Protection Mechanisms

Eugene Tsyrklevich, Steve Churchill, Xiaofeng Gao and Jeff Bernstein

University of California San Diego

Abstract

Information security primarily deals with three different threats: disclosure of information (confidentiality), modification of information (integrity), and denial of access to information (availability) . This paper describes the access control mechanisms that are used to protect against those threats. The described classical discretionary and mandatory access controls primarily deal with the disclosure and modification of information. Discretionary access control lets the users control access to their data while mandatory access control puts all the access decisions under the control of a system. Role-based access control is the newer access control alternative which associates permissions and access rights with roles, and assigns users to these roles. The domain and type enforcement mechanism partitions the host operating system into access control domains and achieves confidentiality, integrity and availability by confining programs to their own domains. The described mechanisms represent the most widely studied and used access controls today. The paper lays the foundation for understanding security protection in modern operating systems as well as other areas of computer science.

1 Introduction

Information security primarily deals with three different threats: disclosure of information (*confidentiality*), modification of information (*integrity*), and denial of access to information (*availability*). This paper describes the access control mechanisms that are used to protect against those threats.

Access controls are used for limiting actions of computer system users to prevent security breaches [11]. There exists a variety of access control mechanisms and most of them are usually defined in terms of subjects and objects. A *subject* is defined as a computer system entity that initiates requests in order to complete some task, in the context of an operating system, subjects are typically the processes that operate on behalf of the users of the system. An *object* is defined as entity that an action is being performed on, in a typical operating system an object might be a file or a process.

1.1 The Access Matrix

A useful conceptual model that helps one to describe the various approaches to access controls is the *access matrix* [12]. The access matrix specifies the rights that each subject possesses for each object. The rows are labeled by subject names and column by object names. Each cell of the matrix specifies the access authorized for the subject in the row to the object in the column. The task of access control is to ensure that only those operations authorized by the access matrix actually get executed. This is achieved by means of a *reference monitor*, which is responsible for *mediating* all attempted operations by subjects on objects. Table 1 illustrates a sample access matrix.

In a large system the access matrix will be enormous in size and mostly sparse. Thus access matrix is rarely implemented as a matrix. Most popular approaches to implementing the access matrix is by means of Access Control Lists (ACL) and capability lists.

1.2 Access Control List

With *Access Control Lists* (ACL) each object is associated with a list of subjects who are allowed to access that particular object. This approach corresponds to storing the matrix by columns. Thus ACL corresponding to the access matrix in Table 1 can be represented as seen in Table 2.

One of the advantages of Access Control Lists is that it is easy to determine which modes of access subjects are currently authorized for that object just by looking at that object's ACL. Thus ACL provide convenient access review with respect to an object. It is also easy to revoke all access to an object by replacing the existing ACL with an empty one. On the other hand determining all the accesses that a subject has is difficult since it is necessary to examine the ACL of every object in the system.

1.3 Capabilities

A related type of mechanism that looks conceptually much like an ACL is known as a *capability list*. Whereas Access Control Lists associate lists of subjects with objects, capabilities associate lists of objects with subjects. Thus a capability list corresponding to the access matrix in Table 1 can be represented as shown in Table 3. Note how capabilities are represented conceptually as the reverse of ACLs.

The advantage of capabilities is that it is easy to review all accesses that a subject is authorized to perform, by simply examining the subject's capability list. However, determination of all subjects who can access a particular object requires examination of each and every subject's capability list. It is also hard to revoke access to an object since system must find all the outstanding capabilities for an object.

A number of capability-based computer systems were developed [14][18], but did not prove to be commercially successful. Modern operating systems typically take the ACL-based approach.

1.4 Authorization Relations

We have seen that both ACL and capability based approaches have advantages and disadvantages with respect to access review. There are representations of the access matrix which do not favor one aspect of access review over the other. For example, the access matrix can be represented by an authorization relation (or table) where each row of the table specifies one access right of a subject to an object as seen in Table 4. If this table is sorted by subject, we get the effect of capability lists. If it is sorted by object we get the effect of ACLs. Relational database management systems typically use such representation.

