Reprinted from Proceedings of the 2009 IEEE MILCOM Conference. Boston, October 2009

SECURITY MODEL FOR RESOURCE AVAILABILITY – SUBJECT AND OBJECT TYPE ENFORCEMENT

Ole-Erik Hedenstad
Norwegian Defence Research Establishment

ABSTRACT

Confidentiality, integrity and availability are the three basic aspects of information security. The purpose of the paper is to refine the availability dimension.

In the context of security an *object* is the passive entity to be protected. “Object” can refer to both an information object and to a resource, e.g. the program (or service) that retrieves the information and enables access to it. Thus, we make a distinction between *information* and *resource* availability.

We propose a new security model for resource availability called “subject and object type enforcement” (SOTE). The model can express policies for information flow between resources of different administrative domains. It controls the *types* of resources that are allowed to interact. The ability to express the security requirements and conditions a resource must fulfill, is also part of the model. SOTE is a variation of type enforcement. The main difference is that SOTE is a model for information flow control instead of operating system access control.

Type enforcement is well suited for restricting information flows. In particular type enforcements can express intransitive (indirect) information flows. The SOTE model can express such information flow policies at a fine-grained level. This is a prerequisite for flexible and secure information flow in heterogeneous environment where the domains do not implement the same set of security policies and security levels.

We also describe how multiple security models can be combined in order to express a composite security policy for information flow. We combine the classic multilevel security models (Bell-LaPadula and Biba) with the SOTE resource availability model.

1. INTRODUCTION

Confidentiality, integrity and availability are the three basic aspects of information security. A concept for multilevel security (MLS) that models the security requirements as a multidimensional vector space with these three basic aspects as axes, is proposed in [1]. The work presented in this report has this multidimensional MLS concept as basis.

Availability is the quality or state of being ready to be used. It is a broad term that comprises many aspects. The classic understanding is that availability is associated with requirements on throughput, redundancy, backups etc. We also include restrictions and conditions resources must fulfill in order to be available.

In this paper we refine the availability dimension and propose a new security model for *resource* availability. It is computer resources that actually make information objects available. These are resources like computers, servers, programs, services and others. In addition communication resources, e.g. routers and switches, are necessary to enable access over communication systems. We address the resource aspects in the new model called “subject and object type enforcement” (SOTE). This model can specify permitted information flows between resources of different administrative domains.

SOTE is a variation of type enforcement [2]. The main difference between SOTE and type enforcement is that SOTE is a security model for information flow control instead of operating system access control.

A formal definition of availability is [3]: Let X be a set of entities and let I be a resource. Then I has the property of *availability* with respect to X if all members of X can access I.

We make a distinction between *information* and *resource* availability to reflect that “object” can refer to both an information object and to a resource. An information object can for example be a document, while a resource can be a program (or service) that retrieves the information object and enables access to it. Our definitions of these two types of availability are:

– Information availability regulates the access to information objects based on some availability criteria, e.g. access to information is determined by the information object’s type.
– Resource availability regulates the access to resources in order to get timely, reliable and secure access to services and data. E.g. interactions between resources are regulated by the resources’ types.

We have developed a data model that describes subjects, objects and other relevant security entities, using the Unified Modeling Language (UML) notation [4]. The definitions from the *Compositional Multiple Policy*

Security Model [5] have been used as basis for our model.

The model elements that are relevant for objects are depicted in Figure 1. We define an *administrative domain* to be a domain consisting of computer systems that implements the same set of security policies and security levels. This means that we have a homogeneous security environment within an administrative domain. Further, an *object* belongs to one (and only one) administrative domain. The object has a label, called *security label*. And the *object* refers to an information object of some type, a resource of some type or both.

