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 2 Deliverables

Week 2 – Literature Survey & Gap Identification

1. Introduction

After defining the RBAC problem and security motivation in Week 1, this week focuses on studying existing standards, tools, and research approaches related to RBAC and access control policy analysis. The objective is to understand how RBAC policies are currently specified and verified, identify their limitations, and clearly justify the need for a compiler-based static analysis solution.

2. Study of Relevant Standards and Tools

The objective of this section is to study existing RBAC standards, commonly used RBAC-based IAM systems, and static security analysis techniques in order to understand how RBAC policies are currently specified, enforced, and analyzed. This study helps identify limitations in existing approaches and motivates the need for a compiler-based static analysis solution**.**

(i) NIST RBAC Standard

The NIST RBAC standard defines a formal and widely accepted model for Role-Based Access Control. It specifies core RBAC entities such as users, roles, permissions, role assignments, role hierarchies, and constraints including separation of duty. This standard forms the conceptual foundation for most enterprise and cloud-based access control systems.

Limitations:
The NIST RBAC standard primarily focuses on policy definition and runtime access enforcement. It does not provide mechanisms for compile-time verification of RBAC policies or static detection of security issues such as role conflicts, redundant permissions, or privilege escalation. Consequently, policy misconfigurations may only be discovered after deployment.

(ii) Existing RBAC and IAM Systems

RBAC is widely implemented in modern Identity and Access Management (IAM) systems used in enterprise software, databases, and cloud platforms. These systems group permissions into roles and assign users to roles based on organizational responsibilities. Access control decisions are evaluated dynamically at runtime.

Limitations:
Existing IAM systems emphasize runtime enforcement rather than design-time verification. They offer limited support for structural analysis of RBAC policies and largely depend on manual inspection to detect role conflicts and redundant permissions. This approach does not scale well for large and complex RBAC configurations.

(iii) Static Security and Policy Analysis Techniques

Static security analysis techniques analyze specifications without executing them and are extensively used in compiler design and software security analysis. When applied to RBAC, policies can be modeled as formal specifications to analyze role hierarchies, permission propagation, and user-role relationships at compile time.

Limitations:
Although static analysis is effective in detecting logical inconsistencies early, its application to RBAC policies remains limited. Existing approaches often focus on specific security properties, use complex formal models, and lack clear explanations of detected issues. Additionally, these techniques are rarely integrated into practical RBAC policy authoring or management workflows

From the study of standards and tools, it is observed that:

- Existing RBAC standards focus on specification, not verification
- IAM systems prioritize runtime enforcement over early error detection
- Static analysis techniques are powerful but underutilized for RBAC

This clearly highlights the need for a compiler-based static analysis approach for RBAC policy verification.

3. Survey of Existing Research Work

This section presents a survey of recent research work related to the analysis and verification of Role-Based Access Control (RBAC) policies. The survey focuses on research papers published in reputed venues such as IEEE, ACM, and Springer within the last four to five years. The objective is to understand the techniques used in existing research for RBAC security analysis, identify their strengths, and highlight the limitations that motivate the proposed work.

3.1 Overview of Research Papers Surveyed

The literature survey includes research papers selected from well-established and peer-reviewed publication venues, ensuring both quality and relevance.

- **Publication sources:**
    - IEEE Transactions and Conferences
    - ACM Security and Computer Systems Conferences
    - Springer Journals and Book Series
- **Time period considered:**
    - Papers published during the last **4–5 years**
- **Selection criteria:**
    - Focus on RBAC or access control policy analysis

  - Address security issues such as misconfigurations, conflicts, or privilege escalation
  - Propose static, formal, or analytical techniques rather than only runtime enforcement

These papers represent the current state-of-the-art in RBAC security research.

3.2 Major Research Focus Areas

Based on the survey, existing research work can be broadly classified into the following focus areas.

(i) RBAC Policy Verification

Several research works propose formal verification techniques to validate RBAC policies. These techniques check policy consistency, constraint satisfaction, and violations of separation of duty using logical inference, constraint solving, or formal modeling.

