A REFERENCE MODEL FOR ENTERPRISE SECURITY
High Assurance Enterprise Security

David W. Enström, D'Arcy Walsh
Communications Security Establishment, 719 Heron Road, Ottawa , Canada
David.Enstrom@cse-cst.gc.ca, D'Arcy.Walsh@cse-cst.gc.ca

Siavosh Hossendoust
Enterprise Security Architect, IBM, 2220 Walkely Road, Ottawa, Canada
siavoshh@ca.ibm.com

Keywords: High Assurance Enterprise Security Architecture

Abstract: This paper defines an enterprise security model that provides a cohesive structure for the definition and implementation of security services. The complete framework is described, but with a focus on subjects, and protected objects and how access is controlled. Multiple layers of security are defined, building upon the "defence in depth" concept, augmented with "domain" and "zone" concepts and associated protections. The dynamic use of roles is described, a concept that along with user self–service provides a practical approach for the management and use of roles for access control. This model may also be used as a reference architecture for the definition and integration of a set of security services that permit multiple vendor implementations to work together, and to establish the level of compliance of specific systems.

1 INTRODUCTION

Enterprise security architecture has been a patchwork of isolated and tactical solutions that solve specific security problems. However, many security problems have been neglected due to the complexity and diversity of these security problems and solutions, resulting in the inability to build overarching enterprise security architecture.

This paper describes an enterprise security reference model that unifies different aspects of security. It was formulated because no one reference provides a clear comprehensive framework for enterprise IT security. The model provides clear divisions of responsibility for the implementation and integration of security services that provide the appropriate level of protection in accordance with defined policies. The model is applicable to a specific part of a company, across the corporation or across a multinational corporation as desired.

This reference model is in general based upon RBAC, Role Based Access Controls (Ferraiolo and Kuhn 1995; Kendall 1999; Yoder and Barcalow 1997). The base objective is to provide access to services and information to those that need it, and mitigate access to those that don't. A unique feature of this reference model is its dynamic aspect. The dynamic sessions that a user establishes with services are included in the model, providing great flexibility and manageability of user permissions.

This model is also based upon the "defence in depth" concept whereby users must pass through multiple hurdles or checkpoints before gaining access to information (CSE, n.d.). The level of protection provided at each layer may be tailored to match the situation, sensitivities, policies and risks that need to be mitigated.

This model may also be used as a reference architecture for the definition and integration of a set of security services that permit multiple vendor implementations to work together, and to establish the level of compliance of specific systems.

2 THE REFERENCE MODEL

2.1 Protected Objects

At the heart of the security model is a Protected

Object, which represents all of the sensitive items that need to be protected. They need to be protected due to their inherent sensitivity (i.e. their classification label), or because they are important components within the environment.

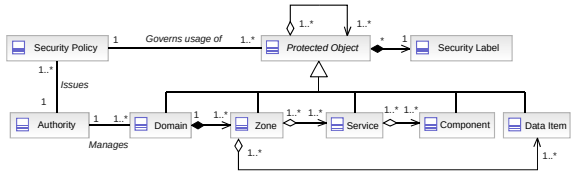

Figure 1 – Protected Object Model.

Protected objects must exist within a context defined by the authority and the policies that the authority issues. Without this context the sensitivity of the protected objects and therefore the protections required would be unclear. The model must provide protections for *all* objects, namely domains, zones, the services, components and data items.

The relationship between security policy and a protected object is a key aspect of the model. It defines how domains, zones, services, components and data items must behave to meet policy. It also provides the context under which domains and zones (and sub–domain and sub–zones) are defined. A domain is defined to ensure that rules defined by the authority are understood and enforced within the required scope (e.g. corporate intranet policy, or corporate extranet policy). Zones and sub–zones are defined to ease the management and enforcement of a sub–set of the policy.

2.1.1 Domains

A domain is a higher–level construct, concerned with security policies, security models, and security architecture, including a set of resources and system entities that are authorized to access the resources. In other words, domains define "like" security environments, such as an organization's intranet *or* its extranet. Policy is a key concept for the definition of a domain. Domains may be extended to allow for policy extensions. A domain (e.g. A corporate extranet) may also contain sub–domains (e.g. A portion of the extranet used by a particular business line) to permit policy refinements.

Through the provision of a layer of access control at domain boundaries, the security required within the domain is simplified. This is equivalent to providing tight physical access control to a building, which simplifies the security implementation within the building.