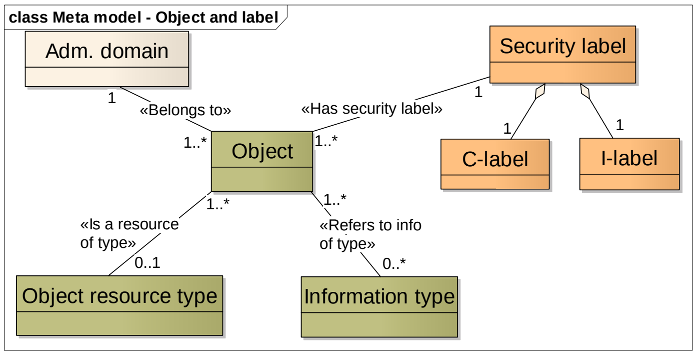

Figure 1 The object and its associations

The idea is to use the SOTE security model together with the classic multilevel security models (Bell-LaPadula and Biba). Thus, we combine confidentiality and integrity models with the SOTE resource availability model (section 4).

The Bell-LaPadula model describes a generic multilevel confidentiality policy [6] that allows information flow from low-confidentiality level to higher levels and disallows flow in the opposite direction. The other classic model, Biba, describes a generic multilevel integrity policy [7] that allows information flow from high-integrity level to lower levels. Figure 1 shows a label with both confidentiality level (C-label) and integrity level (I-label). Each of these labels consists of a policy identifier and a level.

The Bell-LaPadula model may be extended with *categories*. A category is a group of information objects that have the same type, and a user is given access to a category if the user has clearance for this category. Winjum [1] proposes linearly ordered *availability levels* as an alternative to categories. The rationale is that in many applications and scenarios, the rich class structure supported by the category structure gives unnecessary overhead. The availability level of an information object is given by an *A-label*.

The paper is organized as follows. The next section gives the rationale and assumptions for the new model. The proposed model for resource availability is presented in the following section. We then describe how multiple security models can be combined in order to express a composite security policy for information exchange. This section also gives examples. Then related work follows. The conclusions are summarized in the final section.

2. RATIONALE

The basic idea of the SOTE security model is to define the permitted information flows between resources of different types, typically between types of program components. The administrative domain is part of the model in order to express information flows between resources belonging to different administrative domains.

The setting is that computer systems of different administrative domains, inclusive your own, are connected to a common communication network. Thus, an administrative domain will not control all the resources connected to this network, as a lot of the resources belong to other administrative domains. Further, the domains implement security policies with different integrity levels and different confidentiality levels. This means that we have heterogeneous administrative domains. One domain may implement a multilevel security policy comprising all integrity levels (from low to high) and all confidentiality levels (from unclassified to secret). A second domain may implement a security policy comprising one integrity level (low) and one confidentiality level (unclassified). And a third domain may implement a policy comprising two integrity levels and one single confidentiality level (restricted). In such heterogeneous environments the domains have requirements to control and confine the interaction with resources of the other domains.

One special case is that a domain does not allow interaction with any of the computers of one of the other domains. In most cases a domain may allow interaction with some of the application programs, some of the services and/or some of the computers of the other domains. Then the domain must be able to express which interactions are permitted at a fine-grained level. The service registry, one of the basic elements of Service Oriented Architecture (SOA), is an example of a service where fine-grained control is needed. As the registry contains detailed information about the service providers of the domain it belongs to, the domain may want to put restrictions on who are allowed to access the registry. For example, the policy could be that only a subset of other domains is allowed to interact with the registry, while the rest is denied.

In addition, a domain may require that the computers and programs satisfy some conditions in order to allow interaction. For instance the domain might require that program components of a specific domain has been certified and comply with certain assurance measures, e.g. an evaluation assurance level defined in [8].

Type enforcement is well suited for restricting information flows, and these restrictions can be flexibly tailored to support the principle of least privilege. In addition, the strength of type enforcements is that it can express intransitive (indirect) information flows. The SOTE model can express such information flow policies at a fine-grained level. This is a prerequisite for flexible and secure information flow in heterogeneous environment where the domains do not implement the same set of security policies and security levels.

TRANSITIVE INFORMATION FLOWS

The classic MLS models (Bell-LaPadula and Biba) may express transitive information flows, i.e. direct information flow from the source to the recipient. An example of transitive flows is shown as a directed graph in Figure 2 (a). Nodes in the graph represent security domains and arrows (directed edges) indicate the direct information flows that are allowed. The figure shows conventional information flow between the confidentiality classification levels in the US.

