40. RBAC Policy DSL Compiler with Static Security & Privilege Escalation Detection

Course Name: Compiler Design

Course Code: CS1202

Name: Pinikeshi Meghana

Roll No: 24CSB0A54

Sec: CSE-A

Document Details: Week 3 Deliverables

Week 3 – Requirement Analysis & Threat Modeling

1. Introduction

Week 3 focuses on Requirement Analysis and Threat Modeling for the RBAC Policy DSL Compiler. The goal of this phase is to clearly define the expected system functionality, operational constraints, and security objectives before moving to design and implementation.

A well-defined requirement analysis ensures that RBAC policies are analyzed correctly and securely. In this week, both functional and non-functional requirements are documented using a Software Requirements Specification (SRS). In addition, threat modeling is performed to identify common RBAC misconfigurations such as over-privileged roles, misuse of role inheritance, and transitive privilege escalation.

This phase establishes a structured foundation for implementing a compiler-based static analysis framework that detects RBAC security issues at design time.

2. Software Requirements Specification (SRS)

2.1 Purpose

The purpose of this document is to specify the functional and non-functional requirements of the proposed RBAC Policy DSL Compiler with Static Security Analysis.

This Software Requirements Specification (SRS) clearly defines:

- What the system is expected to do
- The security problems it aims to address
- The constraints under which it will operate

The RBAC Policy DSL Compiler is intended to:

- Allow structured specification of RBAC policies using a DSL
- Perform compile-time semantic validation and static security analysis
- Detect RBAC misconfigurations such as role conflicts, redundant permissions, and privilege escalation before deployment

This document serves as a reference for system design, implementation, and evaluation in subsequent project phases.

2.2 Scope

The proposed system is a compiler-based static analysis framework for RBAC policies defined using a Domain-Specific Language (DSL). The scope of the system includes the specification, validation, and analysis of RBAC policies at compile time.

In Scope

The system will:

- Accept RBAC policies written in a well-defined DSL
- Parse RBAC policy specifications and generate an Abstract Syntax Tree (AST)
- Perform semantic validation of RBAC components such as:
  - Users
  - Roles
  - Permissions
  - Role hierarchies
- Statistically analyze RBAC policies to detect:
  - Role conflicts
  - Redundant permissions
  - Over-privileged roles
  - Direct and indirect privilege escalation
- Provide explainable and human-readable security reports
- Prevent insecure RBAC policies from being approved for deployment

Out of Scope

The system will not:

- Perform runtime access control enforcement
- Integrate with live enterprise or cloud IAM systems
- Support access control models other than RBAC (e.g., ABAC, MAC)
- Handle dynamic policy updates during execution
- Enforce authentication or user identity verification

The focus of this project is strictly on "static, compile-time analysis of RBAC policies" to ensure security by design.

2.3 Definitions, Acronyms, and Abbreviations

This section defines key terms and abbreviations used throughout this document to ensure clarity and consistency.

- **RBAC (Role-Based Access Control):**
  An access control model in which permissions are assigned to roles, and users are granted access by being assigned to these roles.
- **DSL (Domain-Specific Language):**
  A specialized language designed to express RBAC policies in a structured, readable, and analyzable form.
- **AST (Abstract Syntax Tree):**
  A tree-based representation generated after parsing the DSL, used for semantic validation and static analysis.
- **IAM (Identity and Access Management):**
  Systems used to manage user identities and control access to resources in enterprise and cloud environments.
- **Privilege Escalation:**
  A security flaw where a user gains access to permissions beyond their intended authorization, often due to role inheritance or misconfiguration.
- **Role Hierarchy:**
  A structured relationship between roles where higher-level roles inherit permissions from lower-level roles.
- **Static Analysis:**
  Analysis performed without executing the system, used to detect errors and security issues at compile time.
- **Semantic Validation:**
  The process of verifying the correctness and consistency of RBAC policy components beyond syntax checking.
- **Compile-Time:**
  The phase in which the DSL is parsed and analyzed before policy deployment or execution.