A domain is used for very coarse–grained access control based upon the policy issued by the authority. Access control between domains (e.g. between corporations) is usually enforced at the network layer, through various mechanisms, including Firewalls and other network (and perhaps applications) level protections. A demilitarized zone (or DMZ) implements these protections in a well–defined and standard manner. This reference model includes this in its approach, through the definition of a Domain Policy Enforcement Point.

2.1.2 Zones

A second important concept is that of a zone. Zones are used to partition a domain, making it easier to implement security controls that enforce policies within and between them. Again, this is equivalent to providing strong physical access to a room, which simplifies security practices within the room. Unlike a Domain, the concept for zone is not based upon an authority and the policy that it defines, but upon enforcement of specific aspects or instances of policies within the domain (e.g. security services).

A zone is used for coarse–grained access control for specific policies, enforced by "Zone Policy Enforcement Points". Service level controls (i.e. User accounts) may also be used to control access to the services and information within zones. See 2.3.2.

Zones may be sub–divided into smaller zones. The objective is to group like–services and systems together — "like" in this context, just as in the high–level zones, means services, components and data items at the same level of sensitivity and policy.

A zone is also defined to control access from zone to zone. They define a logical separation that is normally implemented at the network layer or above, through various mechanisms, providing the coarse–grained access control required.

A description of the concepts/classes in Figure 1:

Authority – Person (e.g. corporate head of policy) who issues and manages the security policies that applies to a domain, as well as operates and manages the domain itself (i.e. they have complete control over the domain and its associated policy). An authority may manage multiple policy sets (e.g. extranet policy and the intranet policy) and associated domains (e.g. Extranet and Intranet).

Security Policy – The rules defining needed levels of information security to achieve the desired confidentiality and integrity goals. A policy is a statement of information values, protection

responsibilities, and organization commitment for a system (e.g. The Corporate Extranet Policy).

Protected Object – An object within a domain that, from a security perspective, needs protection. This protection is required either because of the intrinsic sensitivity of the object (e.g. highly classified information) or due to its position within the infrastructure (e.g. a directory or firewall).

Security Label – The security classification of a protected object. It defines the mandatory access control information, and the sensitivity of the labelled object, i.e. the classification plus other restrictions on the handling of the information (e.g. Confidential//Management Eyes Only).

Domain – An environment or context that is defined by security policies, security models, and security architecture, including a set of resources and system entities that are authorized to access the resources. A domain is managed by a single authority, and may contain one or more sub–domains. Domains are created when security models or policies and architecture are significantly different from one domain to another, or are conflicting. Separate domains provide clearer separation of concerns and ease policy enforcement and management. The traits defining a given domain typically evolve over time (ISO 2002).

Zone – A zone is a subdivision of a domain with a well–defined boundary that has a common level of protection for all objects within its boundary. Zones may be logical or physical and may be defined at any layer within the domain architecture, from physical zones (e.g. building / room), to application–level zones (e.g. application–level access controls and protections). Typically, zones are defined at the physical, network and application layers. A zone can have multiple sub–zones, but can have only one super–zone. Sub–zones inherit their parents' properties (i.e. Policies that apply to the super–zone apply to the sub–zone). Zones cannot span domain boundaries.

Service – A function that is well defined, self–contained, and does not depend on the context or state of other services (e.g. Directory). It is a function packaged as a re–usable component within business processes.

Component – An assembly or part thereof, that is essential to the operation of some larger assembly and is an immediate subdivision of the assembly to which it belongs (e.g. DBMS). In the software context, this may in turn be composed of components. There are also hardware components (e.g. router, switch, PC, etc.).

Data Item – A semantically meaningful unit of information exchanged between two parties to a transaction (e.g. An Invoice).

2.2 User Identity

In the IT security context, identity refers to the creation of an electronic equivalent to a person, a user account for example. Within the system a "picture" of a user is defined that includes required characteristics or attributes; in our model this is called a "subject". It includes characteristics that may be used to relate this electronic representation of the person with the real person (i.e. name, employer, phone number, photo, fingerprint, etc.). Security related metadata is also included, such as login ID and password for example, which in our model is represented as a "credential". A credential is used in establishing the association between the physical person and their electronic representation as a subject (that has credentials).

Within any IT environment a key security feature that needs to be addressed is user identity, which is important in providing or prohibiting access. User identity has two main aspects: One, establishing, at login with an appropriate level of assurance, that the person logging in is associated with the proper subject (i.e. the electronic representation of the person/user), otherwise called authentication (See 3.1.1); Two, defining and managing the various credentials that a user may have within the domain (i.e. Identity Management).

2.2.1 Roles Model

Roles are categories that collect together users who share the same levels of security privilege.

