Methods for Access Control: Advances and Limitations

Ryan Ausanka-Crues
Harvey Mudd College
301 Platt Blvd
Claremont, California
rausanka@cs.hmc.edu

ABSTRACT

This paper surveys different models for providing system level access control and explores the benefits and limitations inherent to various model implementations.

Included in the model survey are Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role-Based Access Control (RBAC), Domain Type Enforcement (DTE)).

Implementations explored are matrices, access control lists (ACLs) capability lists, role based transactionsDomain Types.

Limitations covered include scalability, sparse matrices, "safety" problem, complexity, maintenance, and development costs.

Keywords

Access Controls, DAC, MAC, RBAC

1. INTRODUCTION

The application of security policies for computer systems into mechanisms of access control is a vast and varied field within computer security. The fundamental goal of any access control mechanism is to provide a verifiable system for guaranteeing the protection of information from unauthorized and inappropriate access as outlined in one or more security policies. In general, this translation from security policy to access control implementation depends on the nature of the policy but involves the inclusion of at least one of the following controls:

- *Confidentiality* - Control disclosure of information
- *Integrity* - Control modification of information

The wide array of policies, usage patterns, and protectable objects make it difficult to develop an umbrella definition of "unauthorized and inappropriate access" to guide development of a comprehensive access control model. Bishop[4] identifies *military security policies* and *commercial security policies* as two distinct types of policies that underline the difficulty in developing an all-purpose security model.

Military security policies are defined as primarily concerned with preserving information confidentiality while commercial security policies primarily focus on guaranteeing information integrity.

The dichotomy between governmental and commercial needs led to the development of two distinct access control mechanisms, Mandatory Access Control (MAC) and Discretionary Access Control (DAC). MAC focuses on controlling disclosure of information by assigning security levels to objects and subjects, limiting access across security levels, and the consolidation of all classification and access controls into the system. Conversely DAC focuses on fine-grained access control of objects through Access Control Matrices and object level permission modes.

Limitations in each model can be summarized as failings of one or more of three characteristics of an ideal security model:[6]

- *Inescapable* inability to break security policies by circumventing access controls set by the model
- *Invisible* seamless user and administrative interaction with model
- *Feasible* cost-effective and practical to implement model

Discovered limitations that break the inescapability characteristic include lack of support for principle of least-privilege, assurance violation, storage inefficiency, reliance on un-verifiable trusted components. Limitations that affect workflows, interactions, and implementation feasibility include difficulty-of-use, difficulty-of-management, incompatibility, performance hits, and high implementation costs. Many of these problems are fixable but at the cost of making the model either infeasible or more visible.

These flaws in MAC and DAC led to research in new ways of modeling and implementing access control. Among the recent developments, Role Based Access Control (RBAC) enjoys a strong and loyal following that advocates its extension of UNIX groups to incorporate features from both MAC and DAC.

2. ACCESS CONTROL MODELS

Access control models are generally concerned with whether subjects, any entity that can manipulate information (i.e. user, user process, system process), can access objects, entities through which information flows through the actions of a subject (i.e. directory, file, screen, keyboard, memory, storage, printer), and how this access can occur. Access control models are usually seen as frameworks for implementing and ensuring the integrity of security policies that mandate how information can be accessed and shared on a system. The most common, oldest, and most well-known access control models are Mandatory Access Control and Discretionary Access Control but limitations inherent to each has stimulated further research into alternatives including Role Based Access Control, Dynamic Typed Access Control, and Domain Type Enforcement.

2.1 Mandatory Access Control (MAC)

Loosely defined as any access control model that enforces security policies independent of user operations, Mandatory Access Control is usually associated with the 1973 Bell-LaPadula Model[2] of multi-level security.

2.1.1 Bell-LaPadula Confidentiality Model

Bell-LaPadula assigns security labels to subjects and objects and uses two security properties, *simple security property* and **-property* to verifiably ensure military classification policies that restrict information flow from more secure classification levels to less secure levels.

The *simple security property* states that no process may access information labeled with a higher classification.

The **-property* prevents processes from writing to a lower classification.

These two properties are supplemented by the *tranquility property*, which is stated in two forms: strong and weak. Under the *strong tranquility property*, security labels can never change during system operation. Under the *weak tranquility property*, however, labels can change during operation but never in a way that violates defined security policies.

