17

Going Beyond MAC and DAC Using Mobile Policies

Amgad Fayad, Sushil Jajodia, Don Faatz, Vinti Doshi
The MITRE Corporation

Key words: Access control, Attribute Certificates, DBMS Security, Middleware Security, Mobile Policy, PKI

Abstract:

Many access control requirements cannot be automated using traditional mandatory access control (MAC) and discretionary access control (DAC) security mechanisms. Examples include user-attribute-based access control and owner-retained access control for handling specially marked data. While several researchers have identified the need for access controls that provide more flexibility than MAC and DAC, the proposed mechanisms for implementing these controls have several shortcomings. In this paper, we describe an access control mechanism that combines attribute certificates with mobile policy to overcome these shortcomings. Attribute certificates permit fine-grained authorisations based on user attributes, such as group membership, rank, and role. Mobile policies allow application-specific policies to move along with the object to other elements of the system. Mobile policies are expressed using an extension to a high-level definition language that we previously proposed in Reference [5].

1. INTRODUCTION

Traditionally, mandatory access control (MAC) and discretionary access control (DAC) security mechanisms have been used for providing information protection. MAC groups users and data based on classification levels. Users can only access data that is classified at their level or lower. Sharing of classified data with users who do not have the required classification clearance is strictly prohibited. In comparison, DAC restricts access to information based on the user's identity and authorisations stating the accesses each user can execute on the objects of the system.

Many access control requirements within the United States (U.S.) Department of Defense (DoD) and the Intelligence Community cannot be automated using traditional MAC and DAC mechanisms. In particular, these communities need to enforce, within an automated system, the true intent of access control requirements associated with special markings, including dissemination controls and release markings, caveats, and warnings [1-3,7,13]. Particular examples include *user attribute-based access control* and *owner-retained access control* for handling specially marked data [13].

Release markings such as Not Releasable to Foreign Nationals (NOFORN) and Top Secret Releasable to Canada (TS REL CANADA) are examples of user attribute-based access control. An object with the NOFORN marking can only be released to an U.S. citizen. Similarly, an object marked with TS REL CANADA can only be given to an individual who has a TS clearance from either the U.S. or Canada.

Originator Controlled (ORCON) release represents an example of owner-retained access control. Any object marked ORCON may be released to users belonging to a specified list of organisations or users; any release to others not on the list requires permission of the originator of the object.

While requirements such as REL XX and ORCON can be implemented using MAC, the solution is cumbersome and, therefore, not an acceptable general-purpose solution. The solutions proposed by others [1-3,13] also have several shortcomings. We will elaborate on this topic in Sections 2 and 3.

Some of the difficulties in carrying out the intent of the aforementioned access control requirements are that a number of these markings are based on user attributes, provisions are made for exceptions, information cannot sometimes be released with the consent of the originator, and the composition of labels when data with different markings are combined is not straightforward.

In this paper, we describe an alternative solution that combines attribute certificates (ACs) with mobile policy. Public key certificates, also known as identity certificates (ICs), allow the identities of the users to be mobile. A user can prove his/her identity, for access control purposes, using his/her IC. The advent of ACs extends ICs beyond identity-based authorisations. That is, attribute certificates provide the capability of assigning *attributes* to users, and, therefore, they are ideal for providing user attribute-based dissemination control.

Mobile policies allow access control rules to move with the objects to which the policies apply. Enforcement can take place within any trustworthy component of the system. We extend the high-level definition language called *Mobile Policy Language* (MPL) (pronounced "maple") proposed by us [5] to specify policies such as NOFORN and ORCON. MPL extends the traditional Structured Query Language (SQL) access control commands to incorporate attribute certificate information as well as *provisions* [10] that specify required actions to be taken during policy enforcement.

The remainder of this paper is organised as follows. Section 2 describes some motivating examples, Section 3 describes solutions from previous work, Section 4 describes ACs and MPL, Section 5 describes the mobile policy framework, Section 6 shows how to apply MPL and attribute certificates to address the examples from Section 2, and, finally, Section 7 provides conclusions and describes future work.

2. MOTIVATING EXAMPLES

In this section, we take several examples from [13] to illustrate the need for flexible access control policies that go beyond traditional MAC and DAC security mechanisms.