Two types of role are defined, as shown in Figure 2. The *new* classes are defined as follows:

Role – object that represents a collection of permissions (e.g. IT roles) or a collection of users (e.g. user roles), and that generally relates to a class of business functions. They collect together users who share the same levels of security privilege; subjects are granted roles in order to obtain particular access permissions. Roles are defined based upon the business functions to be performed (e.g. Financial Officer), or upon general actions (e.g. Administering, Authoring, etc.). There are two types of roles: functional roles and contextual roles.

Functional Role – Role tied to a subject's primary job function within the organization, but not necessarily reported in the human resources system.

Therefore, all organizational roles are functional roles, but not all functional roles are organizational roles (E.g. “Financial Analyst”).

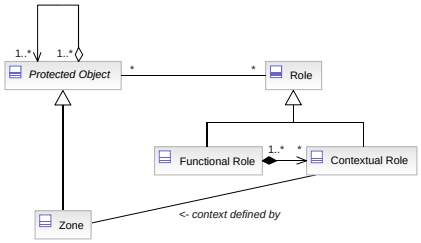

Figure 2 – Roles & Zone Model.

Contextual Role – Role defining permissions in the context of a zone, and particular object within the zone. For a given business object, a contextual role can be granted directly to a user or associated with one of its functional roles. The contextual roles currently defined are *Administering*, *Auditing*, *Authoring*, *Collaborating*, and *Monitoring*.

In order to avoid confusion between functional roles and contextual roles the following conventions have been adopted: Functional and organizational roles will always be expressed as nouns (e.g. TigerTeam); Contextual roles will always be expressed as verbs (e.g. collaborating). This rule reflects the core differences between the role types: functional (and organizational) roles really do refer to persons, places, or things; contextual roles are actions performed by a subject on an object.

Roles are used as the fundamental method for defining access permissions to protected objects.

2.2.2 Static Model of Identity

A standard model is used for “identity”, where notions of a subject (user), credentials and roles are utilized. The relationships between these aspects of identity (and RBAC) are illustrated in Figure 3. This is the static view of identity, that is, it describes a user’s identity as statically defined within the domain (i.e. before the subject establishes sessions). The *new* classes in Figure 3 are:

Subject – An entity that is represented within a security domain by one or more identities from trusted authorities (e.g. driver’s license and a health card). A subject has associated security information such as security clearances. Subjects may have many roles, which in turn may have associated identity and credential information (e.g. passwords). Note that a subject does not necessarily equate to a human being, it may define a service within the domain.

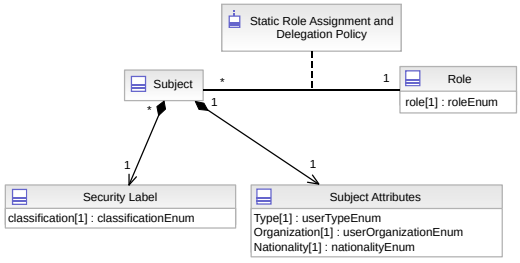

Figure 3 – Static Identity Model.

Static Role Assignment & Delegation Policy – the policy that governs the allocation of roles to subjects at the time of their enrolment within a domain (e.g. all employees get HRInfoUpdate role).

Subject Attributes – Defines additional static characteristics of subjects, such as the type of user (i.e. Full–time, term, contractor, etc.), their affiliation (i.e. IBM, CSE, etc.) and nationality (i.e. CAN, USA, etc.).

2.2.3 Dynamic Model of Identity

Specific access rights are provided to a subject through the granting of one or more roles, either when a user is defined within the domain, or dynamically as sessions are established with services. Note that roles define access rights within a *single* domain, zones in the domain, and provide fine–grained access rights to services and data items.

In order to support the principle of “least privilege” (a subject possesses just enough rights to do their job), roles and thus access rights may be assigned when subjects establish, through secure mechanisms, sessions with services, normally *within* a domain; however sessions with services in *external* domains are also supported. This dynamic aspect of roles is shown in the model in Figure 4.

A description of the *new* classes in Figure 4:

Session – A lasting connection, using a network protocol, between a user (or service) and a peer, typically a service, usually involving the exchange of many packets of information (e.g. an HR application). Session may be stateless or stateful.

Dynamic Role Assignment and Delegation Policy – the policy that governs the allocation of roles to subjects when sessions are established with services within a domain (e.g. Users with the Analyst role get the Update role when establishing a session with the Financial Analysis application).

Session Establishment Policy – the policy that governs the establishment of a session with services within a domain. Subject attributes and credentials

are used in determining if a session is allowed or not (e.g. Users with the Analyst role may establish a session with the Financial Analysts application).