The benefit of the weak tranquility property is that it allows least privilege by starting a user session in the lowest security session, regardless of the user's clearance level, and only upgrades the session when objects requiring higher clearance levels are accessed. Once upgraded, the session can never have its classification level reduced and all objects created or modified will have the clearance level held by the session when the object is created or modified, regardless of the objects initial level. This is known as the *high water mark principle.*

2.1.2 Biba Integrity Model

While Bell-LaPadulas model describes methods for assuring confidentiality of information flows, Biba[3] developed a similar method aimed at information integrity. Integrity is maintained through adherence to reading writing principles that can be thought of as a reverse of the Bell-LaPadula principles. In the Biba model, integrity levels are low to high with objects labeled high having high integrity. A subject can read objects at a higher level but can only write to objects of lower levels. This is known as the *low water mark principle* and assigns created objects the lowest integrity level that contributed to the creation of the object. Because the MAC method is primary developed for purposes where confidentiality is far more important than integrity, Bibas influence was minor on further development of MAC models.

2.1.3 Benefits

Through its implementation of Bell-LaPadula in Multi-Layer Secure (MLS) systems, MAC is the main access control model used by the military and intelligence agencies to maintain classification policy access restrictions. The combination of Bell-LaPadula and trusted component assurance also has the nice benefit of making MLS systems immune to Trojan Horse attacks. In perfect implementations, MLS systems implementing Bell-LaPadula MAC are not susceptible Trojan Horse forced security violations because users do not have the ability to declassify information. Additionally, MAC is relatively straightforward and is considered a good model for commercial systems that operate in hostile environments (web servers and financial institutions) where the risk of attack is very high, confidentiality is a primary access control concern, or the objects being protected are valuable.[9]

2.1.4 Problems

MAC, however, is not without serious limitations. The assignment and enforcement of security levels by the system under the MAC model places restrictions on user actions that, while adhering to security policies, prevents dynamic alteration of the underlying policies, and requires large parts of the operating system and associated utilities to be "trusted" and placed outside of the access control framework.

Trusted components are processes and libraries, such as declassifying cryptographic processes, that need to violate MAC principles and thus must sit outside of the MAC model. In order to maintain the security policies and prevent unauthorized or inappropriate access, the code behind these components is assumed (hopefully with verification) to be correct and conforming to the underlying security policies of the system. Additional access control methods must be used to restrict access to these trusted components. Unfortunately, in practice it has been shown that it is virtually impossible to implement MLS using MAC without moving essentially the entire operating system and many associated utilities outside the MAC model and into the realm of trusted components.[1]

Additionally, MAC can unnecessarily over-classify data through the high-water mark principle and hurt productivity by limiting the ability to transfer labeled information between systems and restricting user control over data. MAC also does not address fine-grained least privilege, dynamic separation of duty or security or validation of trusted components.

MAC systems are difficult and expensive to implement due to the reliance on trusted components and the necessity for applications to be rewritten to adhere to MAC labels and properties. It is rumored that application developer reluctance to take these steps is the reason behind Microsofts abandonment of MAC related trusted computing in their long awaited new operating system.[10]

2.2 Discretionary Access Control (DAC)

MAC, while immensely important to military applications, is not the most widely used method of access control. That distinction belongs to DAC largely thanks to spawning from primarily commercial and academic research as well as the integration of DAC Access Control integration into UNIX, FreeBSD, and Windows 2000. DAC was developed to implement Access Control Matrices defined by Lampson in his paper on system protection.[8] Access Control Matrices are usually represented as three dimensional matrices where rows are subjects, columns are objects and the mapping of subject and object pairs results in the set of rights the subject has over the object.

Unlike the MAC framework where decisions to allow or deny access are made by the system according to pre-determined policies, DAC allows subjects the discretion to decide access rights on objects they own. Because Access Control Matrices have one row for every subject and one column for every object, the number of entries is intuitively the number of subjects times the number of objects. This means that $O(n)$ growth in subjects and objects results in $O(n^2)$ growth in the size of the matrix. The size of the access control matrix would not be a concern if the matrix was dense, however, most subjects have no access rights on most objects so, in practice, the matrix is very sparse. If access control information was maintained in this matrix form, large quantities of space would be wasted and lookups would be very expensive. Thus, DAC access settings are typically stored as either per-object file permission modes (default on UNIX) or as lists.