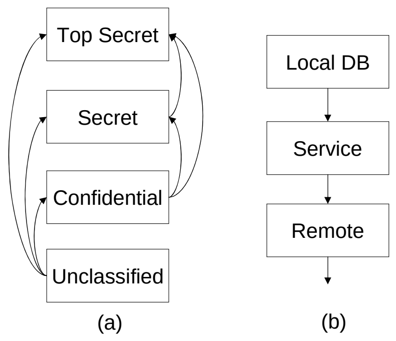

Figure 2 (a) Transitive; (b) Intransitive information flow

However, not all security issues can be addressed by transitive flow expressions. Rushby [9] states that channel-control policies seem able to specify a number of security concerns that are beyond the reach of standard security modelling techniques.

An example channel control policy is given by the directed graph in Figure 2 (b). It shows external access to information located in a local database. The flows represented by the arrows (the directed edges) are *intransitive* because, although information can flow from the database (local DB) to the remote program (called *remote*), it cannot do that directly. The information has to go through the mediation of the local service (called *service*) that provides remote access. The security policy specified by this graph is that the *only* channels for information flow from the local database to the remote recipients must be those through the serial connection of the *service* and the *remote* components.

ASSUMPTIONS

Based on the definitions and descriptions given above, we assume:

– Computers of the different administrative domains are connected to a common network.
– There is a homogeneous security environment within an administrative domain, i.e. the computer systems within a domain implement the same set of security policies and levels.
– The cooperating parties (administrative domains) implement a *common* set of confidentiality, integrity and information availability policies, e.g. a set of NATO policies. However, the implemented security levels may vary from domain to domain.
– The SOTE resource availability policy is implemented in all actual administrative domains.
– Trust between cooperating parties has been established, and the cooperating parties have knowledge of the security policies and levels of the other part.
– Confidentiality, integrity and availability are independent security properties.

3. THE SOTE PROPOSAL

Our proposal is to control information flow by defining the permitted interactions between types of *subject resources* and *object resources*. Also the permitted interactions between two types of *subject resources* can be defined.

The SOTE model defines permissions at the type level:

– A *subject resource type* element is associated with each subject instance.
– An *object resource type* element is associated with each object instance.
– An administrative domain element is also associated with each instance of object and subject.
– Permitted subject-to-object interactions are specified for pairs of *subject resource type* and *object resource type* (e.g. in a table where each row represents a *subject resource type* and each column represents an *object resource type*). The permission modes are none, read-related or write-related. Examples of read-related modes are *read*, *execute* and *getattribute*. Examples of write-related modes are *write*, *append*, *create*, *delete* and *setattribute*.
– Permitted subject-to-subject interactions are specified for pairs of *subject resource types*.
– A set of security requirements and conditions can be associated with a *subject resource type*, an *object resource* type and associations between resources. See section below.

A formal definition of the associations between the model elements are given in the UML representation of SOTE, as illustrated in Figure 3. The UML model specifies the multiplicity of the model elements attached to each end of an association. The multiplicity gives the range of the set of instances that can be active in the association. For example, in the one-to-many association «Is a resource of type» only one *subject resource type* can be assigned to a *subject* instance. And one or many, denoted by (1..*), *subject* instances can be assigned to the same *subject resource type*.

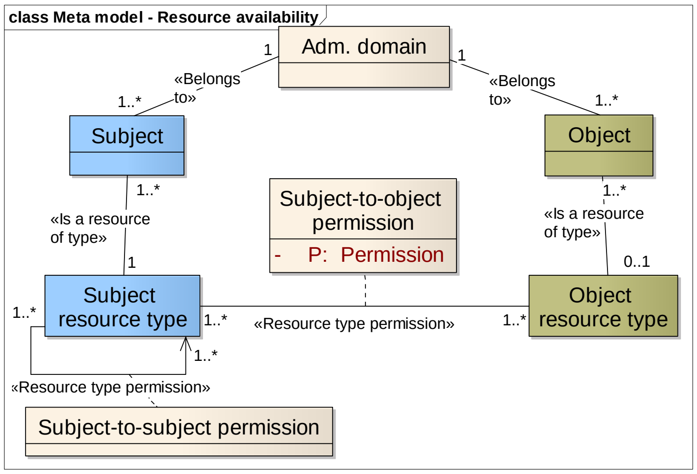

Figure 3 Resource availability – the SOTE model

The SOTE model uses UML generalizations to define a *resource type* hierarchy, both for subject and object resource types. A generalization is a relationship in which one model element (the subclass) is based on another element (the superclass). Generalized types will reduce complexity and ease the specification and administration of a SOTE policy, and should be used.