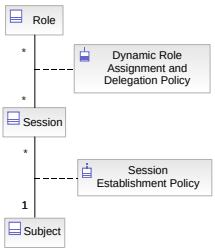

Figure 4 – Dynamic Identity Model.

2.3 Authority & Policy Enforcement

2.3.1 Authority and Policy

Figure 5 illustrates the generic model of authority, policy and policy enforcement. The *new* classes in Figure 5 are:

Policy Administration Point – An entity that manages and issues policies to a policy decision point (e.g. the policy administrator for the domain). Also see *authority*.

Policy Decision Point – An entity that evaluates an access request against one or more policies to produce an access decision. Access decisions are based upon attributes of subjects (including from their sessions/roles) and attributes of the protected object (its security classification, etc.) plus the rules governing the comparison/usage of these attributes.

Policy Enforcement Point (PEP) – An entity that enforces access control for one or more resources (protected objects). When attempting to access a protected object, a PEP sends an access request describing the attempted access to a policy decision point (PDP). The PDP returns an access decision that the PEP then enforces.

2.3.2 Policy Enforcement Points

All protected objects need protection, and therefore there are protection services, or Policy Enforcement Points (PEP), defined that provide this protection. They are in effect the layers of the security onion (i.e. Defence in depth). Figure 6 shows the types of PEPs required and how they relate to other objects in the model. There is a one–to–one correspondence between protected object types and the types of PEPs required and defined.

Simply knowing that a subject has the clearances to access an object is not sufficient for our purposes, the *type* of permissions granted to the subject needs to be understood. Also, permissions may be different depending upon the way in which the information or service is accessed. A local user may have complete access to resources, but the same user accessing services remotely (e.g. Telework) will have reduced capabilities. A dynamic model is required.

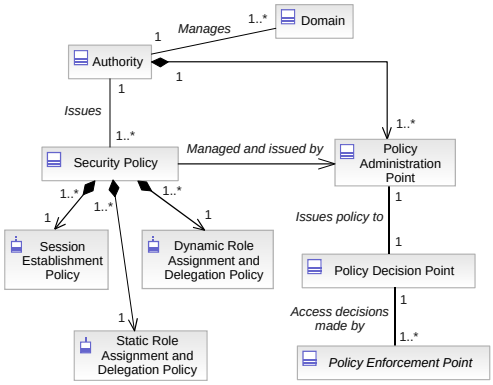

Figure 5 – Authority and Policy Model.

Roles provide fine–grained access control, and may be defined to mitigate access to specific services, functions or information. Other attributes such as the users "locale" must also be used to determine access permissions. The rules used to make access decisions based upon roles and other attributes may get fairly complex. They also need to be consistent across the domain. One way to ensure this consistency is though definition of a single logical "authorization" service (PDP), which uses subject attributes plus the protected objects' attributes to make access control decisions for PEPs.

The *new* classes in Figure 6 are:

Domain PEP – The interface point that provides secure access and interoperability between different domains (e.g. A DMZ with proxies, IP filtering and Firewalls). A domain may have more than one PEP, permitting different levels of access (based upon policy) and for non–functional reasons.

Zone PEP – The interface point that provides secure access and interoperability to protected objects within a zone (e.g. intelligent proxy). A zone may have more than one zone PEP, for different levels of access and for non–functional reasons.

Service PEP – The interface point that provides secure access to a service and its methods (e.g. a Proxy). A service may have more than one PEP, permitting different levels of access (based upon policy) and for non–functional reasons.

Component PEP – The interface point that provides secure access to a component and its methods (e.g. an authorization step when accessing the component). A component may have more than one PEP, permitting different levels of access (based upon policy) and for non–functional reasons.

Data Item PEP – The interface point that provides secure access to a data item (e.g. an authorization step when accessing the data). A data item may have more than one PEP, permitting different levels of access (based upon policy) and for non–functional reasons.

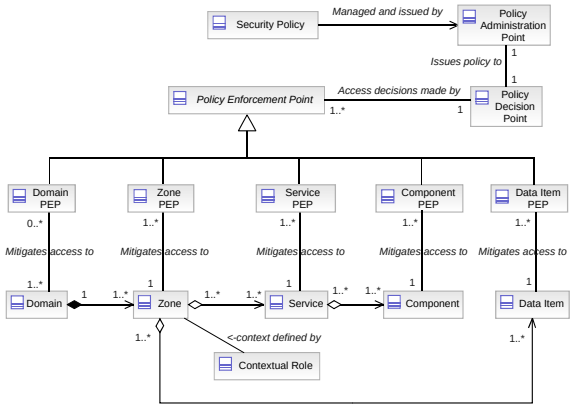

Figure 6 – Types of Policy Enforcement Points.