2.2.1 *Lists*

Access lists can be stored in a number of configurations with each configuration offering benefits and drawbacks under varying circumstances.

Access Control Lists(ACLs)

Access control lists are the representation of object rights as a table of subjects mapped to their individual rights over the object. ACLs are the default representation of access rights on UNIX systems and essentially correspond to individual columns in the system Access Control Matrix. ACLs are effective but not time-efficient with a low number of subjects. Typically, the operating system knows who the user of a process is but doesnt know what rights the user has over objects on the system. ACLs require the operating system to either perform a rights lookup on each object access or somehow maintain the subjects active access rights. Because of this rights management issue, and the difficulty in performing multi-object rights modifications for individual users, ACLs dont scale well on systems with large numbers of subjects or objects.

Capabilities Lists

Capabilities Lists are similar to ACLs but instead of tables of subjects and rights, capability lists represent subject rights as mappings of objects to rights.

2.2.2 *Benefits*

A primary benefit associated with the use of DAC is enabling fine-grained control over system objects. Through the use of fine-grained controls, DAC can easily be used to implement least-privilege access. Individual objects can have access control restrictions to limit individual subject access to the minimum rights needed. DAC is also intuitive in implementation and is mostly invisible to users so it is regarded as the most cost-effective for home and small-business users.[14]

2.2.3 *Problems*

DAC, however, is not without issues. Allowing users to control object access permissions has a side-effect of opening the system up to Trojan horse susceptibility. Additionally maintenance of the system and verification of security principles is extremely difficult for DAC systems because users control access rights to owned objects. The so-called “Safety Problem, the lack of constraints on copy privileges, is another liability inherent to DAC. The lack of constraints on copying info from one file to another makes it difficult to maintain safety policies and verify that safety policies have are not compromised while opening potential exploits for Trojan horses.

2.3 Role-Based Access Control(RBAC)

RBAC is considered a much more generalized model than either MAC or DAC, encompassing both models as special cases while providing a policy neutral framework that allows RBAC to be customized on a per-application basis.[12] As a blend of the MAC and DAC models and integri, RBAC is partially founded on principles outlined in Biba.

2.3.1 *Differences from MAC/DAC*

First proposed by Ferraiolo and Kuhn in 1992,[7] RBAC addresses most of the failings of DAC while maintaining DACs focus on non-military systems. RBAC approaches commercial and civilian governmental security needs from an integrity first, confidentiality second position based on Clark and Wilsons research into commercial security policies.[5] Security policies are maintained in RBAC through the granting of rights to roles rather than individuals. Right granting and policy enforcement is consolidated in the hands of a security administrator and users are prevented from transferring permissions assigned to a role they are allowed to perform to other users. This rule removes the ownership rights granted in DAC and thus behaves like a finer-grained version of the MAC model. In fact, this is precisely how Ferraiolo and Kuhn see RBAC.

RBAC also stands apart from the more traditional MAC and DAC by granted rights on transactions, not on underlying subjects. These rights are granted to roles, which at first glance appear to be a synonym for DAC groups. The difference lies in that groups consist of a collection of users while roles are a bridge between a collection of users and a collection of the Clark-Wilson model of transaction rights.[5]

2.3.2 *Benefits*

Transaction based rights help ensure system integrity and availability by explicitly controlling not only which resources can be accessed but also how access can occur. In large organizations, the consolidation of access control for many users into a single role entry allows for much easier management

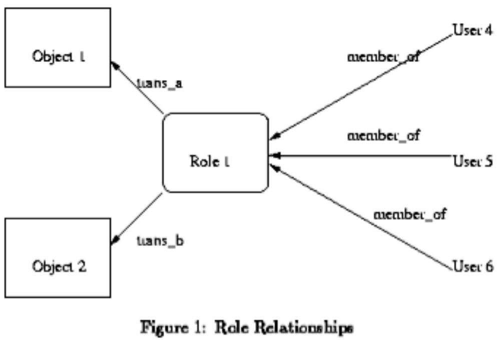

Figure 1: Example of RBAC role relationships

of the overall system and much more effective verification of security policies.

Another benefit of RBAC is integrated support for principle of least-privilege, separation of duties, and central administration of role memberships and access controls. Separation of duties and least-privilege are not a part of MAC while central administration is loosely supported in MAC with trusted components and impossible in DAC due to the violation of the safety principle.

