationworld.com/cds_print.html?rec_id=506).

[142] [Miro] Heydon, A., Maimone, et al. “Miro: Visual Specification of Security”, IEEE Transactions on Software Engineering, vol. 16(10), pp. 1185-1197, October 1990.

[143] [Moffett 1993] Moffet, J. and M. S. Sloman 1993. “Policy Hierarchies for Distributed Systems Management”. IEEE JSAC 11(9 (Special Issue on Network Management)): 1404-1414.

[144] [Moffett 1998] Moffett, J. D. 1998. “Control Principles and Role Hierarchies”. Third ACM/NIST Role Based Access Control Workshop, Fairfax, Virginia, USA, ACM Press.

[145] [Mont] Mont, M. C., A. Baldwin, Et al. 1999B. POWER Prototype: Towards Integrated Policy-Based Management. Bristol, UK, Extended Enterprise Laboratory, HP Laboratories.

[146] [Moore] Moore, B., E. Ellesson, Et al. 2001. Policy Core Information Model -- Version 1 Specification. Network Working Group - RFC3060, http://www.ietf.org/rfc/rfc3060.txt.

[147] [Moses 2005a] T. Moses (ed.) 2005. “eXtensible Access Control Markup Language (XACML) Version 2.0”, OASIS Open

[148] [Moses 2005b] T. Moses (ed.) 2005. “Privacy Profile of XACMLv2.0”, OASIS Open

[149] [Mossakowski 2003] Mossakowski, Drouineaud, Sohr 2003. “A temporal-logic extension of role based access control covering dynamic separation of duties”. Proceedings of the 4th International Conference on Temporal Logic

[150] [Nadalin] A. Nadalin et al. (ed.) 2006. “Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)”, OASIS Open (http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf).

[151] [Neuman 1993] C.B. Neuman 1993. “Proxy-based authorization and accounting for distributed systems”. 13th International Conference on Distributed Computing Systems.

[152] [NIST 2006] Draft Paper 2006. “Role Based Access Control Implementation Standard Version 0.1”, National Institute of Standards and Technology (http://www-08.nist.gov/rbac/draft-rbac-implementation-std-v01.pdf)

[153] [Nochta] Zoltn Nochta, Peter Ebinger, and Sebastian Abeck. “Pamina: A certificate based privilege management system”. In Proceedings of the IEEE Symposium

[154] [OASIS] OASIS 2001. XACML language proposal, version 0.8.

[155] [ODRL 1.1] R. Iannella. 2002. “Open Digital Rights Language (ODRL) Version 1.1”. W3C Note (http://www.w3.org/TR/odrl).

[156] [OMG] OMG 1999B. Object Constraint Language Specification, version 1.3.

[157] [OpenIPMP 2003] Various authors. 2003. “OpenIPMP [tm] Open Source Rights Management”. Objectlab LLC (http://objectlab.com/clients/openipmp/index.htm).

[158] [Ortalo]ORTALO, R. 1998. “A Flexible Method for Information System Security Policy Specification”. 5th European Symposium on Research in Computer Security (ESORICS 98), Louvain-la-Neuve, Belgium, Springer-Verlag.

[159] [Park 2002] Park, Jaehong and Ravi Sandhu. “Towards Usage Control Models: Beyond Traditional Access Control”, SACMAT 02, June 3-4, 2002, Monterey, California. Pgs. 57 – 64.

[160] [Park 2004] Park, Jaeohong and R. Sandhu. “The UCONABC Usage Control Model”, ACM Transactions on Information and System Security (TISSEC), Vol. 7, No. 1. February 2004, pgs. 128 – 174.

[161] [Policy 2001] Policy 2001, Workshop on Policies for Distributed Systems and Networks, Bristol, 29-31 January 2001.

[162] [Prakken] Prakken, H. and M. J. Sergot 1997. “Dyadic Deontic Logic and Contrary-to-duty Obligations”. Defeasible Deontic Logic: Essays in Nonmonotonic Normative Reasoning. D. Nute. Boston, Kluwer Academic Publishers. Synthese Library: 263: 223-262.

[163] [RFC 3060] B. Moore et. al 2001. “RFC 3060 – Policy Core Information Model – Version 1 Specification”. The Internet Society.

[164] [RFC 3460] B. Moore (ed.) 2003. “RFC 3460 – Policy Core Information Model (PCIM) Extensions”. The Internet Society.

[165] [RFC 3703] Strassner et al 2004. “Policy Core Lightweight Directory Access Protocol (LDAP) Schema”. The Internet Society.