We are now going to present the access control mechanisms that are used for protecting against information security threats. We start with the discretionary access control which is one of the most flexible and popular mechanisms in use today. Section 3 is going to describe the mandatory access control, as well as different models used for insuring the confidentiality and integrity of the information. In Section 4, we are going to talk about role-based access control and the advantages that schema has over the classical approaches. The last presented access control mechanism is going to be domain and type enforcement described in Section 5.

2 Discretionary Access Control

Discretionary access control (DAC) introduced by Lampson [8] is based on the idea that individual users are "owners" of objects and therefore have complete discretion over who should be authorized to access the object and in which mode (e.g., read or write).

2.1 UNIX File Permissions

One DAC implementation that readers might be familiar with is file permissions in UNIX operating system. UNIX systems control access to files by dividing the world of users into three categories, and telling the system what each group can do with a file [10]. In UNIX the user categories are

Self - you, the owner of a file
Group - a set of users, for example members of a particular group like faculty

Subject \ Objects	File1	File2	Process1	Process2
Alice	Read Write	Read	Wakeup Kill	
Bob	Read Execute	Write	Wakeup	
Eve	Execute			Kill

Table 1: Access Matrix

Object			
File1	Alice (Read Write)	Bob (Read Execute)	Eve (Execute)
File2	Alice (Read)	Bob (Write)	
Process1	Alice (Wakeup Kill)	Bob (Wakeup)	
Process2	Eve (Kill)		

Table 2: Access Control List

Subject			
Alice	File1 (Read Write)	File2 (Read)	Process1 (Wakeup Kill)
Bob	File1 (Read Execute)	File2 (Write)	Process1 (Wakeup)
Eve	File1 (Execute)	Process2 (Kill)	

Table 3: Capability List

Other - everyone else, users other than yourself and the members of your group

Each file has a set of protection bits which specify the type of access each category of users is granted. Each category is assigned a 3-bit string that is used to define read, write and execute access permissions. For example, suppose that a file A has the following permission bit string:

(r w x) (r w x) (r w x)

This tells us that the owner of the file A, all members of the group to which file A belongs, and all the other users have read, write and execute access to file A. Systems often denote that certain permissions are not present by marking a '-' in the associated position of permission bit string. Thus the following permission string

(r w -) (r- -) (– -)

tells us that the owner of a file can read the file

Subject	Object	Protection
Alice	File1	Read
Alice	File1	Write
Alice	File2	Read
Alice	Process1	Wakeup
Alice	Process1	Kill
Bob	File1	Read
Bob	File1	Execute
Bob	File2	Write
Bob	Process1	Wakeup
Eve	File1	Execute
Eve	Process2	Kill

Table 4: Authorization Relations

and write to it, file group members can read the file, while everyone else is denied access to the file. Notice that the UNIX file protection scheme is just a special case of ACL compressed to 9 bits.

It is also important to notice that UNIX supports the rights amplification mechanism that is used for granting temporary permissions to users when they invoke special programs that are defined to be *setuid* (set-user-identification). A setuid program executes with the privileges of the owner of a file and can be used for accomplishing tasks that a user otherwise wouldn't have the proper authorization for. Programs that are setuid-to-root on UNIX systems are the ones that can cause the most potential damage. A whole class of UNIX-based attacks has arisen to find and compromise those setuid programs. The pervasive use of root privilege is a central problem for UNIX security because an attacker who subverts a single root program gains complete control over the system. The Domain and Type Enforcement mechanism, described in section 5, reduces the risk of setuid-root programs by confining them to a single domain.

2.2 DAC Overview

DAC mechanisms tend to be very flexible and are widely used in both commercial and military sectors. Even though DAC schemes are very popular nowadays, they have inherent weaknesses that information can be copied from one object to another, so access to a copy is possible even if the owner of the original does not provide access to the original . Furthermore, since DAC parameters are easily changed, users protected by DAC mechanisms may be susceptible to *Trojan horse*[1] attacks since the protection state can be modified without explicit user instructions.

[1] A Trojan horse program is defined as any program that is expected to perform some desirable function, but that actually performs some unexpected and undesirable function. See [19] for an amusing example.