2.4 Access Control

Access restrictions ensure that only authorized Subjects are granted access to a Protected Object within the Domain. Many mechanisms are used to control access for security purposes. Access controls are defined at the physical (i.e. buildings/rooms), network and application layers.

Physical level access controls are beyond the scope of this model, however the physical access controls to buildings and offices need to be considered when defining the IT security model (i.e. deciding on security approaches and technology given the level of physical access controls).

Network level access controls are included in this model through the definition of domain and zone boundaries, which enforce coarse–grained access controls. Access to services and information within a domain or zone from other domains or zones is controlled through various protections at these boundaries (e.g. firewalls, IP address filtering, proxy services, etc.) and protections provided at the service, component and data item layers. The specifics of implementation are beyond the scope of this logical model. The IT Security Zones Baseline Security Requirements was used to help formulate the solution defined here (CSE, 2003).

Application level access controls are included through the definition of a single authority (i.e. a Policy Decision Point), policy enforcement points, authorization services and other mechanisms, which define fine–grained access control. Access decisions are made based upon a set of rules that use attributes, roles, and other information about subjects, zones and protected objects (OASIS 2005a). In some cases, access decisions may be pre–computed, resulting in a permission set for the subject, which is used to control access.

2.4.1 Basic Access Control

Some policies need to be enforced in all systems within the domain; they are hard givens. This includes a standard set of labels used to indicate the sensitivity of the object. Labels on objects are typically subdivided into a classification, community of interest (COI) and national dissemination. The classification is used for mandatory access control, and COIs, & national dissemination used for discretionary access control.

Both users and protected objects are labelled: object labels indicate their sensitivity; user labels indicate the clearances they possess. In order for access to be granted the security properties of the subject must *minimally* match the security properties placed upon the entity accessed. All users within a domain may be permitted to handle “Company Confidential” material, but the need–to–know principle must still be enforced, through the use of roles, COIs and other attributes.

2.4.2 Static Model of Access Control

Subjects are assigned basic attributes when they are enrolled within the domain. These include their security clearances and other information contained in their credentials such as public and private keys. Roles may also be given to users at this time. This static view of a Subject in the context of access control is shown in the model in Figure 7.

The *new* classes in Figure 7 are:

Credential – A token (e.g. biometric data) or a shared secret provided by a subject that imparts confidence in the claimed identity of the subject within a domain. A trustworthy authority issues credentials. Credentials are created as a result of a successful authentication, and are a set of one or

more access tokens that will allow a principal to connect to a domain or other object. E.g. user ID/password combination, PKI certificate, Kerberos ticket, etc.

Protected Object Attributes – Defines additional characteristics of protected objects such as the "Author" and "Owner".

Permission – The authority–based right that allows a subject to perform an action (such as read, write, delete, execute, or create) on a protected object. Access is granted (provided mandatory access controls are met) only by associating a permission with a role. Permissions do not override mandatory access controls. The usage of permissions is well covered in OASIS (2005a).

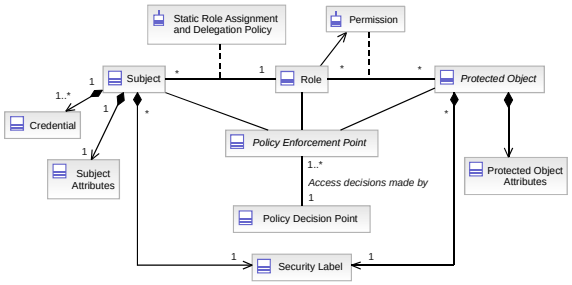

Figure 7 – Static Access Control Model.

The rules governing access may get quite complex, every effort should be made to simplify models and the rule set. This in part is accomplished through the definition and implementation of "zones", but also the intelligent use of roles, as in Basin and Doser (2005).

2.4.3 Dynamic Model of Access Control

Determining whether or not a user has access to a particular protected object based upon static identities and credentials (i.e. Domain login–time) does not address all requirements. Users, during the course of their day, need to perform various activities with many different zones and services, and perhaps other domains as well. This dynamic aspect, and the concept of "least privilege", requires a dynamic solution. The solution is to associate roles with sessions, so that a user may be assigned more privileges when a session with a particular service is established. This demands proper controls over the establishment of sessions. This dynamic model of access control is shown in Figure 8.

Additional privileges are acquired as needed through contextual role(s) that are assigned when sessions are established. Each role has associated permissions.

