



# **COURSE PROJECT**

## Mandatory Access Control

---

**Fred Schneider**

Faculty Author  
Cornell Bowers Computing and Information Science

**Cornell University**

Project Completed by: Germaine Wong

## COURSE PROJECT — PART ONE

# Multi-Level Security

1. For this part of the course project, you will identify which parts of your authorization relation could be implemented with multi-level security, then you will revise your authorization mechanisms for these components. You should specify the set of labels that will be used, who will assign labels to principals and objects, what restrictions will be enforced, and whether/how labels can be modified. Feel free to revise some of your designs from the previous module, if appropriate.

The parts of my authorization relation that could be implemented with MLS are grant and modify.

Revisions:

Access decisions for the grant command are based on integrity levels (low/medium/high), confidentiality labels, and mandatory access policies. The grant command allows the end user's token to send encrypted messages and establish an authorization relationship, in accordance with security policies that govern access to resources that are based on classification levels of subject (user), objects (resources) and labels associated with them. Integrity labels focus on making sure messages have not been tampered with, modified, or corrupted. Access control rules are: no read up, no write down. To enforce, the system issues a grant to a user to access the confidential message if the user's clearance matches. Labels will be assigned by system administrator.

Access decisions for the grant command are governed by mandatory policies and security labels. The grant command establishes new authorization relationships for principals to gain access to resources, in accordance with MLS policies and security label-based access controls. When the grant command is used to establish a new authorization relationship, it will assign a specific security label to the resource, principal (user/device) or both. MLS policies are used to control the granting of access. Access control is established by associating a principal with a resource based on security labels. The clearance level needs to be greater than or equal to the resource's level. The integrity level ensures users with a high level cannot modify data labeled with a low level. To enforce integrity: a system issues a grant command to allow a system admin with a high integrity clearance to modify a high integrity program, and a user with a low integrity clearance to modify a low integrity program. To enforce confidentiality: the system issues a grant



to a system admin to access a confidential file if the clearance matches it. Labels will be assigned by system admin.

Access decisions for the modify command are based on security labels with clearance and classification levels. The modify command allows for the adjustment of a principal's permissions (elevation from read-only to send-and-receive), following MLS policies and security clearance. The system issues a modify command to upgrade permissions and checks the security clearance to ensure a principal can access confidential resources and also checks integrity clearance to allow send-and-receive actions. Once this is done, the permissions are updated in the ACL and the system makes sure the change is compliant with MLS policies (no write-down or read-up). Enforcement: If a principal attempts to access a resource above their clearance, access is denied based on security labels. Labels will be assigned by system admin.

In terms of reclassification of encrypted communications with MLS, a high label must be assigned to an encrypted object  $L(F) < L(Enc(F))$ . When you encrypt plaintext with a key, you reclassify it, giving it a new label.

### **Facilitator's Comments**

Very well detailed and documented.

### **Grade**

**100%**



## COURSE PROJECT — PART TWO

# Commercial Access Control

**1. You now need to decide which parts of your authorization policy could be implemented with either (1) commercial security policies or (2) role-based access control. You should then design an access control scheme to authorize those components. Feel free to revise some of your designs from Part One and/or the previous module, if appropriate.**

For a secure, anonymous communication system, RBAC would be the best way to manage access control. The parts of my authorization policy that can be implemented with RBAC include grant, revoke, modify, and audit.

Roles: end user (revoke), sys admin (grant/revoke/modify access privileges), auditor (audit logs)

The first system admin can be given permission to establish authorization relationships via the grant command. The system admin is given their role by the security admin, along with permission to execute the grant command via security policy. Separation of duty means not all system admins will have this role and the admin cannot occupy another role when executing a grant command. When the system admin issues the command in order to create new authorization relationships, the system/reference monitor has to check if the admin has the grant authorization permission tied to their role, and, if true, the command executes. Each grant executed by the system administrator is logged to ensure proper audits can be conducted.

An end user can be given the permission tied to their role to terminate their own sessions, if need be, due to an unauthorized device being logged in or due to suspicious activity the user notices. A user cannot be acting on behalf of another user in order for separation of duties to work. The user role would have the permission "revoke own session" and the reference monitor would check if the user is permitted to invoke the revoke command and if the session ID is tied to the user, and, if true, the session is terminated . Revoke actions are logged with the user. The system mediates the user's access attempts to ensure consistency with the privileges associated with the roles occupied by the user.

The second system admin can be given the permission to revoke any session, and separation of duties are enforced by preventing the system admin with the grant command from invoking the revoke command (R, R'). Also, the system admin cannot be in another role during revoke command execution. Only the second system admin role would have the permission to execute



the revoke command and revoke third party access privilege. When the admin issues the revoke command, the reference monitor checks to see if user has the system admin's role and if it contains the revoke auditor permissions, and, if true, the system enforces the revocation. A log is created for each revoke action that includes the system admin's role and their identity.

A system admin can be given the permission to modify access to third party auditors. In order to have separation of duty, a system admin would propose a modify, but a security access manager would have to approve it. The system admin would have to have the "modify third party auditor" permission, and the reference monitor would check if the system admin's role has that permission before an access modification could be proposed and sent to the security access manager for approval.

For a third party auditor, the system admin creates a role for the auditor. This role includes the "invoke audit command" permission, and a read-only permission. The reference monitor checks if the user has the auditor role and checks whether or not the role includes the invoke audit command permission, and if requirements met, the auditor is granted access to audit logs. The third party auditor's audit command is logged to ensure accountability. This role is given least privileged access because an auditor does not need to access other data, such as system configurations, to be able to perform an audit.

### **Facilitator's Comments**

Its great to see how well you were able to be descriptive in designing the access control scheme of the required components.

### **Grade**

**100%**



## COURSE PROJECT — PART THREE

# Credentials-Based Authorization

1. The final step is to identify one component of your authorization policy that could be implemented with credentials-based authorization. You will design credentials and guards for your system then specify what state the guards will store and what credentials a principal must present when they make a request. Feel free to revise some of your designs from Part One and Part Two, if appropriate.

The component of my authorization policy that could be implemented with credentials-based authorization is the modify command used by the system admin for third party auditor access. In this authorization scheme the guard stores the authoritative credentials (policy authority says Alice is a sysadmin) and Alice provides the object credentials/label assignment. Credentials for my secure anonymous communication system include: identity, which proves who makes the request; label-assignment, which comes from the label authority with the label of "sysadmin"; and object-labeled, which is issued by the owner and says what label an object has. Guards for this system include: credential validity (identity, labels), and policy logic (only system admins can modify auditor access). These guards create a defense-in-depth mitigation strategy. The states that the guard will store include: (1) policy rules - intensional formulas that define who may modify auditor access; (2) authoritative label assignments - bindings between principal and label, labels are issued by a policy authority or human resources; (3) authoritative object levels - bindings between objects and labels, guard may store signed credentials from the owner; (4) root of trust verification - CA root for identity certificates, label authority, encryption key, object owner key to verify credential signatures. The credentials a principal must present when they make a request include: (1) identity credential for sysadmin (signed by CA); (2) signed modify request - operation, time stamped, signed by sysadmin's private key; (3) object-level credential OR label assignment (the guard will provide the one not provided by the principal). The guards in the authorization scheme must share the same policy rules and roots of trust in order for it to work. Also, audit logs should be merged to ensure consistency.



## **Facilitator's Comments**

Great provided details and content. Very well done.

## **Grade**

**100%**



Cornell University

© 2024 Cornell University



Provide Feedback on this PDF

Submit Feedback



Cornell University

© 2024 Cornell University