3 Mandatory Access Control

As we have seen, Discretionary Access Control mechanisms tend to suffer from problems such as the dissemination of information. *Mandatory access control* (MAC) introduced by Bell and LaPadula [2] was designed to solve the information flow problem. MAC tends to be very strict and is mostly used to govern access to very sensitive data, e.g. military classified information or secure corporate data. Unlike discretionary access control systems, which let users to have complete discretion over who should be authorized to access their information, mandatory access control puts all such access decisions under the control of the system.

3.1 Information Confidentiality

3.1.1 Security Labels

The system controls the access on the basis of classification of subjects (users, programs) and objects (data, devices, sockets etc) in the system. Each subject is assigned a sensitivity label to specify the level of trust which is also often called a *security clearance*. Every object has a security label, called *security classification*, that reflects the sensitivity of the information contained in the objects. In classical MAC, the security level is an element of a hierarchical ordered set. For example, the ordered set for the military area can consist of TS, S, C and U as shown in figure 2 (a).

For the subjects, each level is more trusted that the level below it and for the objects, each level is more important than the level below it. We can use $>$ to describe the order between these levels. Each security level is said to *dominate* the levels below it. In our case, we can write TS $>$ S $>$ C $>$ U.

3.1.2 Confidentiality Model Rules

MAC will grant the access of an object to a subject only if certain relationship is satisfied between the security levels associated with the two. In particular the following two principles are required to hold:

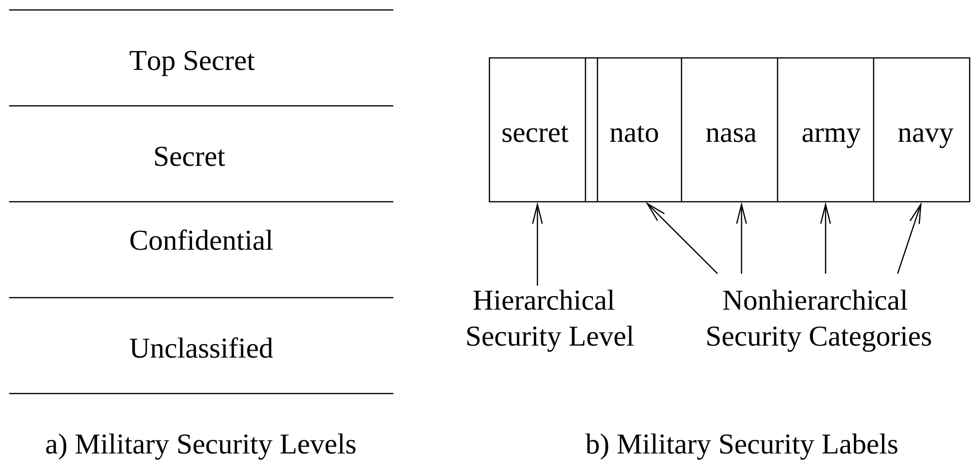

Figure 1: Mandatory Access Control Security Labels

Read Down (simple-security property) A subject's clearance must dominate the security level of the object being read.

Write Up (*-property). A subject's clearance must be dominated by the security level of the object being written.

Satisfaction of these principles prevents information in high level objects from flowing to objects at low levels. In such a system, information can only flow upwards or within the same level. This satisfies the disclosure of information security principle.

Figure 2 illustrates the information flow between various security levels by means of a level diagram. The level diagram consists of several horizontal lines and a collection of circles and squares. The lines are viewed as boundaries between the various security levels. Circles are drawn to represent subjects, while squares represent objects. Operations are depicted by drawing directed arcs from subjects to objects. The diagram in Figure 2a demonstrates various allowed and disallowed read and write operations in a Bell-LaPadula model. The diagram depicts a read down from a subject at the highest security level to an object at the lowest security level. It also depicts a write up from a subject at the middle security level to an object at the highest security level, as well as write from a subject at the lowest security level to an object also at the lowest security level. Notice how a write from a higher level to a lower level is prohibited just as is a read operation from a lower level to a higher level.

There are certain problems with the write-down rule in the above example. Even though a lower level subject cannot read a file, it can still write to it (the so called *blind writes*). This is obviously not desired and thus most systems don't allow the write-up operation.