[166] [RFC 4011] S. Waldbusser et al 2005. “Policy Based Management MIB”. The Internet Society.

[167] [RFC 4104] M. Pana et al 2005. “Policy Core Extension Lightweight Directory Access Protocol Schema”. The Internet Society.

[168] [Ribeiro 1999] Ribeiro, C. and P. Guedes. “SPL: An Access Control Language for Security Policies with Complex Constraints”, Technical Report RT/0001/99, INESC, January 1999.

[169] [Ribeiro 2001a] Ribeiro, C., A. Zuquete, Et al. 2001A. “SPL: An access control language for security policies with complex constraints”. Network and Distributed System Security Symposium (NDSS'01), San Diego, California.

[170] [Ribeiro 2001b] Ribeiro, C., A. Zuquete, Et al. 2001B. “Enforcing Obligation with Security Monitors”. Third International Conference on Information and Communications Security (ICICS 2001), Xian, China.

[171] [Rightscom 2003] White Paper. 2003. "The MPEG-21 Rights Expression Language ". Rightscom, London, UK.

[172] [Ryutov-Neuman] Tatyana Ryutov, Clifford Neuman. 2000. “Access Control Framework for Distributed Applications”. IETF CAT Working Group, USC/Information Sciences Institute (http://gost.isi.edu/info/gaaapi/doc/drafts/frmw_draft5.txt).

[173] [Saltzer] Saltzer, J.H. “The Protection and Control of Information Sharing in Multics,” Communications of the ACM, Volume 17, Number 7, July 1974.

[174] [Samarati] Samarati, P. and S. Vimercati 2000. “Access Control: Policies, Models, and Mechanisms”. Foundations of Security Analysis and Design (Tutorial Lectures). R. Focardi and R. Gorrieri, Springer -Verlag: 137-196.

[175] [Sandhu 1985] Sandhu, R. S. “Analysis of acyclic attenuating systems for the SSR protection model,” appears in the Proceedings of the 1985 IEEE Symposium on Security and Privacy (Oakland, Calif., Apr.). IEEE, New York, 1985, pp. 197-206.0 1985 IEEE.

[176] [Sandhu 1988] Sandhu, R. S. “Schematic Protection Model.” Journal of the Association for Computing Machinery, Vol. 35, No. 2, April 1988.

[177] [Sandhu 1994] Sandhu, R. S. and P. Samarati 1994. “Authentication, Access Control, and Intrusion Detection”. Part of the paper appeared under the title "Access Control: Principles and Practice" in IEEE Communications 32(9): 40-48.

[178] [Sandhu 1996] Sandhu, R. S., E. J. Coyne, Et al. 1996. “Role-Based Access Control Models.” IEEE Computer 29(2): 38-47.

[179] [Sandhu 1998] Sandhu, R. S. 1998. “Role Activation Hierarchies”. Third ACM/NIST Role Based Access Control Workshop, Fairfax, Virginia, USA, ACM Press.

[180] [Sandhu 2000] Sandhu, R., D. Ferraiolo, Et al. 2000. “The NIST Model for Role-Based Access Control: Towards A Unified Standard”. 5th ACM Workshop on Role-Based Access Control, Berlin, Germany.

[181] [SDSI] SDSI, http://theory.lcs.mit.edu/~cis/sdsi.html.

[182] [Sergot] Sergot, M. J., F. Sadri, Et al. 1986. “The British Nationality Act as a logic program”. Communications of the ACM(29): 370-386.

[183] [Sloman 1994] Sloman, M. S. 1994. ”Policy Driven Management for Distributed Systems”. Journal of Network and Systems Management 2(4): 333-360.

[184] [Sloman 1995] Sloman, M. 1995. “Management Issues for Distributed Services”. Proc. IEEE Second International Workshop on Services in Distributed and Networked Environments (SDNE 95). Whistler, Canada.

[185] [Sloman 2002] Sloman, M. and E. Lupu. 2002. “Security and Management Policy Specification”, IEEE Network Magazine, March/April 2002, pgs. 10 – 19.

[186] [Smith 2004] M. Smith et al 2004. “OWL Web Ontology Language Guide”, W3C.

[187] [Snir] Snir, Y.,Y. Ramberg, Et al. 2001. Policy QoS Information Model.

[188] [Spivey] Spivey, J. M. 1989. “An Introduction to Z and Formal Specifications”. IEE/BCS Software Engineering Journal 4(1): 40-50.