The model can also express intransitive information flow policies. A domain may for example require that remote access to a local database must go through a specific program or service. This means the domain do not allow remote programs to access the local database directly; see Figure 2 (b). Such a policy can be formulated by a combination of subject-to-object permissions (e.g. between the service and local database) and subject-to-subject permissions (e.g. between the remote program and the service).

Different approaches can be used to actually configure a SOTE policy database. One is the classic table-oriented approach. Another approach is to use a high-level policy language like the *domain-type enforcement language* (DTEL) [2]. Also the eXtensible Access Control Markup Language (XACML) [10] may be used. XACML defines a policy language for expressing policies.

CONDITIONS AND REQUIREMENTS

The SOTE model can express conditions a resource must fulfill in order to allow interaction. One condition can be that a program component has been certified and complies with certain assurance criteria, e.g. an evaluation assurance level (EAL) defined in Common Criteria [8]. Another condition can be to check the integrity of the program component, i.e. to verify the program component's authenticity with respect to origin and content. A domain may also require that the environment of the program component (such as hardware and operating systems) is verified. These examples of conditions are found as constraints in the SOTE model shown in Figure 4.

In addition the model can express security related QoS (Quality of service) requirements on object resources. One such requirement can be that an object must be able to resist Denial of Service (DoS) attacks (specified by some availability measure). This requirement, called *DoS resistance*, is expressed as a responsibility of the *object resource type* in Figure 4.

We use UML *constraints* to specify conditions and UML *responsibilities* to specify requirements on resources. A UML constraint [4] is an assertion that indicates a restriction that must be satisfied. Further, a responsibility is a contract or obligation of a model element in its relationship to other elements. We propose to use the formal language OCL - Object Constraint Language - [11] to specify the conditions (invariants and pre-conditions).

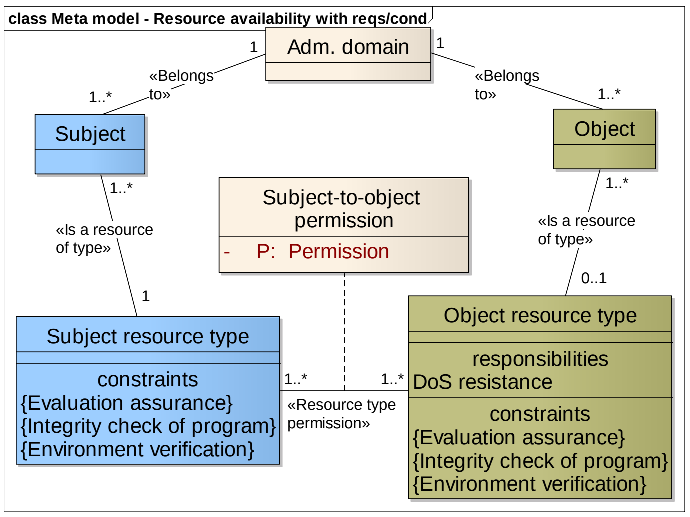

Figure 4 SOTE with security requirements and conditions

4. COMPOSITE POLICY FOR CROSS-DOMAIN INFORMATION FLOW

This section describes how multiple security models can be combined in order to express a composite security policy for information flow between administrative domains. We will combine the Bell-LaPadula confidentiality model and the Biba integrity model with the SOTE resource availability model.

CROSS-DOMAIN MULTILEVEL PERMISSIONS

First we need to supplement the UML security model with cross-domain multilevel permissions. Each domain will control the cross-domain information flow by specifying the permitted security policies and associated security levels towards other domains. These permissions can be specified by three set of security labels (a label consists of both policy identifier and level):

– I-labels: the permitted set of integrity labels
– A-labels: the permitted set of availability labels
– C-labels: the permitted set of confidentiality labels

Each set of labels for cross-domain permissions are specified as a UML constraint in the UML model. We assume that trust between cooperating parties (administrative domains) has been established. Thus, the policy permission will explicitly specify the security policies and security levels of the other domain that own domain trust.