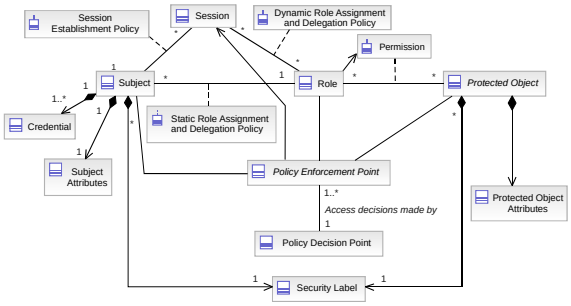

Figure 8 – Dynamic Model of Access Control.

2.5 Audit and Logging

Auditing is the collection of event information about PEP, service, component and data item activities that affect security. Services, components, PEPs and data items may utilize the audit service to capture audit events. These events may be informational (e.g. User *x* logged in) or they may represent security exceptions (e.g. Unauthorized user attempted to access resource *y*).

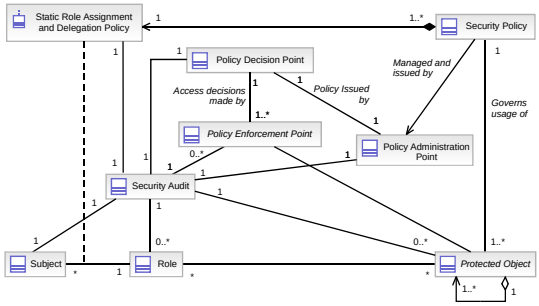

Figure 9 – Static Model of Audit.

The audit and logging model provides a framework for the capture of security event information in logs. For performance reasons, the security framework needs to be circumspect in what events need to be logged. At the same time it must be recognized that the requirements for audit and logging will vary depending upon the sensitivity (i.e. contents of the security label) of the information being processed. As a result, the audit and logging model is a flexible one, allowing the specification of audit assertions that are applied within the context of subjects, roles, policy and protected objects.

As with the other aspects of the security model there is both a static view and a dynamic view of

audit. The static view is shown in Figure 9. The *new* classes in Figure 9 are:

Security Audit – implements the logging of security events and information for and about the protected objects and other sensitive information within a domain. Audit information is always traceable to the subject involved in the activity. Logging of events may be turned on and off as directed by security personnel.

Audit administrators (i.e. a subject with this role) can configure the set of events to be audited. These events are based upon commands (behaviours) defined within PEPs, services, components and data items and their utilization of the audit service. Audit events may be turned on and off in a coarse or fine–grained manner; they can be controlled at the zone, PEPs, service, component and data item levels.

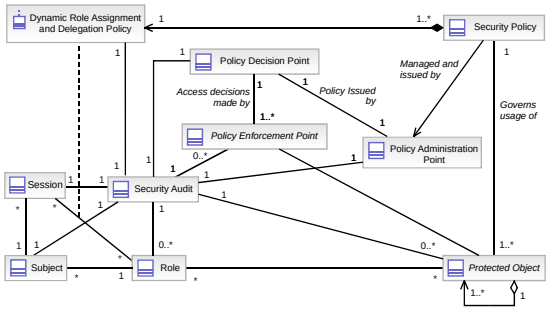

Figure 10 – Dynamic Model of Audit.

The dynamic view of audit is shown in Figure 10. The only changes to this model of audit (compared to Figure 9) are the addition of the "Dynamic Role Assignment and Delegation Policy", and the removal of the "Static Role Assignment and Delegation Policy" (for clarity purposes).

Many details regarding audit need to be addressed at implementation, such as the required information to be captured in the logs. This model defines the relationships that need to be supported and the audit controls that are needed so that correlations and other analysis may be performed. The audit process needs to support domain policy and verification of its application within the domain.

3 SUMMARY RELATED/FUTURE WORK AND CONCLUSION

3.1 Summary

The scope for roles is the domain, but cross–domain interoperability is required (i.e. Enterprises interoperate with suppliers, partners and clients). This is accomplished through the establishment of "trust" between domains, and through the standardization of credentials. The extension of trust is done through a Public Key Infrastructure (PKI) being cross–certified with the other parties (i.e. supplier and partner) or via a third party. The standardization of credentials is also needed (i.e. Subject/object attributes), along with the ability to share them as in OASIS (2005b).

There is a trade–off at implementation regarding the level of protection provided at each of the various security layers (i.e. PEPs). For example, one may provide very strong zone protection (e.g. A Firewall implementation of a zone PEP), which may negate protection of the data at rest within the zone.