Example 1 Release Markings: NOFORN is a marking used by the DoD/Intelligence community to indicate that access to data requires U.S. citizenship. It is another access control in addition to the restrictions imposed by MAC. Exceptions are often made to the NOFORN control, generally with the use of the REL markings. For example, if a document is marked "NOFORN/REL CAN", only U.S. and Canadian citizens can access that document. One way to handle NOFORN data within a system's MAC controls is to define a compartment in the system for NOFORN data. This compartment would be added to each user's clearance credentials to give the user access to NOFORN data. However, this action may lead to undesirable results, as individuals would then become privy to all data marked with NOFORN control rather than just selected data objects. Also, every time a special-case release needs to be accommodated, a cumbersome process involves adding new compartments and going through the process of changing a user's clearance.

Example 2 Originator Controlled: Another type of access control used in the IC is ORCON. When ORCON is specified, only users or groups specified by the originator of the data are allowed access. If any additional users or organisations require access to the data, prior consent of the originator is required. For example, suppose a user x from department X releases an object O, marked with ORCON, to users in department Y. Any copy of O made by any user y in department Y would be subject to the same restrictions as object O. Object O (or its copy) is not releasable by users in department Y to users in other departments without the permission of user x, the originator.

Example 3 Label Composition: When two data objects with NOFORN-REL or ORCON markings are combined, how does an automated system combine these markings? For example, suppose object O_1's label states that all government employees and non-government employees cited in access control list A (ACL-A) are granted access to O_1. Also suppose that O_2's label states that all U.S. citizens and foreign nationals listed in ACL-B are granted access to O_2. We will call the object O_3, which is the result of combining O_1 and O_2. The composite policy for O_3 should state that for a user to have access, the user would need to satisfy one of the following requirements:

- A U.S. government employee with U.S. citizenship
- A U.S. government employee who is not a U.S. citizen and has been given access on ACL-B

- A U.S. citizen non-government employee who has been given access on ACL-A
- A foreign national non-government employee who has been given access on ACL-A and ACL-B

Traditional access control mechanisms based on MAC and DAC are not adequate to automate the policies in these examples. We discuss other solutions and their shortcomings in the next section.

3. PREVIOUS SOLUTIONS

In this section, we describe the solutions provided by McCollum et al. [13] and Abrams et al. [1-3]. We also identify the shortcomings of their solutions.

Since NOFORN and ORCON markings rely on the user's *attributes,* not just clearance level, and ORCON requires specifying an ACL, McCollum et al. [13] introduced user attribute-based access control and Owner-Retained Access Control (ORAC).

User attribute-based access control calls for associating attributes to users. The following are examples of user attributes:

- NOFORN is an attribute that describes U.S. citizens
- CONTRACT is an attribute that describes a government employee
- NOCONTRACT is an attribute that describes a non-government employee

To provide attribute-based access control, McCollum et al. [13] make two assumptions: First, attributes must be assigned in data object labels. Second, access control rules must specify those attributes required for access. While this is a good paradigm, it stops short of discussing how attribute-based access control would be implemented in a distributed environment. Also, it does not provide a mechanism for handling caveats and exceptions to general rules. Finally, it provides no solution to handle exporting data objects from one system to another.

The solution in [13] defines ORAC to enforce ORCON. ORAC calls for using ACLs to allow or explicitly deny access to data objects. The user who

creates the data object is considered its owner and has the right to define an ACL on the object. In contrast to DAC ACL, an ORAC ACL, with its associated owner, propagates along with the data. For example, suppose user x creates object1 with ACL-x. Suppose further that user y copies object1 to produce object2 with ACL-y. Object2 will retain ACL-x in addition to ACL-y.

When enforcing ORCON through ORAC, every time a subject wants to give data access to a new subject who is not on the original ACL, the owner of the data must explicitly approve the access. While this procedure is required in some cases, in other instances the owner may wish to grant another user the right to grant access to the object. Reference [13] suggests a solution using a new parameter, *owner privilege,* which like other privileges can be granted to a subject, and by doing so the owner relinquishes his/her ownership of the data object to others. In this paper, we have another option, called *grant option,* which allows an owner to give grant authority to others for some, but not all, of the privileges on the objects.

Abrams et al. [1-3] built a prototype that had the ability to enforce an open set of access control policies. For each policy, the prototype required that a separate policy-specific module be added to the trusted computing base (TCB) that can enforce the policy. This means that before an object is exported to another component, one must ensure that the appropriate module is available at that component.

4. ATTRIBUTE CERTIFICATES AND MOBILE POLICY LANGUAGE

In this section, we begin with an overview of attribute certificates, followed by a description of MPL. We include several examples to show how our framework can incorporate ORCON and REL XX markings.

4.1 Attribute Certificates

In this paper, we assume that user attributes can be made available through ACs [4, 5]. ACs are digital documents that contain a list of

attributes, each of which is an ordered pair (Tag, Value), where Tag is an identifier and Value is a text string.