Figure 5 gives an example of a composite information flow policy for *own* domain towards domain D1.

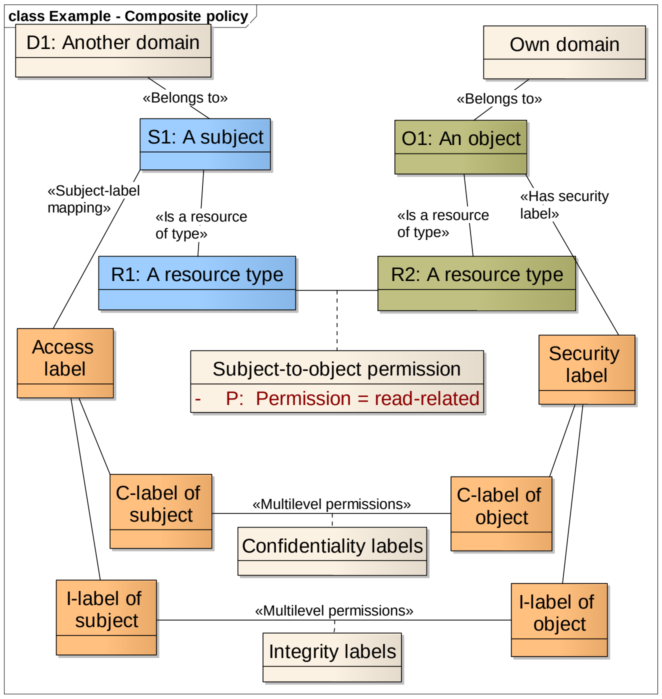

Figure 5 Composite policy for information flow

Own administrative domain, called *own* in Figure 5, is our fixed place. The subject instance S1 of domain D1 requests read access to an object instance O1 of own domain. The *integrity labels* element of Figure 5 gives the permitted cross-domain integrity policies and levels, and the *confidentiality labels* element of the same figure gives the permitted cross-domain confidentiality policies and levels. In this example we assume the Biba integrity policy and the Bell-LaPadula confidentiality policy.

Now the policy enforcement component of *own* domain can give subject S1 read access to O1 if, and only if, access is permitted in all three dimensions.

– The integrity label of subject S1 is inspected to check that it is a member of the permitted set of integrity labels (specified in the *integrity labels* element). If the integrity label is accepted, the rules of the Biba model are used to decide whether S1 can be given read access to object O1 or not.
– The confidentiality label of subject S1 is inspected to check that it is a member of the permitted set of confidentiality labels (specified in the *confidentiality labels* element). If accepted, the Bell-LaPadula rules are used to decide whether S1 can be given read access to object O1 or not.
– Resource availability: The SOTE configuration database is looked-up to check whether the S1 and O1 resources are allowed to interact or not.

EXAMPLE OF INFORMATION FLOW

An example of information flow between a tactical and a combat administrative domain is given in Figure 6. *C* is the set of confidentiality levels and *I* is the set of integrity levels, whereas *A* gives the availability policy in force.

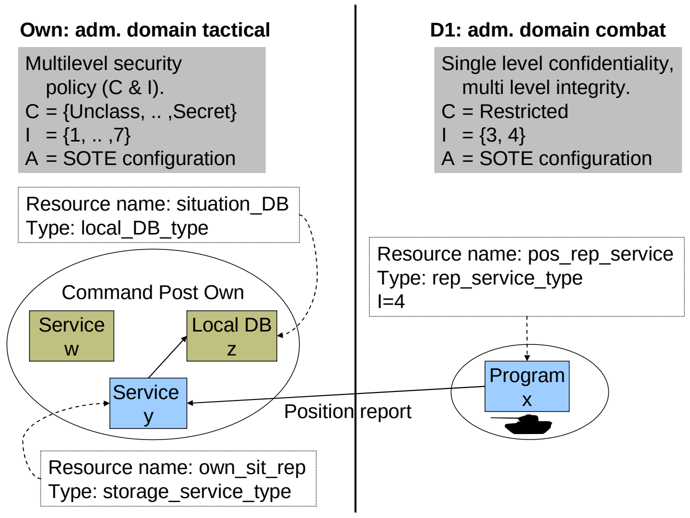
Own: adm. domain tactical
Multilevel security policy (C & I).
C = {Unclass, .. ,Secret}
I = {1, .. ,7}
A = SOTE configuration
D1: adm. domain combat
Single level confidentiality, multi level integrity.
C = Restricted
I = {3, 4}
A = SOTE configuration
Resource name: situation_DB
Type: local_DB_type
Command Post Own
Service w
Local DB z
Service y
Resource name: pos_rep_service
Type: rep_service_type
I=4
Program x
Position report
Resource name: own_sit_rep
Type: storage_service_type