2.4 Overall Description

This section provides a high-level overview of the RBAC Policy DSL Compiler, including its perspective, core functions, user categories, and operating environment.

2.4.1 Product Perspective

The RBAC Policy DSL Compiler is a “standalone static analysis tool” inspired by traditional compiler design principles. It treats RBAC policies as formal specifications, similar to programs, and applies parsing, semantic analysis, and security checks before policy deployment.

The system does not replace existing IAM platforms; instead, it complements them by providing design-time security verification. The output of the system can be used by policy designers to correct misconfigurations before policies are enforced in real systems.

2.4.2 Product Functions

The primary functions of the system include:

- Parsing RBAC policies written in the proposed DSL
- Constructing an Abstract Syntax Tree (AST)
- Performing semantic validation of RBAC components
- Analyzing role hierarchies and permission inheritance
- Detecting role conflicts and constraint violations
- Identifying redundant permissions
- Detecting direct and indirect privilege escalation
- Generating explainable and structured security reports

2.4.3 User Classes

The system is intended for the following users:

- **Security Engineers:** To validate RBAC policies before deployment
- **System Administrators:** To analyze and debug complex role hierarchies
- **Policy Designers:** To author correct and secure RBAC policies using the DSL
- **Researchers and Students:** To study compiler-based security analysis techniques

2.4.4 Operating Environment

- The system will be implemented as a command-line based static analysis tool
- Runs on standard operating systems such as Linux or Windows
- Requires a standard development environment (e.g., Java or Python runtime)
- No dependency on external IAM systems or network connectivity

3. Functional Requirements

The system shall support the following functional requirements.

FR1. RBAC DSL Parsing

- The system shall accept RBAC policies written in the defined Domain-Specific Language (DSL).
- The system shall perform lexical and syntactic analysis of the DSL input.

FR2. Abstract Syntax Tree (AST) Generation

- The system shall generate an Abstract Syntax Tree (AST) from the parsed RBAC DSL.
- The AST shall represent users, roles, permissions, role hierarchies, and constraints.

FR3. Semantic Validation of RBAC Policies

- The system shall validate the semantic correctness of RBAC policy elements.
- The system shall detect undefined roles, users, or permissions.
- The system shall verify consistency of role assignments and role hierarchies.

FR4. Role Hierarchy Analysis

- The system shall analyze role inheritance relationships.
- The system shall verify correct permission propagation across role hierarchies.

FR5. Role Conflict Detection

- The system shall detect conflicts caused by incompatible role assignments.
- The system shall identify violations of separation of duty constraints.

FR6. Privilege Escalation Detection

- The system shall detect direct privilege escalation due to improper role assignments.
- The system shall detect indirect or transitive privilege escalation through role inheritance.

FR7. Redundant Permission Detection

- The system shall identify redundant permissions assigned through multiple roles.
- The system shall report unnecessary or duplicate permission propagation.

FR8. Over-Privileged Role Detection

- The system shall identify roles that grant excessive permissions beyond intended scope.

FR9. Security Report Generation

- The system shall generate a structured security analysis report.
- The report shall provide human-readable explanations of detected issues.
- The report shall guide users in correcting RBAC misconfigurations.

4. Non-Functional Requirements

The system shall satisfy the following non-functional requirements.

NFR1. Static Analysis

- The system shall perform all RBAC policy analysis at compile time.
- No runtime execution or monitoring of policies shall be required.

NFR2. Deterministic Behavior

- Given the same RBAC policy input, the system shall always produce the same analysis results.
- The analysis shall not depend on external or dynamic factors.

NFR3. Explainability

- The system shall generate clear and human-readable explanations for all detected security issues.
- The output shall help users understand the cause of misconfigurations.

NFR4. Scalability

- The system shall efficiently analyze RBAC policies with a large number of users, roles, and permissions.
- Performance shall remain acceptable for enterprise-scale RBAC configurations.