2.3.3 Problems

While RBAC marks a great advance in access control, the administrative issues of large systems still exist, albeit in a markedly more manageable form. In large systems, memberships, role inheritance, and the need for finer-grained customized privileges make administration potentially unwieldy.

Additionally, while RBAC supports data abstraction through transactions, it cannot be used to ensure permissions on sequences of operations need to be controlled.[11] To do this, a less general and more sophisticated access control model must be used.

2.4 Domain Type Enforcement(DTE)

Domain Type Enforcement (DTE) is an extension of Type Enforcement (TE) and is itself extended into Dynamic Typed Access Control (DTAC). The principle of type enforcement is more that flexible policy expressions are possible when objects are assigned to types and thus columns in the access control matrix are replaced by types.[16] The DTE extension to this is to assign subjects to domains and complete the matrix transformation so the access control matrix is now a domain definition table (DDT) with rows of domains and columns of types. DTAC expanded upon this to include RBAC type administrative controls.[15] It is claimed that DTE models can implement the Bell-LaPadula confidentiality model as well as some of the more robust integrity features in DAC and RBAC. As of yet, this is not demonstratable.[13]

3. CONCLUSIONS

Access Control models have come quite a ways since the initial implementations of MAC and DAC in the early 70's. Researchers have learned volumes about the complexities of maintaining security policies through model applications and with RBAC have come very close to seamlessly integrating integrity and confidentiality.

Work still needs to be done on translating policies into verifiable model implementations and in efficient and accurate management of these implementations.

Future work in the area of models for access control is likely to be focused on the proliferation of Role-Based Access Control models and case study analysis of their relative effectiveness. Oracle already has incorporated RBAC as part of their database management access controls as has the SQL:1999 standard, PostgreSQL, and SAP. Solaris, Windows Active Directory, and SELinux all also provide support for the use of Role Based Access Control.

Additionally, further development of new models DTAC and Attribute-Based Access Control (ABAC) and the specialization of these and other access control models is very likely. Operating systems are also likely to expand support for additional access control models both internally and with Pluggable Policy Modules to allow users and administrators more comprehensive and user-friendly ways to secure systems.

4. REFERENCES

[1] R. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems.* Wiley Computer Publishing, New York, New York, 2001.

[2] D. Bell and L. LaPadula. Secure computer system: Unified exposition and multics interpretation. *TR M74-244*, March 1976.

[3] K. Biba. Integrity considerations for secure computer systems. *Technical Report MTR-3153*, April 1977.

[4] M. Bishop. *Computer Security: Art and Science.* Addison-Wesley Publishing Company, Boston, Massachusetts, 2004.

[5] D. D. Clark and D. R. Wilson. A comparison of commercial and military computer security policies. *IEEE Symposium on Security and Privacy*, pages 184–194, April 1987.

[6] J. Daniel E. Geer. The shrinking perimeter: Making the case for data-level risk management. *Veradsys White Paper*, January 2004.

[7] D. Ferraiolo and R. Kuhn. Role-based access control. *15th National Computer Security Conference*, pages 554–563, October 1992.

[8] B. W. Lampson. Protection. *ACM SIGOPS Operating System Review*, 8(1):18–24, January 1974.

[9] M. L. Matthews. Position paper. In *SACMAT'01*, page 144. SACMAT, May 2001.

[10] A. Orlowski. Ms trusted computing back to drawing board. *The Register*, May 2004.

[11] H. L. F. Ravi S. Sandhu, Edward J. Coyne and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, February 1996.

[12] R. Sandhu. The next generation of access control models: Do we need them and what should they be? In *SACMAT'01*, page 53. SACMAT, May 2001.

[13] C. Schaulfler. They want froot loops: Why industry will continue to deliver multi-level security. In *SACMAT'01*, pages 145–146. SACMAT, May 2001.

[14] S. Smalley. Which operating system access control technique will provide the greatest overall benefit to users? In *SACMAT'01*, page 147. SACMAT, May 2001.

[15] J. A. Solworth and R. H. Sload. Security property based administrative controls. 2005.

[16] R. Watson. Statement for the sacmat 2001 panel. In *SACMAT'01*, page 149. SACMAT, May 2001.