Figure 6 Information flow between tactical and combat domains

The tactical domain is *own* and the combat domain is D1. *Own* domain has multilevel policies for both confidentiality and integrity, whereas domain D1 has a single level confidentiality policy and a multilevel integrity policy (two levels). Both domains implement the SOTE resource availability policy.

Our fixed place is at the own domain. We assume that own domain allows cross-domain information exchange towards domain D1 at two integrity levels, I = {3, 4}, and at the *restricted* confidentiality level.

In Figure 6 the military vehicles of domain D1 report their positions to a resource called *service y* in own domain. The integrity of these reports is handled according to the cross-domain integrity policy, which prevents that reports from the vehicles (reports are at integrity level 4) are mixed with information at higher integrity levels.

We assume that the cross-domain resource availability policy requires that position reports from D1 to the local database (z) go via service y. Direct reporting from domain D1 to the local database is not allowed in order to protect the local database. This information flow policy can be enforced as follows:

– The SOTE database is configured to allow interactions between resources of type *rep_service_type* (the resource type of program x) and resources of type *storage_service_type* (the resource type of service y).
– The SOTE database is also configured to allow interactions between resources of type *storage_service_type* and resource of type *local_DB_type* (the resource type of z).
– The policy enforcement component of *own* domain is responsible for verifying that program x belongs to domain D1 and that it is a resource of type *rep_service_type* before access to service y can be given.

Also a third resource (service w) exists in the tactical domain, see Figure 6. Service w is of type *local_domain_only* and interactions between resources of this type and *rep_service_type* are not allowed. Thus, service w will not be available to program x.

Further, we assume that the resource availability policy between the two domains mandates that the program components comply with one of the assurance levels defined in Common Criteria [8]. In our example this affects *program x*, *service y* and *local DB z*. We also assume that the cross-domain policy requires that authenticity of the information (origin and content) is protected on communication channels between program x and the local database. Thus, the authenticity of the information will be protected both when sent on communication networks and when processed by program components. In addition the cross-domain policy may require confidentiality protection of the information exchanged.

5. RELATED WORK

Type enforcement is a table-oriented form of access control originally proposed by Boebert and Kain [12]. An enhanced version of type enforcements is Domain and Type Enforcement (DTE) [2]. DTE is also included in the “Multi-Policy Views Security Model (MPVSM), which is a hybrid security model that combines DTE with confidentiality, integrity and RBAC (Role-based access control) models [5].

The type enforcement model is well suited for restricting information flows [2]. In this model, an invariant access control attribute called a *domain*[1] is associated with each subject, and another attribute called a *type* is associated with each information object. Further, a global table, the Domain Definition Table (DDT), represents allowed interactions between domains and types. Each row of the DDT represents a domain, and each column represents a type. Subject-to-subject access control is based on a second table, the Domain Interaction Table (DIT), which relates domains to domains.

Suitably configured, type enforcement partitions a system according to the principle of *least privilege*, which grants each subject only those access rights needed to perform its assigned functions [13].

An enhanced version of type enforcements is Domain and Type Enforcement (DTE). Two techniques distinguish DTE and simple type enforcement. These are [2]:

– DTE policies are expressed in a high-level language that includes file security attributes associations as well as other access control information
– During system execution, DTE file security attributes are maintained in a runtime DTE policy database thus removing the need for security-specific low-level data formats.

