Access Control Systems

Design Laboratory(CS59001) report submitted to

Indian Institute of Technology Kharagpur

in partial fulfillment for the award of the degree of Bachelor of Technology

in

Computer Science and Engineering

by

Khushboo Singhania
(19CS30023)

Under the supervision of

Professor Sural Shamik

and

Professor Swagato Sanyal

Department of Computer Science and Engineering,

Indian Institute of Technology, Kharagpur
Autumn Semester, 2024-25

November 03, 2024

Contents

Abbreviations

RBAC	Role_based Access Control System
ABAC	Attribute-based Access Control System

Chapter 1

Introduction

Starting in the 1970s, computer systems featured multiple applications and served multiple users, leading to heightened awareness of data security issues. System administrators and software developers alike focused on different kinds of access control to ensure that only authorized users were given access to certain data or resources.

An access control system refers to a security system that is designed to limit and control employee access to certain areas or resources within a facility. It uses various forms of verification, such as personal identification numbers, cards, tokens, fingerprints, or iris recognition, to grant or deny access. Access control systems range in size and complexity, from a single door keypad to thousands of doors and sensors connected through a computer network. They are cost-effective alternatives to issuing physical keys and provide enhanced security for organizations.

Chapter 2

RBAC

2.1 Introduction

One kind of access control that emerged is role-based access control (RBAC).

A role is chiefly a semantic construct forming the basis of access control policy. With RBAC, system administrators create roles according to the job functions performed in a company or organization, grant permissions (access authorization) to those roles, and then assign users to roles on the basis of their specific job responsibilities and qualifications.

Sophisticated variations of RBAC include the capability to establish relations between roles, between permissions and roles, and between users and roles. For example, two roles can be established as mutually exclusive-the same user is not allowed to assume both. Roles can also acquire inheritance relations, whereby one role inherits permissions assigned to a different role. These role-role relations can enforce security policies, including separation of duties and delegation of authority. Previously, these relations would have required application software encoding; with RBAC, they can be specified once for a security domain. With RBAC, role-permission relationships can be predefined, which makes it simple to assign users to the predefined roles.

Access control policy is embodied in RBAC components such as role-permission, user-role, and role-role relationships. These components collectively determine whether a particular user is allowed access to a certain piece of system data. RBAC components can be configured directly by the system administrator or indirectly by appropriate roles as delegated by the system administrator. The policy enforced in a given system results from the specific configuration of RBAC components as directed by the system administrator. Because the access control policy can, and usually does, change over the system life cycle, RBAC offers a key benefit through its ability to modify access control to meet changing organizational needs.

2.2 RBAC data generation$_0$

Input: A program has been coded that takes user input of total numbers of users, total number of roles and total number of permissions for RBAC.

Constraints: It also asks for the maximum number of roles that can be assigned to a user and the maximum number of permissions that can be assigned to a role.

Output: Random assignment of user-role and role-permission combinations subject to above written constraints.

2.3 RBAC data generation$_1$

Input: A program has been coded that takes user input of total numbers of users, total number of roles and total number of permissions for RBAC.

Constraints: Previous constraints of RBAC data generation$_0$ as well as maximum number of users that have a role and maximum number of roles that have a permission.

Output: Random assignment of user-role and role-permission combinations subject to above written constraints.

2.4 RBAC$_0$

Input: A program has been coded that takes user input of user id and permission id for RBAC.

Constraints: It has no constraints.

Output: Printes whether the given user can access the requested permission.

2.5 RBAC role hierarchy$_0$

Input: A program has been coded that takes user input of ordered pairs of role-role combinations to establish hierarchy.

Constraints: It assumes that the user will not enter role-role combinations such that a cycle forms.

Output: Writes the given hierarchy into file for future use subject to above constraint.

2.6 RBAC$_1$

Input: A program has been coded that takes user input of user id and permission id for RBAC.

Constraints: It also checks if the inherited roles of the roles user has can access requested permission.

Output: Prints whether the given user can access the requested permission subject to above constraint.

2.7 Role-Permission Input

Input: A program has been coded that takes user input of ordered pairs of role-permission combinations.

Constraints: It checks for the 4 basic constraints that RBAC data generation$_1$ follows.

Output: Writes the role-permission combination into file for future use subject to the constraints.

2.8 User-Role Input$_0$

Input: A program has been coded that takes user input of ordered pairs of user-role combinations.

Constraints: It checks for the 4 basic constraints that RBAC data generation$_1$ follows.

Output: Writes the user-role combination into file for future use subject to the constraints.

2.9 RBAC role exclusivity

Input: A program has been coded that takes user input of ordered pairs of role-role combinations to establish exclusivity.

Constraints: It takes into account role hierarchy. If role hierarchy present between the input roles, then role exclusivity can’t be established.

Output: Writes the given role exclusivity into file for future use subject to above constraint.

2.10 RBAC role hierarchy$_1$

Input: A program has been coded that takes user input of ordered pairs of role-role combinations to establish hierarchy.

Constraints: It assumes that the user will not enter role-role combinations such that a cycle forms and takes into account role exclusivity. If role exclusivity present between the input roles, then role hierarchy can’t be established.

Output: Writes the given hierarchy into file for future use subject to above constraint.

2.8 User-Role Input$_1$

Input: A program has been coded that takes user input of ordered pairs of user-role combinations.

Constraints: It checks for the 4 basic constraints that RBAC data generation$_1$ follows. It also checks if the input role is mutually exclusive to any of the roles that user already has. If yes, then input user-role combination is not established.

Output: Writes the user-role combination into file for future use subject to the constraints.

Chapter 3
ABAC

3.1 Introduction

ABAC is an access control method where subject requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions.

Attributes are characteristics of the subject, object, or environment conditions. Attributes contain information given by a name-value pair.

A **subject** is a human user, such as a device that issues access requests to perform operations on objects. Subjects are assigned one or more attributes. Assume that subject and user are synonymous.

An **object** is a system resource for which access is managed by the ABAC system, such as devices, files, records, tables, processes, programs, networks, or domains containing or receiving information. It can be the resource or requested entity, as well as anything upon which an operation may be performed by a subject including data, applications, services, devices, and networks.

An **operation** is the execution of a function at the request of a subject upon an object. Operations include read, write, edit, delete, copy, execute, and modify.

Policy is the representation of rules or relationships that makes it possible to determine if a requested access should be allowed, given the values of the attributes of the subject, object, and possibly environment conditions.

3.2 ABAC data generation

Input: A program has been coded that takes user input of total numbers of subjects, subject attributes, possible subject attribute values, objects, object attributes, possible object attribute values, operations and rules.

Output: Random assignment of user attributes to user attribute value for every user, object attributes to object attribute value for every object and creation of rules.

3.3 ABAC

Input: A program has been coded that takes user input of user id, object id and action id for ABAC.

Output: Prints whether the given user can perform the requested action on given object according to the rules.

Bibliography

Role-Based Access Control Models by Ravi S. Sandhu (George Mason University and SETA Corporation), Edward J. Cope, Hal L. Feinstein, Charles E. Youman, SETA Corporation

Guide to Attribute Based Access Control (ABAC) Definition and Considerations at https://doi.org/10.6028/NIST.SP.800-162