3.1.3 Security Categories

In the system using the MAC policy as described above, a user can access all the objects in the same security level as his clearance or in the levels below. This is not desirable in a large system, especially in military fields. We want the system can only grant access to the objects in the user's fields. This can be implemented by dividing the objects into *categories* as shown in figure 2b. Now the classification label associated with each subject and object consists of a pair composed of a security level and a set of categories. The set of categories associated with each user reflects the specific areas in which the user operates. With categories associated with security label, MAC can provide a finer grained security classification. This can enforce restriction on the base of a

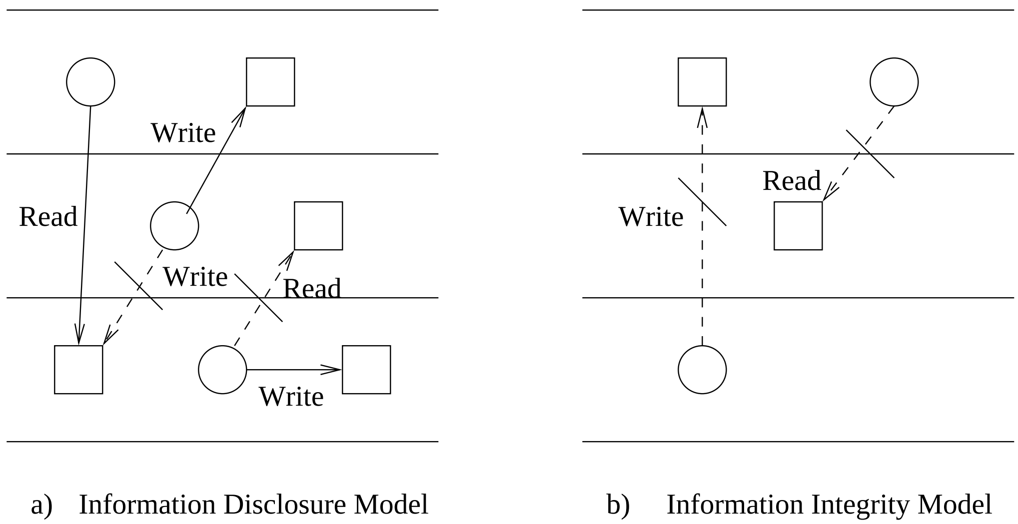

Figure 2: Mandatory Access Control Read and Write Operations

least privilege principle, i.e. a subject should be only given those accesses which are necessary to the subject to fulfill his responsibility. The example of this is illustrated in figure 2b.

3.2 Information Integrity

MAC can also be used to enforce *information integrity.* Integrity models such as those described in [3] and [5] are primarily concerned with unauthorized modifications of information.

3.2.1 Integrity Model Rules

For example, the integrity levels can be Crucial (C), important (I) and Unknown(U). The integrity level associated with each object reflects the degree of trust that can be placed in the information stored in the object, and the potential damage that could result from unauthorized modification. The integrity level associated with a user reflects the user's trustworthiness for modifying the data at that level. The control policies are just the counterpart to those used for disclosure integrity:

Read Up (simple-integrity property) A subject's integrity level must be dominated by the integrity level of the object being read.

Write Down (integrity *-property) A subject's integrity level must dominate the integrity level of the object being written. Here the information can only flow from to top to bottom.

Similarly to Figure 2a, Figure 2b demonstrates various allowed and disallowed read and write operations in a system enforcing information integrity. Writes from lower levels are not allowed, just as reads from higher levels to lower levels are not allowed. At first, this might seem counter-intuitive but since the information integrity model is only concerned with the integrity of data the rules do enforce the correct behavior. It is also possible to combine the disclosure and integrity models to get the best of both worlds. Example of that is shown in [1].

3.3 MAC Overview

As we have seen, Discretionary Access Control mechanisms tend to suffer from problems such as Trojan horses and dissemination of information. Mandatory access control was designed to solve the information flow problem. Unfortunately, mandatory controls don't solve the Trojan Horse problem completely. Information can still leak from higher levels down to lower levels by means of covert channels. A *covert channel* [7] exists whenever some computer system mechanism is used in an unexpected manner to provide a means by which information can flow to an unauthorized individual. For example by manipulating a computer system load a high level user can slowly leak information to a lower level user who closely monitors the system load and interprets load peaks as bit '1' and other loads as a bit '0'. The covert channel problem remains a major bottleneck for high assurance MAC. Mandatory access control also does not solve the *inference problem* where high information is deduced by assembling and intelligently combining low information. More information on both problems can be found in [1].