[189] [Steen 2000] Steen, M. W. A. and J. Derrick 2000. “ODP Enterprise Viewpoint Specification”. Computer Standards and Interfaces 22: 65-189.

[190] [Steen 1999] Steen, M. W. A. and J. Derrick 1999. “Formalising ODP Enterprise Policies”. 3rd International Enterprise Distributed Object Computing Conference (EDOC '99), University of Mannheim, Germany, IEEE Publishing.

[191] [Stone] Stone, G. N., B. Lundy, Et al. 2001. “Network Policy Languages: A Survey and a New Approach”. IEEE Network: 10-20.

[192] [Strassner] John Strassner, Ed Ellesson. 1999. “Terminology for describing network policy and services”. Internet Draft. Internet Engineering Task Force.

[193] [Su] Su, S. Y. W., H. Lam, Et al. 2001. “An Information Infrastructure and E-services for Supporting Internet-based Scalable E-business Enterprises”. 5th IEEE Annual Enterprise Distributed Object Conference (EDOC2001), Seattle, WA, IEEE Computer Society.

[194] [Tennenhouse] Tennenhouse, D. L., J. M. Smith, Et al. 1997. “A Survey of Active Network Research”. IEEE Communications Magazine 35(1): 80-86.

[195] [Thomas] Thomas, R. K. 1997. “Team-based Access Control (TMAC): A Primitive for Applying Role-based Access Controls in Collaborative Environments”. Second ACM/NIST Role Based Access Control Workshop, Fairfax, Virginia, USA, ACM Press.

[196] [Thomsen] Thomsen, D., D. O'Brien, Et al. 1998. “Role Based Access Control Framework for Network Enterprises”. 14th Annual Computer Security Applications Conference.

[197] [Tonti] Tonti, G., J.M. Bradshaw, R. Jeffers, and R. Montanari. “Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KAoS, Rei, and Ponder”, 3rd International Semantic Web Conference (ISWC2004), Nov. 7 – 11, 2004, Hiroshima, Japan.

[198] [Ungureanu] Ungureanu, V. and N. H. Minsky 2000. “Establishing Business Rules for Inter-Enterprise Electronic Commerce”. 14th International Symposium on Distributed Computing (DISC 2000), Toledo, Spain, Springer-Verlag.

[199] [Ungurenanu 1998] Ungureanu, V. and N. H. Minsky 1998. “Unified Support for Heterogeneous Security Policies in Distributed Systems”. 7th USENIX Security Symposium, San Antonio, Texas.

[200] [VA 2006] United States Department of Veterans Affairs, 2006, “Role-Based Access Control (RBAC) Website” (http://www.va.gov/rbac/)

[201] [Van Gelder] Van Gelder, A. 1988. “Negation as Failure Using Tight Derivations for General Logic Programs”. Foundations of Deductive Databases. J. Minker. San Mateo, CA, Morgan Kaufmann: 149-176.

[202] [Van Lamsweerde 1995] Van Lamsweerde, A., R. Darimont, Et al. 1995. “Goal-Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt”. 2nd IEEE Symposium on Requirements Engineering (RE '95), York, UK, IEEE Computer Society Press.

[203] [Van Lamsweerde 1996] Van Lamsweerde, A. 1996. “Divergent Views in Goal-Driven Requirements Engineering”. ACM SIGSOFT Workshop on Viewpoints in Software Development, San Francisco, ACM.

[204] [Van Lamsweerde 1999] Van Lamsweerde, A. 1999. “Goal-Oriented Requirements Analysis with KAOS (Presentation)”. Policy Workshop 1999, HP-Laboratories, Bristol, UK.

[205] [Verma 2001a] Verma, D., M. Beigi, Et al. 2001. “Policy Based SLA Management in Enterprise Networks”. Policy Workshop 2001, HP Labs, Bristol, UK, Springer-Verlag.

[206] [Verma 2001b] Verma, D. C. 2001. Policy-Based Networking: Architecture and Algorithms, New Riders Publishing.

[207] [Virmani] Virmani, A., J. Lobo, Et al. 2000. “Netmon: network management for the SARAS softswitch”. 2000 IEEE/IFIP Network Operations and Management Seminar (NOMS 2000), Hawaii.

[208] [Von Wright] Von Wright, G. H. 1951. “Deontoc Logic”. Mind 60: 1-15.

[209] [Wenning 2006] Rigo Wenning (ed.) 2006. “The Platform for Privacy Preferences 1.1 (P3P1.1) Specification”, W3C.