NFR5. Modularity and Extensibility

- The system shall be designed in a modular manner.
- New security checks and analysis rules shall be easily extendable.

NFR6. Reliability

- The system shall correctly detect RBAC misconfigurations without producing false positives where possible.
- The analysis results shall be consistent and dependable.

5. Threat Modeling

Threat modeling is performed to identify and analyze potential security risks arising due to misconfigured RBAC policies. This helps in understanding how improper role and permission assignments can lead to security violations and guides the design of effective static analysis checks.

5.1 Identified RBAC Threats

The following major RBAC-related threats are considered in this project:

T1. Over-Privileged Roles

- Roles assigned with more permissions than required
- Violates the principle of least privilege
- Increases attack surface if a role is compromised

T2. Role Inheritance Abuse

- Improper use of role hierarchies
- Child roles unintentionally inherit sensitive permissions
- Leads to privilege expansion across roles

T3. Transitive Privilege Escalation

- Privileges gained indirectly through multi-level role inheritance
- Difficult to detect manually in complex role hierarchies
- Can grant unauthorized access paths

T4. Conflicting Roles

- Users assigned to mutually exclusive roles
- Violates separation of duty constraints
- Can lead to fraud or policy violations

T5. Redundant Permissions

- Same permission granted through multiple roles
- Causes policy bloating and poor maintainability
- Makes security analysis more complex

5.2 Threat Modeling Objectives

The main objectives of threat modeling in this project are:

- Identify RBAC misconfigurations at design time
- Understand how role and permission relationships lead to security threats
- Enable early detection of privilege escalation scenarios
- Prevent insecure RBAC policies from being deployed
- Support the design of targeted static analysis techniques

6. Threat Model Diagram

This section presents the threat model diagram for the RBAC Policy DSL Compiler. The diagram visually represents how users, roles, permissions, and role hierarchies interact, and how misconfigurations in these relationships can lead to security threats such as privilege escalation and role conflicts.

6.1 Purpose of the Threat Model Diagram

The threat model diagram is used to:

- Visualize RBAC entities and their relationships
- Identify possible privilege escalation paths
- Highlight misuse of role inheritance
- Illustrate how security threats originate from policy misconfigurations
- Support mapping of threats to static analysis techniques

6.2 Description of the Diagram

The threat model diagram includes the following components:

- **Users**
    - Assigned one or more roles
    - Can inherit permissions indirectly through roles
- **Roles**
    - Act as an intermediate layer between users and permissions
    - Can have hierarchical relationships (parent–child roles)
- **Permissions**
    - Represent access rights to system resources
    - Assigned to roles rather than directly to users

- **Role Hierarchies**
    - Show inheritance relationships between roles
    - Can introduce indirect permission propagation
- **Threat Paths**
    - Paths indicating:
        - Direct privilege escalation
        - Transitive privilege escalation through role hierarchies
        - Conflicting role assignments

6.3 Security Threats Highlighted in the Diagram

The diagram explicitly highlights the following threats:

- Over-privileged roles receiving excessive permissions
- Unauthorized permission inheritance via role hierarchies
- Indirect privilege escalation across multiple role levels
- Conflicting roles assigned to the same user
- Redundant permission assignments across roles