4 Role Based access controls

Role based access control (RBAC) is a flexible mechanism of managing user rights within a system. The management of permissions is greatly simplified by the association of permissions and access rights with roles, and assigning users to these roles. A major advantage of using an RBAC based system is the ease of administration of a users rights. The assignment of roles to a user may be configured and reconfigured easily, and the rights of a user a may be easily determined by looking at the set of roles a user belongs to.

The many implementation decisions required in designing an RBAC system allow for many different RBAC models to be constructed. A basic description of several concepts is given here before detailing the requirements of any RBAC models.

4.1 RBAC Models

Roles are logical groupings with which permissions are associated. Data abstraction is provided with RBAC by associating object and operation pairs with the permissions. The use of the permission pair transaction, rather than access rights such as read, write, execute allows the functionality of a task to be divided from the data operations. The permissions assigned to a role are determined by the site security administrator, to form independent roles with different functional requirements.

Users of an RBAC based system are assigned a set of roles, which they may assume to utilize the system functionality. Roles are often allowed to contain the functionality of other roles, creating a hierarchical set of permissions, which allows administrators to categorize groups of roles which may represent a set of user responsibilities. RBAC systems are able support the concept of a *least privilege* in that roles assigned to a user by a security administrator will reflect only those of their job functions, without granting access to other functionality.

There are various ways of implementing such an access control system, and role base access control has been loosely defined. Many models for RBAC systems have been proposed and implemented. References in this paper will use the National Institute of Standards and Technology (NIST) RBAC models [13], which breaks the classification of different types of RBAC systems into four separate models: Flat, Hierarchical, Constrained, and Symmetric. Each model adds a level of increased functionality, and complexity.

4.1.1 Flat model

The *flat model,* as defined by NIST, requires that user-role and permission-role assignment can be many to many. Also required is the ability to administratively view the roles assigned to a user, or the users associated with a particular role. The flat RBAC model also allows users to assume more than one role at a time, which essentially makes flat RBAC a group-based access control mechanism.

4.1.2 Hierarchical model

The *hierarchical model* allows for the role-role relationships to exist in a hierarchical form. Two types of hierarchies are distinguished: general, and restricted. General hierarchies support an arbitrary ordering of role-role relationships to form the hierarchy. Restricted hierarchies impose such structures as trees on the role-role relations. The general hierarchy allows for greater flexibility in the design of the access control, although the model leaves the specification of hierarchy type open. Different types of role activation may occur within a hierarchy as well. The activation of a role in an inheritance hierarchy implies the activation of all associated roles, whereas the activation of a role in an activation hierarchy does not. The model is open to support either hierarchy sub-type or both.

4.1.3 Constrained model

Constrained RBAC allows for the enforcement of the security concept of separation of duties. *Separation of duties* is a means of distributing the responsibility of a task across multiple users, perhaps enforcing organizational policies concerning actions individuals may perform. The ability to do this decreases the risk of malicious activities, by increasing the number of individuals which must be involved. This is achieved by dividing access control into multiple mutually exclusive roles, which may be done statically with role assignment, or dynamically when based on role activation.

4.1.4 Symmetric model

Symmetric RBAC adds the requirement of being able to administratively view the roles to which a particular permission is assigned, and to view the permissions assigned to a role. This difference was separated as a different RBAC model because of the difficulty of its implementation on distributed systems [13].

A full symmetric implementation of RBAC allows for administrators to easily determine the access rights of a user by looking at the set of roles a user may possess. Although the assignment of permissions to roles may be a difficult task in an RBAC system, most of this configuration occurs at the time of the system setup. Once this has occurred, the delegation of these roles is an easy administrative task. Modifying a users access level becomes an easier task when their rights may be constructed by assigning them a set of roles. Administration of the role-permission, role-role and user-role relationships provides an easy means of viewing the distribution of privileges in a system.