Examples of attributes are the following:

- (group, NATO)
- (role, Treaty Negotiator)
- (rank, Major)
- (citizenship, Canada)

An AC, as described in [4], is a structure represented in Abstract Syntax Notation 1 (ASN.1) just like an IC. However, an AC does not contain a public key. An AC must be cryptographically linked to an IC and can be used only in conjunction with this corresponding IC. This restriction is necessary, since an AC cannot provide any information about a user's identity.

ACs provide several benefits. First, ACs can be managed, and distributed, using deployed Public Key Infrastructure (PKI) systems. Second, ACs allow user attributes to be mobile within the distributed system.

Directories provide an alternative to ACs for storing and disseminating user attributes in a distributed computing environment. Like ACs, directories can be used in conjunction with ICs. The following list describes a likely scenario for using directories:

- User x authenticates to server S using his/her IC.
- Server S performs a directory lookup using Lightweight Directory Access Protocol (LDAP) to locate user x's attributes.
- Server S performs attribute-based access control to determine access privileges.

4.2 Mobile Policy Language

MPL has three types of statements: **Grant, DoNotGrant,** and **MustGrant. Grant** gives access to objects to other users. **DoNotGrant** provides the ability to deny stated accesses to others. Finally, **MustGrant** is a *strong* authorisation; it is used to ensure that stated authorisations would be obeyed by the system and to resolve conflicting **Grant** and **DoNotGrant** statements on a particular access [10-12].

Definition 1 (Authorisation Rule): An authorisation rule is a rule of the following form:

Grant <*Access Type*> **on** <*Object*>
to <*Security Principal* >
[with Grant Authority]
[with provision <*Provisions*>**]**
where [<*Security Principal*> **has attribute** <(tag, value), …> | <predicate>]

Authorisation rules allow grantors to give accesses to other security principals. An access may be granted **with Grant Authority,** which permits the grantee to further grant acquired rights to other users. *Provisions* [10] specify required actions to be taken during the policy enforcement. Stated access to an object is allowed after the conditions in the **where** clause have been validated and the provisions have been applied. The term <predicate> in the where clause is any condition that must be satisfied during the evaluation of the access request.

Example 4: Consider the following authorisation rule:

Grant print **on** Balance_Sheet
to user
with Grant Authority
with provision Add notice "For Accounting Group Only"
where user **has attribute** (group, accounting group) **and**
(rank, manager of accounting group)

This rule states that a user can print the file Balance_Sheet if he/she is a member of the accounting group and has the rank of an accounting group manager. The notice "For Accounting Group Only" will be added to the printed copy. Since the print permission has been given with grant authority, a grantee can grant the print right to other users by issuing the following command:

Grant print **on** Balance_Sheet
to Sue
with provision Add notice "For Accounting Group Only"
where Sue **has attribute** (group, accounting group)

This statement says that Sue can print the Balance_Sheet as long as she is a member of the accounting group. Sue cannot grant the print privilege to others, however.

Definition 2 (Negative Authorisation Rule): A negative authorisation rule is a rule of the following form:

DoNotGrant <*Access Type*> **on** <*Object*>
to <*Subject*>
[with provision <*Provisions*>**]**
where [<*Security Principal*> **has attribute <**(tag, value), …>
| <predicate>]

Negative authorisation rules explicitly deny access to an object by certain security principals. Some applications require explicit negative authorisations.

Definition 3 (Strong Authorisation Rule): A strong authorisation rule is a rule of the following form:

MustGrant <*Access Type*> **on** <*Object*>
to <*Subject*>
[with provision <*Provisions*>**]**
where [<*Security Principal*> **has attribute** <(tag, value), …>
| <predicate>]

Strong authorisation rules facilitate the resolution of conflicting authorisations by superseding decisions from other authorisation rules. We illustrate this by an example.

Example 7: Consider the following authorisation rules:

Grant read **on** file1
to user1
with provision Add copyright notice
where user1 **has attribute** (group, accounts payable)

DoNotGrant read **on** file1
To user1
With provision Notify sysadmin
where user1 **has attribute** (rank, junior)

Suppose further that Alice, who is both a member of the accounts payable group and has rank junior, requests access to file1. The two authorisation rules above are in conflict and, therefore, the system requires a policy to resolve such conflicts.

One way to resolve such conflicts would be to have a default policy that always gives denials precedence over positive grants whenever such conflicts arise. In this case, Alice does not get access to file1. The strong authorisation is more general because it can be used to give either the positive authorisation or the negative authorisation precedence over the conflicting authorisation. For example, we can insert a strong authorisation as follows:

MustGrant read **on** file1
to user1
with provision Notify VP
where user1 **has attribute** (member, accounts payable group)
and (rank, junior)

In this case, Alice will get access to filel, but a notification will be sent to theVP.