Sherman [14] describes how DTE has been integrated with network services in a UNIX-based research prototype. The DTE-based prototype can enforce restrictions on access to information in a networked environment. This is accomplished by assigning processes to different domains and assigning appropriate *data types* to the objects they access via network-based interprocess communication (IPC). The prototype uses DTE to control which pairs of processes can communicate via IPC and which types of information each pair can exchange.

The type enforcement security model is implemented in Security-Enhanced Linux (SELinux) [15]. In SELinux a security context (which includes type) is assigned to

[1] *Domain* in the context of type enforcement is quite different from the *administrative domain* term used elsewhere in the report.

network resources as well as other objects. This allows for specifying the network access policy for a computer. However, the network access policy cannot express generalized types of remote network processes (services). Only specific network services (ports) can be expressed.

6. SUMMARY

A new security model for *resource* availability has been proposed. The model, called “subject and object type enforcement” (SOTE), can express policies for information flow between resources of different administrative domains. It controls the *types* of resources that are allowed to interact. The ability to express the security requirements (e.g. security related quality of service requirements) and conditions a resource must fulfill, is also part of the model. SOTE is a variation of type enforcement. The main difference is that SOTE is a model for information flow control instead of operating system access control.

Type enforcement is well suited for restricting information flows. In particular type enforcements can express intransitive (indirect) information flows. The SOTE model can express such information flow policies at a fine-grained level. This is a prerequisite for flexible and secure information flow in heterogeneous environment where the domains do not implement the same set of security policies and security levels.

We have described how multiple security models can be combined in order to express a composite security policy for information flow. The classic multilevel security models (Biba and Bell-LaPadula) have been be combined with the SOTE resource availability model. Also a data model that describes SOTE and related security elements, using the Unified Modeling Language (UML) notation, has been presented.

REFERENCES

[1] Winjum E. and Mølmann B. K. (2008), "A multidimensional approach to multilevel security", *Information Management & Computer Security*, vol. 16, no. 5, pp. 436-448.

[2] Badger L., Sterne D. F., Sherman D. L., Walker K. M., and Haghighat S. A. (1995), "Practical Domain and Type Enforcement for UNIX", *IEEE Symposium on Security and Privacy*, p. 66.

[3] Bishop M. (2003), *Computer Security: Art and Science*, Addison-Wesley Professional.

[4] OMG UML (2009), "Unified Modeling Language", www.uml.org.

[5] Xia L., Huang W., and Huang H. (2007), "A Compositional Multiple Policies Operating System Security Model", *8th International Workshop, WISA 2007, LNCS 4867*, pp. 291-302.

[6] Bell D. E. and La Padula L. J. (1975), "Secure Computer Systems: Mathematical Foundations", Technical Report MTR-2547, Mitre Corporation, Bedford, MA.

[7] Biba K. J. (1977), "Integrity considerations for secure computer systems", Technical Report MTR-3153, Mitre Corporation, Bedford, MA.

[8] CC (2007), "Common Criteria for Information Technology Security Evaluation (v3.1)", CCMB-2007-09.

[9] Rushby J. (1992), "Noninterference, Transitivity, and Channel-control Security Policies", SRI International, Computer Science Laboratory.

[10] Moses T. (2005), "extensible access control markup language (xacml) version 2.0", *Oasis Standard*, vol. 200502.

[11] OMG OCL (2006), "Object Constraint Language Specification (v2.0)", (May 2006).

[12] Boebert W. E. and Kain R. Y. (1985), "A Practical Alternative to Hierarchical Integrity Policies", *Proc.of 8th National Computer Security Conference, Gaithersburg, MD*, pp. 18-27.

[13] Walker K. M., Sterne D. F., Badger M. L., Petkac M. J., Sherman D. L., and Oostendorp K. A. (1996), "Confining root programs with domain and type enforcement (DTE)", *Proceedings of the 6th conference on USENIX Security Symposium*.

[14] Sherman D. L., Sterne D. F., Badger L., Murphy S. L., Walker K. M., and Haghighat S. A. (1995), "Controlling network communication with domain and type enforcement", *Proceedings of the 18th National Information Systems Security Conference*.

[15] NSA SELinux (2009), "Security-Enhanced Linux", www.nsa.gov/research/selinux.