4.2 RBAC Overview

One major advantage of using an RBAC based system is that RBAC is largely policy independent. RBAC allows for the assignment of permissions to objects to roles, much like in a discretionary access control system. The explicit association of roles with users allows for the restriction of rights much as in a mandatory access control system. While the assignment of roles and permissions may be a difficult task, the possible ways of configuring a system are open. The concept of a role is analogous to that of a job or function, with a set of responsibilities and privileges associated with the role. Organizations may often choose to model their set of roles after the functional units within their own organization. It has been shown that the flexibility in the configuration of RBAC systems yields the ability to form either discretionary or mandatory access control policies [9].

5 Domain and type enforcement

The basic concept of *type enforcement* [4] is that all the objects are partitioned into *types* based on their integrity properties and all the subjects are partitioned into *domains* based on their roles in the system. Subjects in a particular domain are indented to play a particular *role* and permitted to execute code that is appropriate for that role and access the data that should be restricted to the minimal set of accesses required for that role. Subjects are not allowed to change domains unless the system lets them.

5.1 Inside of DTE

A *Domain Definition Table* (DDT) is created to represent what operations subjects in a specific domain can perform on objects of a specified type. Each entry indicates the access granted to a subject operating in the corresponding domain to objects of the

Domain \ Type	system	inetd	ftpd1	ftpd2
system	rwx	—	—	—
inetd	—	rwx	—	—
ftpd	—	—	rwx	rwx

Table 5: A Domain Definition Table (DDT)

Domain \ Domain	system	inetd	ftpd
system	True	-	-
inetd	True	True	-
ftpd	-	True	True

Table 6: A Domain Transition Table (DTT)

corresponding type. Table 5 demonstrates a sample system configuration with 3 domains and 4 types, 2 of the types belong to the same domain (ftpd). Notice how each domain has access only to the data of an appropriate type. That is the ftp process doesn't need and doesn't have access to system files.

An arbitrary static access matrix (see section 1.1), where there is one row for each subject and one column for each object is equivalent to DDT. A general access matrix is converted to DDT by grouping objects and domains by the corresponding types and domains. Grouping produces a smaller matrix, which is desirable from an efficiency point of view, and provides a better abstraction of the security constraints.

Type enforcement extends the general access matrix by creating a *Domain Transition Table* (DTT). The DTT indicates the domains to which a subject in a given domain can pass execution. Subjects in the domain corresponding to a given row can transfer control to subjects in domains for which the corresponding entry is True. DTT is shown in Table 6 which continues our example with 3 domains. Notice that the system domain is allowed to "transition" into the inetd domain, that is the system process is allowed to start new inetd processes, and inetd is allowed to spawn new ftpd processes but not the other way around.

5.2 DTE Example

Let's see now how the DTE mechanism can protect our sample system. Our system consists of three processes which have assigned roles to them (see Table 6). Among system process responsibilities is starting a new inetd process. An inetd process is responsible for accepting incoming network connections and passing them on to other processes for further processing. An ftp process is responsible for accepting new connections from an inetd and processing them. Now let's imagine that an attacker have been able to subvert our ftp server into executing malicious code. On a normal system without type enforcement this usually means that an attacker has full control over the system since daemon processes such as ftpd usually run with root privileges thus malicious code is going to be executed as root. At that point an attacker can access and modify any information stored on our system. With Domain and Type Enforcement mechanism, not everything is lost even if an attacker have been able to trick an ftp server into running arbitrary code. Since the ftpd process is executing in its own compartment (domain), the DTE mechanism is going to deny all the accesses that are outside of the ftpd domain successfully constraining an attacker to the ftpd domain. If the DTE domains and types are carefully chosen the effects of attacks are greatly reduced by confining the attack to a single security domain [16][17].

5.3 DTE Overview

If the DDT and DTT are statically defined, type enforcement provides more protection than methods where the access is left to the discretion of the user and can be used to enforce various mandatory policies.

6 Conclusions

There exist a variety of different access control mechanisms but all of them are trying to achieve the same goal, and that is they are designed to prevent security breaches. A security breach might include disclosure of sensitive information or modification of data.