5. MOBILE POLICY FRAMEWORK

In our framework, we make the following assumptions about the environment in which the mobile policy will execute:

- The data object and its associated policy are inseparable. That is, whenever an object moves from one component (server) to another component within the distributed computing environment, so does the associated policy. In particular, if an object is copied, the associated policy is copied as well. This requirement can be fulfilled since servers that receive copies of data objects and mobile policies are trusted to enforce the policies without altering or separating them from the data objects. The disadvantage of this trust model is that it increases the size of the Trusted Computing Base (TCB), to include all servers that receive the data object and mobile policy pair.

- To convert policy declarations in MPL into executable form, compilation of MPL to executable code is performed. The resulting code is called a *mobile policy module* [4, 5]. We are currently building a prototype that generates Java mobile policy modules. For an initial implementation of mobile policy modules, we chose Java servlets. Input to the servlets include requestor identity, attributes, and request type (i.e., read/write/execute). The output is a binary Grant or NoGrant. In addition, provisions are executed prior to servlet termination.

- Associated policies for data objects contain two components. The first component is the *policy declaration,* and the second component is the *mobile policy module.* The policy declaration specifies the policy using MPL statements. The twofold purpose of attaching the policy declaration to the object is to allow the recipients to understand the policy associated with the object and to permit modifications of the policy by the recipients of the objects. If the Grant statement in the authorisation rule contains the **with Grant Authority** clause, the recipient can modify the policy declaration and recompile it to generate a new mobile policy module to attach to the data object. The owner of the object has the option of not including the policy declaration; this is the indication to the recipient that policy modifications are not allowed. Recipient compliance with this requirement is insured since all recipients are part of the TCB.

- When the object is accessed, the system guarantees that it will always execute, and enforce, the associated policy.

- When an object O_1 with policy P_1 is copied to another object O_2, the creator of O_2 has the option of specifying new access control requirements (call it P_2) that will be enforced in addition to P_1. Therefore, the new policy for O_2 will be P_1 AND P_2.

6. APPLYING THE MPL TO THE EXAMPLES

In this section, we revisit the examples from Section 2 and show how MPL can be used to express the required policies.

Example 1 Release Markings Continued: Using MPL, we can specify the NOFORN/REL CAN marking as follows:

Grant read **on** Data_Object
to user1
where user1 **has attribute** (Citizenship, U.S.)
or (Citizenship, Canada)

Instead of defining labels that are associated with each data object, MPL uses the attributes of the subjects. The policy we defined requires user attributes (Citizenship, U.S.) or (Citizenship, Canada) to grant access.

Example 2 Originator Controlled Continued: Suppose Jeremy, from Department A, creates a new data object, which we will call Object1. Suppose further that Jeremy specifies the following access rules on Object1:

- Read access to everyone in department B
- Read and write access to Mary from Department B
- Jeremy requires that he must be consulted if any user, not in department B, is to be granted access to Object1

The following Grant statements expresses this policy:

MustGrant read **on** Object1
to user1
where user1 **has attribute** (Department, B)

MustGrant read/write **on** Object1
to Mary

DoNotGrant read **on** Object1
to *
with provision Notify Jeremy

If Mary wants to give access to Object1 to a user from department C, say Kelly, she must contact Jeremy and ask Jeremy to give access to Kelly.

If Mary copies Object1 to a new object, say Object2, Mary becomes the owner of Object2. However, since Jeremy did not grant Mary **Grant Authority,** Object2 carries over the access permissions of Object1 (as stated in Section 5). Mary still cannot grant read access to Kelly because Kelly is not included in Object1's policy, as defined by Jeremy. Mary needs to

notify Jeremy to include Kelly in his policy. If Jeremy agrees to allow access to Kelly, he defines a new policy on Object1 as follows:

MustGrant read **on** Object1
to Kelly
where Kelly **has attribute** (Department, Department C)

Mary can exercise ORCON on Object 2 and add the requirement that she be notified whenever Object2 is accessed by Kelley, as follows:

Grant read **on** Object2
to Kelly
with provision "Notify Mary"
where Kelly **has attribute** (Department, Department C)

Example 3 Label Composition Continued: In this example, two objects, Object1 and Object2, are joined to produce Object3. The respective policies of Object1 and Object 2 need to be combined so that the composite policy for Object3 preserves access rules of the two parent objects.