6.4 Diagram Placement

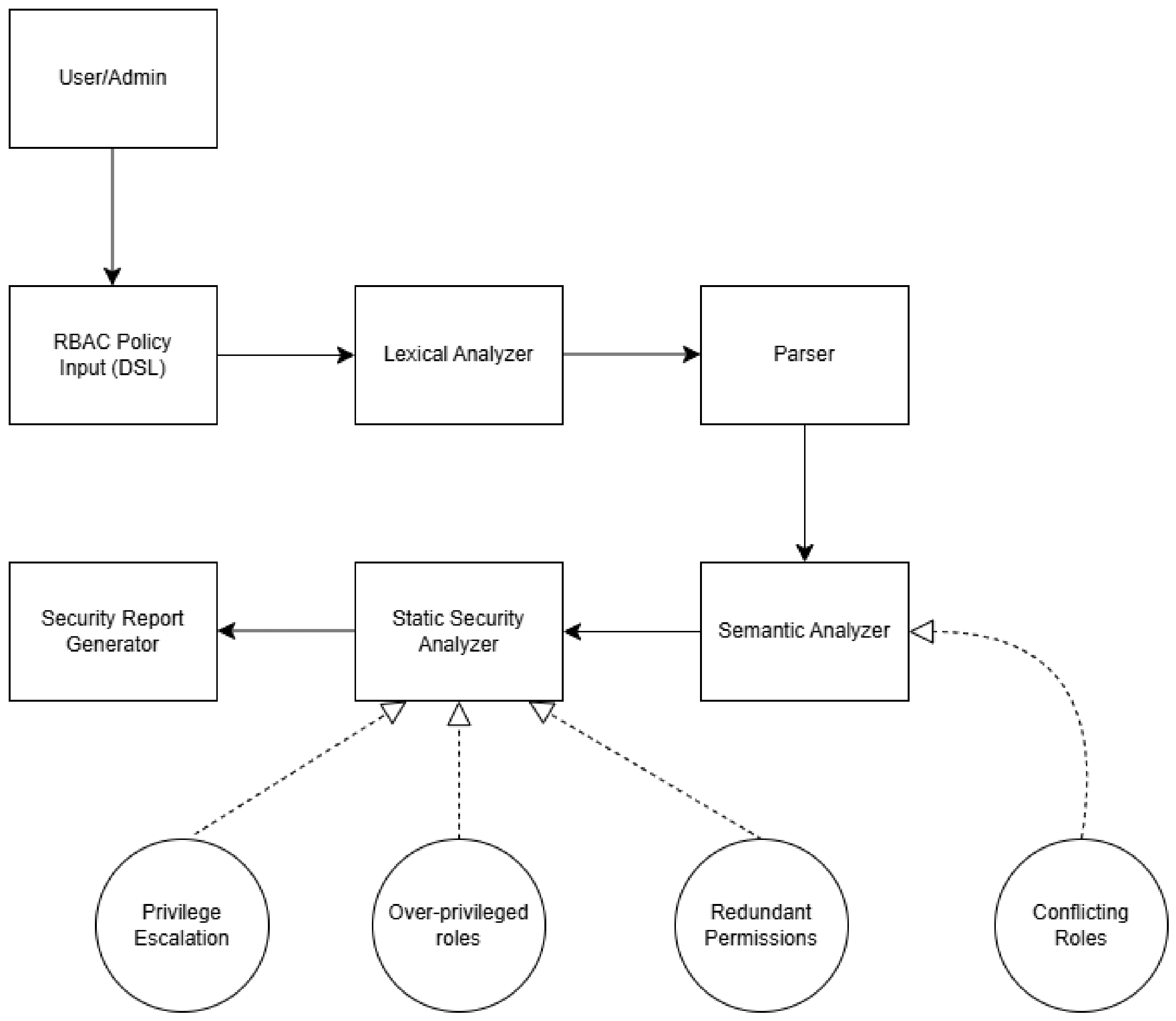
The threat model diagram is included below and referenced throughout the threat analysis section.

(i) Threat Model for RBAC Policy DSL Compiler (Static Analysis):
Illustrates how RBAC policy misconfigurations are detected through a compiler-based static analysis pipeline.
Contains:
- User / Admin (Policy Author)
- RBAC Policy DSL (Input)
- Lexical Analyzer
- Parser
- Semantic Analyzer
- Static Security Analyzer
- Security Report Generator

Threat bubbles connected to Static Security Analyzer**:**

- Conflicting Roles
- Redundant Permissions
- Over-Privileged Roles
- Privilege Escalation

.