The model wouldn't be complete without some clarification of the authentication, confidentiality, availability and integrity aspects of IT security and how they relate to this model.

3.1.1 Authentication

Authentication is the *process* of verification of a subject's identity, along with the provision of basic access to the domain with which the authentication is done (i.e. Static subject attributes and roles are provided). Authentication therefore is a domain PEP providing verification of the identity of the subject, to some desired level of assurance, and providing the subject's basic access permissions.

Authentication is a necessity at the domain level, but other levels may require authentication, namely at the zone, service, component and data item (PEP) levels. Similar mechanisms may be implemented, however practicalities may restrict what can be done at these other levels. Ideally once the subject has authenticated to the domain, the rest is transparent or not required.

3.1.2 Confidentiality

Confidentiality, in its broadest definition, is the protection of information from unauthorized access. This definition can encompass the need for all types of PEPs; however it commonly is limited to the protection of data in transit or of data at rest. Given this more specific definition, the need for confidentiality services is very much policy driven. Protection of data in transit and at rest are typically done through the application of encryption, but specific solutions like this are beyond the scope of

this logical model. A summary follows, but note that combinations of these solutions may be required.

For data in transit, confidentiality for the following areas needs to be addressed:

- Physical WAN communications
- Virtual circuits (e.g. Sessions, etc.)
- Application layer communications (e.g. Email and attachments, FTP, Web, and other data exchanges)

For data at rest, confidentiality for the following areas needs to be addressed:

- Temporary storage
- Permanent storage

3.1.3 Availability

At the logical level one can only indicate the importance of the various services and components within the architecture. Availability requirements, specified as non–functional requirements, can only be met through application of good design implementation, through provision of redundant hardware or software services and components, which is beyond the scope of this model.

3.1.4 Integrity

The need for integrity spans all aspects of architecture. Integrity may be summarized as the protection of protected objects from malicious or inadvertent modification or misuse. The integrity of the following aspects must be addressed at implementation:

- Data integrity (accuracy & completeness of data)
- Service and component integrity
- PEP integrity
- Network integrity

Many options and approaches to addressing integrity exist, such as digital signatures, encryption, etc. Again, choices need to be made at implementation to address integrity requirements.

3.2 Related Work

The reference model reflects current offerings and practices within the industry (Tulloch 2003; Entrust 2003; IBM 2003, 2005; Bücker et al. 2004; Blakely et al. 2004; Buecker et al. 2004). Concepts and definitions are also aligned with the XACML standards, which are complimentary to the concepts defined here (OASIS, 2005a; OASIS 2005b).

No comprehensive model for enterprise security could be found, which motivated the creation of this reference model. Portions of it are covered in some works, such as Basin, Doser and Lodderstedt (2004) and Miller, Yee and Shapiro (2003) and Indrakshi et al. (2004), which validate respective areas of this comprehensive model. Another important distinction is that the reference model attempts to clearly separate concerns, providing a more flexible and manageable solution.

3.3 Future Work

A complete implementation of this model would provide verification of many aspects. This may result in some re–factoring based upon this experience and permit the identification and elaborations of the roles portion of the model, and protection requirements. Specifically allowing for a more computable transform to implementation based upon the context and requirements (e.g. When to use a firewall vs. network address filtering etc.).

The audit and logging portion of this model will be refined over time. Audit and logging information needs to be well–defined to allow the correlation of events across the enterprise.

Much work has been done on the inclusion of non–functional requirements within UML models, and also ontologically precise UML. Integration of this work within this reference model is desirable.

3.4 Conclusion

CSE has implemented a portion of this model within its core systems. Specifically aspects of protected objects, and zone, service, component & data item PEPs, and the static and dynamic access control mechanisms.

The functional and contextual roles concepts defined in this model are used to attain the fine–grained access controls required. Flexibility is an important benefit of the reference model and its implementation, permitting the dynamic application of changes to access control, as required by the business and users.

The zone model provides many benefits. Having policies inherited from parent zones greatly reduces the level of maintenance of rules. The zone model also permits optimizations, for example service–to–service calls within a highly trusted zone (e.g. a security zone) may be optimized for performance. Security behaviour is tailored for each zone permitting the same code to run unaltered in different zones but with required security policies for each zone applied appropriately.

As threats and vulnerabilities increase, mitigation of the associated risks needs to be done in a more architected (i.e. Structured and cohesive) manner. This model provides this structure, which also enables heterogeneous interoperable implementation of security services. The roles and relationships between security components provide defence in depth, to any strength and depth required for a given situation. The concept and usage of roles in a dynamic manner provides a practical and flexible way to implement fine–grained access controls.