Now let us review Object1's policy, which states that all government employees and those non-government employees who are in ACL-A are granted access to Object1. This can be represented as follows:

Grant read **on** Object1
to *
where * **has attribute** (Employer_Type, Government)
or UserID ∈ ACL-A

Object2's policy states that all U.S. citizens and those foreign nationals who are in ACL-B are granted access to Object2. This can be represented as follows:

Grant read **on** Object2
to *
where * **has attribute** (Citizenship, US) **or** UserID ∈ ACL-B

The composite policy on Object3 can be represented as follows:

Grant read **on** Object3
to *
where * {**has attribute** (Employer_Type, Government)

or UserID ∈ ACL-A}
and { **has attribute** (Citizenship, US) **or** UserID ∈ ACL-B}

7. CONCLUSIONS AND FUTURE WORK

In this paper, we provided a flexible solution to security policy needs, such as NOFORN and ORCON, using MPL. By using MPL, we see that any kind of policy can be easily expressed. It is more expressive and flexible than previously defined methods. Our language also gives various options to enforce NOFORN/ORCON labels. It reduces the difficulty of constantly changing the labels associated with data objects. This is accomplished by using ACs. The SQL-like syntax of MPL makes it fairly easy to implement and understand. The provision clause and the **with Grant Authority** in MPL keeps owner control from being too restrictive.

One of the issues identified with the owner privilege approach [11] was of storage and performance overhead. With MPL, storage is not an issue because the policy is mobile and travels with the data; that is, it is not centralised at one site.

The advantages MPL offers include providing an easy path to implementation in a distributed environment using mobile code, providing a powerful mechanism for handling caveats and exceptions to general rules using provisions, and providing capabilities to handle exporting data objects from one system to another along with the policies associated with them.

We are currently building a prototype to demonstrate how to implement access control using mobile policies and ACs. Our prototype will include an AC parser in order to generate attribute information in text form from ACs in ASN.1 form, it will also include a policy module generator that will produce Java mobile policy modules from MPL declarations, finally we will integrate these modules into a web based demonstration.

8. REFERENCES

1. M. D. Abrams,
"Renewed understanding of access control policies''
Proc. National Computer Security Conf., 1993.

2. M. D. Abrams, K. E. Eggers, L. J. LaPadula, and I. M. Olson,
"A generalized framework for access control"
Proc. National Computer Security Conf., October 1990.

3. M. D. Abrams, J. Heaney, O. King, L. J. LaPadula, M. Lazear, and I. M. Olson,
"Generalized framework for access control: Towards prototyping the ORCON policy"
Proc. National Computer Security Conf., October 1991.

4. S. Chapin, S, Jajodia, and D. Faatz,
"Distributed Policies for Data Management Making Policies Mobile"
Proc. 14th IFIP 11.3 Working Conference on Database Security,
Schoorl, Netherlands, August 2000.

5. V. Doshi, A. Fayad, S, Jajodia, and R. Maclean,
"Using Attribute Certificates and Mobile Policies in Electronic Commerce Applications"
Proc. 16th Annual Computer Security Applications Conference,
New Orleans, LA, December 2000.

6. S. Farrell and R. Housley,
"An Internet Attribute Certificate Profile for Authorization"
PKIX Working Group, Internet-Draft, March 2000.

7. T. D. Graubart,
"on the need for a third form of access control"
Proc. National Computer Security Conf., October 1989.

8. R. Housley, W. Ford, T. Polk, and D. Solo,
"Internet Public Key Infrastructure - X.509 Certificate and CRL profile"
RFC 2459, January 1999.

9. ITU-T Recommendation X.509 (1197 E): *Information Technology - Open Systems Interconnection - The Directory: Authentication Framework,* June 1997.

10. S. Jajodia, M. Kudo, and V. S. Subrahmanian,
"Provisional authorizations"
Proc. 1st Workshop on Security and Privacy in E-Commerce,
Athens, Greece, November 2000.

11. S. Jajodia, P. Samarati, V. S. Subrahmanian, and E. Bertino,
"A Unified Framework for Enforcing Multiple Access Control Policies"
Proc. ACM SIGMOD Int'l. Conference on Management of Data,
May 1997, pages 474-485.

12. S. Jajodia, P. Samarati, and V. S. Subrahmanian,
"A logical language for expressing authorizations"
Proc. IEEE Symp. on Research in Security and Privacy,
Oakland, Calif., May 1997, pages 31-42.

13. C. J. McCollum, J. R. Messing, and L. Notargiacomo,
"Beyond the Pale of MAC and DAC – Defining new forms of access control"
Proc. IEEE Symp. on Security and Privacy, 1990, pages 190-200.