(ii) Conceptual RBAC Threat Model

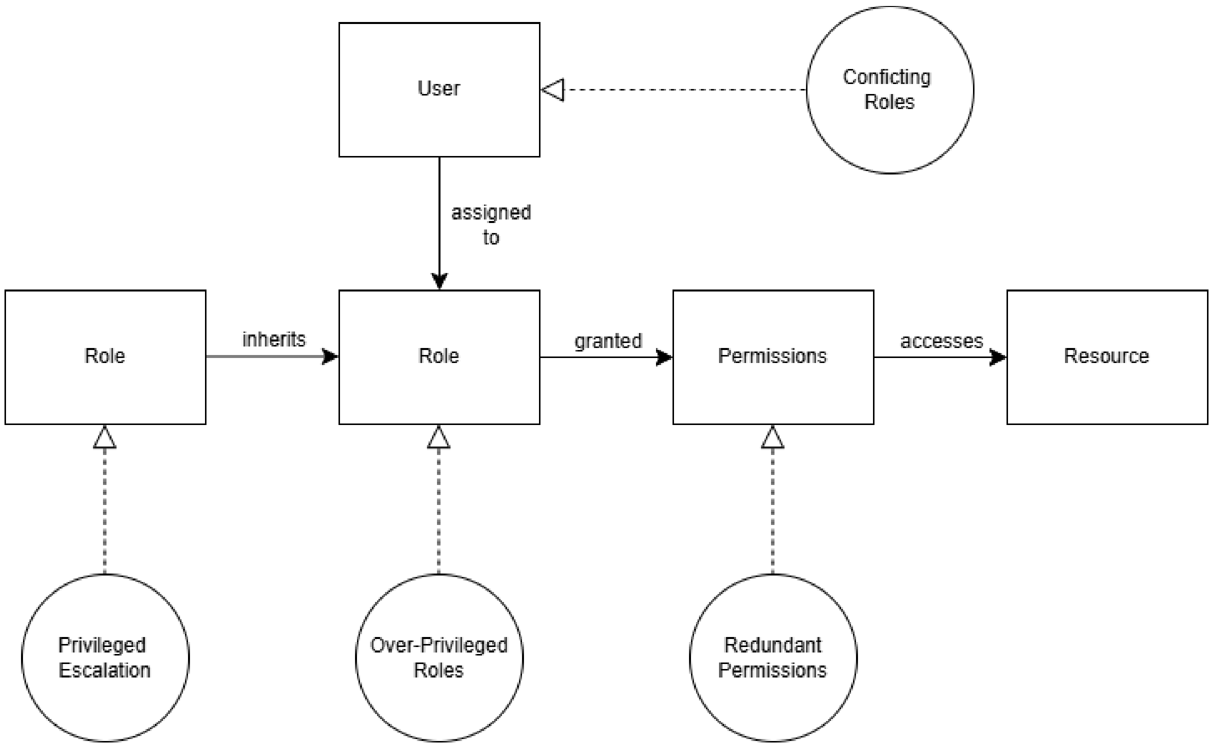
Illustrates RBAC entities and how misconfigurations in role–permission relationships lead to security threats.

Contains:

- User
- Role
- Permission
- Resource
- Role Hierarchy (Role → Role)

Threat bubbles:

- Conflicting Roles
- Redundant Permissions
- Privilege Escalation
- Over-Privileged Roles



RBAC Threat	**Analysis Technique / Component**
Conflicting Roles	Semantic Analyzer / User
Privilege Escalation	Static Security Analyzer / Role Hierarchy Analysis
Redundant Permissions	Static Security Analyzer / Permission Analysis
Over-privileged Roles	Role-Permission Evaluation / Semantic Analyzer

7. Assumptions and Constraints

- **Assumptions**:
  - Policies follow the defined DSL.
  - Users, roles, permissions, and resources are modeled correctly.
  - No external runtime enforcement is included.
- **Constraints**:
  - Analysis is static only, not runtime.
  - Focused on RBAC policies only, not other access control models.
  - Only supports compile-time detection.