4 ACKNOWLEDGEMENTS

The authors would like to thank Scott Cluett for his valuable insight. Scott's implementation of many aspects of this model provided the grounding required to make the model practical, and provided useful validation of its concepts and structures.

5 REFERENCES

Buecker, Axel, Filip, Werner, Becke, Richard, Cowan, Tony, Godbole, Subodh, Hinton, Heather, Kariyawasam, Sampath, Stranden, Harri, 2004, 'Federated Identity Management with IBM Tivoli Security Solutions', *IBM Redbooks*, First Edition.

Basin, David, Doser, Jürgen, Lodderstedt, Torsten, 2004, 'Model Driven Security: From UML Models to Access Control Infrastructures', *ACM Transactions on Software Engineering and Methodology*, Vol. 15, No. 1, pp. 39–91.

Blakely, Bob, Heath, Craig & members of The Open group Security Forum, 2004, *Technical Guide – Security Design Patterns*, viewed January 2005, <http://www.opengroup.org/bookstore/catalog/g031.htm>.

Bücker, Axel, Gontarczyk, Andrew, Heiser, Mari, Karekar, Santosh, Saunders, Patricia, Taglioni, Matteo, 2004, 'Enterprise Security Architecture Using IBM Tivoli Security Solutions' *IBM Redbooks*, Second Edition.

Clarke, Siobhán, Harrison, William, Ossher, Harold, Tarr, Peri, 1999, 'Subject–Oriented Design: Towards Improved Alignment of Requirements, Design and Code', *OOPSLA '99 Proceedings*, pp. 325–339.

CSE, 2003, *IT Security Zones Baseline Security Requirements*, Sue Greaves, Communications Security Establishment, Ottawa.

CSE, n.d., *Introduction to Information Technology Security*, viewed January 2006, <http://www.cse-cst.gc.ca/tutorials/english/section1/m1/index_e.htm>.

Entrust, 2003, *Privilege Management Infrastructure Using Getaccess Version 7 – Detailed Architecture Report*, from Contract: W2213-2-6111, CSE, Ottawa.

Ferraiolo, David, Kuhn, Rick, 1995, *An Introduction to Role–Based Access Control*, NIST/ITL Bulletin, viewed January 2006, <http://csrc.nist.gov/rbac/NIST-ITL-RBAC-bulletin.html>.

IBM Microsoft, 2003, *Federation of Identities in a Web Services World*, viewed February 2005, <http://www-128.ibm.com/developerworks/library/specification/ws-fedworld/>.

IBM, 2005, 'Federated Identity Management and Web Services Security with IBM Tivoli Security Solutions', *IBM Redbooks*, Second Edition.

Indrakshi, Ray, Li, Na, France, Robert, Dae–Kyoo, Kim, 2004, 'Using UML to Visualize Role–Based Access Control Constraints', *Proceedings of SACMAT '04*, pp. 115–124.

International Organization for Standardization, 1989, *Information Processing Systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security architecture*, ISO 7498-2:1989, International Organization for Standardization, Geneva.

International Organization for Standardization, 2002, *Security techniques — Security information objects for access control*, ISO/IEC 15816:2002, International Organization for Standardization, Geneva.

Jürgens, Jan, 2002, 'UMLSec: Extending UML for Secure Systems Development', *UML 2002 - The Unified Modeling Language Proceedings*, pp. 412–425.

Kendall, Elizabeth, 1999, 'Role Model Designs and Implementations with Aspect–oriented Programming', *OOPSLA '99 Proceedings*, pp. 353–369.

Lodderstedt, Torsten, Basin, David, Doser, Jürgen, 2002, 'SecureUML: A UML-Based Modeling Language for Model-Driven Security', *UML 2002 - The Unified Modeling Language Proceedings*, pp. 426–441

Miller, Mark, Yee, Ka–Ping, Shapiro, Jonathan, 2003, *Capability Myths Demolished*, Technical Report SRL2003-02, Johns Hopkins University Systems Research Laboratory, Baltimore.

Organization for the Advancement of Structured Information Standards, 2005a, *eXtensible Access Control Markup Language, XACML Version 2.0*, OASIS, Billerica.

Organization for the Advancement of Structured Information Standards, 2005b, *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*, OASIS, Billerica.

Tulloch, Mitch, 2003, *Microsoft Encyclopedia of Security*. Microsoft Press, Redmond.

Yoder, Joseph, Barcalow, Jeffrey, 1997, 'Architectural Patterns for Enabling Application Security, *PloP '97 Proceedings*.