The most widely used access control mechanism today is Discretionary Access Control. DAC is based on the idea that users are the owners of data and therefore have complete discretion over who can access their data. DAC is very flexible but has some inherent weaknesses such that it doesn't provide assurance on information flow and because of that it is vulnerable to Trojan horse attacks.

Mandatory Access Control solves the information flow problem by labeling all the data and letting the system decide who can access what. MAC can be used for protecting data from various threats such as disclosure or modification of data. Unfortunately, MAC schemes don't protect data from leaking by means of covert channels. MAC also does not solve the inference problem where high information is deduced by assembling and intelligently combining low information.

Role based access control is a form of mandatory access control which bases access control decisions on the functions a user is allowed to perform within an organization. Some of the advantages of RBAC schemes include hierarchical roles, least privelege and separation of duties. All of those characteristics enchance the security of a system. It has been also shown that the flexibility in the configuration of RBAC systems yields the ability to form either discretionary or mandatory access control policies.

Domain and Type Enforcement mechanism is based on grouping of related objects into types and related subjects into domains. DTE provides greater security by running each task in its own compartment that cannot affect tasks in other domains.

All of the described access control mechanisms have their uses and their applications. While DAC schemes might work fine for user workstations, military might require more protection such as offered by Mandatory Access Control. The use of a particular protection scheme depends on the expected system security, performance as well as usability and cost.

References

[1] E. Amoroso, Fundamentals of Computer Security Technology. Prentice Hall, 1994.

[2] D.E.Bell, L.J.LaPadula, Secure Computer System: Unified Exposition and Multics Interpretation. Technical Report, MTIS AD-A023558, MITRE Corporation, 1975.

[3] K.J. Biba. Integrity Considerations for Secure Computer Systems, USAF Electronic Systems Division, Bedford, MA, 1977.

[4] W. E. Boebert and R. Y. Kain, A Practical Alternative to Hierarchical Integrity Policies, In Proceedings of the Eighth National Computer Security Conference, 1985.

[5] D. C. Clark and D. R. Wilson. A comparison of Commercial and Military Computer Security Policies, IEEE Symposium on Security and Privacy, 1987.

[6] D. Ferraiolo and R. Kuhn. Role-based access controls. In 15th NIST-NCSC National Computer Security Conference, Baltimore, MD, October 13-16 1992.

[7] B.W. Lampson. A note on the confinement problem. Communications of the ACM, 1973.

[8] B.W. Lampson. Protection, Fifth Annual Princeton Conference on Information Sciences and Systems, March 1971.

[9] S. Osborn, R. Sandhu, and Q. Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. ACM Transactions on Information and System Security, May 2000.

[10] D. Russell, G.T.Gangemi, Sr. Computer Security Basics, 1st Edition O'Reilly, 1991.

[11] R. Sandhu, Access Control: the Neglected Frontier. Proc. First Australian Conference on Information Security and Privacy, Springer, 1996.

[12] R. Sandhu, and Pierangela Samarati. Access control: Principles and practice. IEEE Communications, September 1994.

[13] R. Sandhu, Ferraiolo, D., and Kuhn, R. 2000. The NIST model for role-based access control: Towards a unified standard. 5th ACM Workshop on Role-Based Access Control, July 2000.

[14] J. S. Shapiro, J.M. Smith, D.J. Farber, EROS: a Fast Capability System, 17th ACM Symposium on Operating Systems Principles , Kiawah Island Resort, December 1999.

[15] K. Thompson, Reflections on Trusting Trust, Communication of the ACM, Vol. 27, N o. 8, August 1984, pp. 761-763.

[16] K. Walker, D. Sterne, L. Badger, et al. Confining Root Programs with Domain and Type Enforcement (DTE). Sixth USENIX UNIX Security Symposium, San Jose, 1996.

[17] Why mandatory controls are necessary in an internet environment. http://research-cistw.saic.com/cace/dte.html

[18] W. Wulf, E. Cohen, W. Corwin, A. Jones, R. Levin, C. Pierson, and F. Pollack. HYDRA: The kernel of a multiprocessing operating system. Communications of the ACM, 17(6):337–345, July 1974.