[210] [White 2003] White, Tony and E. Bacic, “Implementing Policy-based Content Filtering for Web Servers”, 2003 International Conference on Security and Management. June 2003. Las Vega, Nevada.

[211] [Wieringa] Wieringa, R. J. and J.-J. C. Meyer 1998. “Applications of Deontic Logic in Computer Science: A Concise Overview”. Practical Reasoning and Rationality (PRR 98), Brighton, UK, John Wiley & Sons.

[212] [Yao 2003] W. Yao 2003. “Trust Management for Widely Distributed Systems”. Jesus College, University of Cambridge,

[213] [Yuan 2005] Eric Yuan, Jin Tong 2005. "Attribute Based Access Control, A New Access Control Approach for Service Oriented Architectures". Proceedings, New Challenges for Access Control Workshop, 2005

[214] [Zeber 2002] Zeber, Dr. S., and A. Magar, “Managing Identity and Access in the Defence Environment”, Proceedings of the 7th International Command and Control Research and Technology Symposium, September 2002.

[215] [Zeber 2006] Zeber, Dr. S., et al. (2006). “Secure Access Management for Secret Operational Networks (SAMSON) Concept of System Operation”. DRDC Ottawa TM 2006-119, June 2006, Defence R&D Canada - Ottawa.

[216] [Zhang]Zhang, Xinwen, F. Parisi-Presicce, R. Sandhu, and J. Park. “Formal Model and Policy Specification of Usage Control”, ACM Transacctions on Information and System Security (TISSEC), Vol. 8, No. 4. November 2005, pgs 351-387.

# List of abbreviations & acronyms

| | |
|---|---|
| ABAC | Attribute-based Access Control |
| ACL | Access Control List |
| ANSI | American National Standards Institute |
| API | Application Programming Interface |
| ASL | Authorization Specification Language |
| CAD | Computer Aided Design |
| CDSA | Common Data Security Architecture |
| CMD | Client Meta Database |
| COTS | Commercial-off-the-shelf |
| DAC | Discretionary Access Control |
| DAML | DARPA Agent Markup Language |
| DMTF | Distributed Management Task Force |
| DND | Department of National Defence |
| DRDC | Defence R & D Canada |
| DRM | Digital Rights Management |
| DSD | Dynamic Separation of Duty |
| EACL | Extended Access Control List |
| ECA | Event-Condition-Action |
| EDAC | Enterprise Dynamic Access Control |
| EJB | Enterprise JavaBeans |
| EPAL | Enterprise Privacy Authorization Language |
| ERM | Enterprise Rights Management |
| ETR | Event-Trigger-Rule |
| EU-IST | European Union Information Society Technologies |
| GAA | Generic Authorization and Access-control |
| GBAC | Governance-based Access Control |
| GIG | Global Information Grid |
| GSS | Generic Security Service |
| GTRBAC | Generalized Temporal Role-based Access Control |
| GUI | Graphical User Interface |

| | |
|---|---|
| HIPAA | Health Insurance Portability and Accountability Act |
| HTTP | Hypertext Transfer Protocol |
| IBAC | Identity-based Access Control |
| ID-FF | Identity Federation Framework |
| IDS | Intrusion Detection System |
| ID-SIS | Identity Services Interface Specifications |
| ID-WSF | Identity Web Services Framework |
| IETF | Internet Engineering Task Force |
| INCITS | International Committee for Information Technology Standards |
| IP | Internet Protocol |
| ISA | Instrumentation, Systems, and Automation Society |
| ISO | International Organization for Standardization |
| IT | Information Technology |
| KRS | Key Release Server |
| LDAP | Lightweight Directory Access Protocol |
| MAC | Mandatory Access Control |
| MBAC | Metadata-based Access Control |
| MeSMO | Meta Security Model |
| MIB | Management Information Base |
| MMI | Mother-May-I |
| MPEG | Moving Picture Experts Group |
| NIST | National Institute of Standards and Technology |
| OASIS | Open Architecture Security for Information Systems |
| OASIS | Organization for the Advancement of Structured Information Standards |
| ODP | Open Distributed Processing |
| ODRL | Open Digital Rights Language |
| ODP-RM | Open Distributed Programming Reference Model |
| OIL | Ontology Inference Layer |
| OWL | Web Ontology Language |
| PAC | Privilege Attribute Certificate |
| PAMINA | Privilege Administration and Management INfrAstructure |
| PAP | Policy Authorization Point |
| PC | Personal Computer |
| PDP | Policy Decision Point |

| | |
|---|---|
| PDL | Policy Description Language |
| PEP | Policy Enforcement Point |
| PIPEDA | Personal Information Protection and Electronic Documents Act |
| PKI | Public Key Infrastructure |
| PMI | Privilege Management Infrastructure |
| QoS | Quality of Service |
| RAdAC | Risk Adaptive Access Control |
| RBAC | Role-based Access Control |
| RDD | Rights Data Dictionary |
| RDF | Resource Description Framework |
| RDL | Role Definition Language |
| REFEREE | Rule-controlled Environment for Evaluation of Rules, and Everything Else |
| REL | Rights Expression Language |
| RES | Rules Engine Service |
| RFC | Request For Comments |
| RM | Resource Manager |
| RSL | Role Specification Language |
| RuleML | Rule Markup Language |
| SAML | Security Assertion Markup Language |
| SAMPOC | Secure Access Management Proof-Of-Concept |
| SAMSON | Secure Access Management for Secret Operational Networks |
| SDK | Software Development Kit |
| SDL | Standard Deontic Logic |
| SDSI | Simple Distributed Security Infrastructure |
| SESAME | Secure European System for Applications in a Multi-Vendor Environment |
| SNMP | Simple Network Management Protocol |
| SPKI | Simple Public Key Infrastrcuture |
| SPL | Security Policy Language |
| SQL | Structured Query Language |
| SSO | Single Sign-On |
| SULTAN | Simple Universal Logic-oriented Trust Analysis Notation |
| SSD | Static Separation of Duty |
| SWRL | Semantic Web Rule Language |
| TCB | Trusted Computing Base |

| | |
|---|---|
| TD | Technology Demonstrator |
| TMAC | Team-based Access Control |
| TPL | Trust Policy Language |
| UCON | Usage Control Model |
| UML | Unified Modeling Language |
| URL | Universal Resource Locators |
| VPN | Virtual Private Network |
| W3C | World Wide Web Consortium |
| WS | Web Services |
| WS-I | Web Services Interoperability Organization |
| XACML | eXtensible Access Control Markup Language |
| XML | eXtensible Markup Language |
| XrML | eXtensible Rights Markup Language |
| XSL | eXtensible Style Language |

**DOCUMENT CONTROL DATA**

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

| | |
|---|---|
| 1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)<br><br>Cinnabar Networks ( Bell Security Solutions Inc.)<br>265 Carling Avenue, Suite 200<br>Ottawa ON K1S 2E1 | 2. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.)<br><br>Unclassified |

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C, R or U) in parentheses after the title.)

Options for the Policy Server Component of the DRDC Architecture for Secure Access Management: Revision 2006

4. AUTHORS (last name, followed by initials – ranks, titles, etc. not to be used)

Bacic, E., Klump, S., Magar, A.,

| | | |
|---|---|---|
| 5. DATE OF PUBLICATION (Month and year of publication of document.)<br><br>August 2006 | 6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.)<br><br>141 | 6b. NO. OF REFS (Total cited in document.)<br><br>216 |

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Contract Report

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

NIO Section, DRDC Ottawa

| | |
|---|---|
| 9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)<br><br>15BQ03 | 9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)<br><br>W7714-6-3292 |
| 10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.) | 10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)<br><br>DRDC Ottawa CR 2006-166 |

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

( X ) Unlimited distribution
(   ) Defence departments and defence contractors; further distribution only as approved
(   ) Defence departments and Canadian defence contractors; further distribution only as approved
(   ) Government departments and agencies; further distribution only as approved
(   ) Defence departments; further distribution only as approved
(   ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.))

Full Unlimited

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

**Defence R&D Canada – Ottawa** (DRDC Ottawa) implemented a proof-of-concept system that combines Privilege Management Infrastructure (PMI) technology and Public Key Infrastructure (PKI) technology to demonstrate a caveat separation capability for the defence environment. A key component of the demonstrated system is a policy server product that provides content-based security. This report examines alternative products and solutions for the implemented policy server that would be consistent with the existing policy component of the proof-of-concept system. This report discusses existing technologies from industry, academia, the military, and research laboratories as well as the possibilities and complexities of designing and implementing a work-alike replacement. This report, which is a 2006 revision of a report originally written in 2003, documents progress in policy research over the ensuing three year period.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Access control policy, authorization, Bell-LaPadula, computer policy, network policies, policy, policy-based management, policy decision, policy distribution, policy enforcement, policy engines, policy implementation, policy interpreter, policy languages, policy logic, policy mediation, policy models, policy processing, policy products, policy research, policy specification, policy specification language, policy standards, programmable policies, security policies