Limitation:
Such verification methods are often complex and difficult to apply to large real-world RBAC systems. They also lack user-friendly mechanisms for policy specification.

(ii) Privilege Escalation Detection

Some studies focus on detecting privilege escalation caused by improper role inheritance and transitive permission propagation. These approaches commonly use graph-based models and reachability analysis to identify unintended access paths.

Limitation:
Privilege escalation detection is frequently performed after policy deployment or as part of audits. The results often lack clear explanations that can help policy designers correct the underlying issue.

(iii) Role Conflict Analysis

Research in this area analyzes conflicts arising from incompatible role assignments and violations of separation of duty constraints. Constraint-based checking is widely used to enforce organizational security rules.

Limitation:
Most approaches rely on predefined conflict rules and do not deeply analyze the interaction between role conflicts and role hierarchies.

(iv) Static Security Analysis

A limited number of studies apply static analysis techniques, inspired by compiler and program analysis research, to analyze RBAC policies without execution. These techniques aim to detect structural inconsistencies and security flaws early.

Limitation:
Static analysis approaches for RBAC are not widely adopted and are rarely implemented as part of a complete compiler-style analysis pipeline.

3.3 Summary of Surveyed Research

From the survey of existing research work, the following observations are made:

- Extensive research exists on RBAC security analysis
- Most approaches emphasize runtime enforcement or post-deployment audits
- RBAC policies are treated mainly as configuration artifacts rather than formal specifications
- Very few approaches support:
  - DSL-level RBAC policy specification
  - Compile-time security verification
  - Explainable security diagnostics

These observations highlight the need for a compiler-based static analysis framework that integrates policy specification, analysis, and reporting in a unified manner.

4. Literature Review Table

Paper / System	**Year**	**Source / Technique**	**Key Contributions**	**Limitations**	**Key Learnings**
A Survey on Empirical Security Analysis of Access Control Systems — Parkinson & Khan	2022	ACM Computing Surveys (survey of access-control analysis techniques) (pure.hud.ac.uk)	Comprehensive review of access-control policy analysis research, including RBAC, ABAC, misconfig detection, analysis techniques	Focuses on empirical analysis; not a verification tool itself	Shows what kinds of analysis have been tried, gaps in practice vs theory; highlights need for scalable and explainable security analysis

Model Checking Access Control Policies — Google Cloud IAM — Gouglidis et al.	2023	ArXiv / model checking	Applies **model checking** to RBAC/IAM policies to verify security properties; demonstrates formal verification on a real platform (arXiv)	Requires formal modelling expertise; not integrated into a DSL/compiler	Tells how formal methods can detect violations early but aren't practical for general use
Automated SELinux RBAC Policy Verification Using SMT — Pahuja et al.	2023	ArXiv / SMT solver techniques	Converts SELinux RBAC policies into SMT formulas to **verify correctness and detect misconfigurations** (arXiv)	Focuses on SELinux policies; not full RBAC DSL verification	Example of how automated theorem proving can check RBAC constraints
A Model-Driven Engineering Approach for Detecting Privilege Escalation — A. Abu Zaid et al.	2022	ArXiv / static & model-driven analysis	Detects direct & indirect **privilege escalation** in permission models (IoT example) (arXiv)	Domain-specific to IoT; not general RBAC DSL	Shows how static approaches help detect escalation but not a general compiler pipeline
Research on Policy-as-Code and Modern RBAC/ABAC — Vakhula et al.	2025	CEUR Workshop Proceedings / policy-as-code	Explores RBAC/ABAC implemented as code (PaC), CI/CD integration, automated validation, compliance checks (CEUR-WS)	Not strictly RBAC static analysis; more on policy automation	Useful to show how treating policies as *code* can improve validation and enforcement

Identifying High-Risk Over-Entitlement in Access Control Policies — Parkinson & Khan	2022	SpringerAccess (fuzzy logic for access control analysis) (Springer Link)	Focuses on **over-entitlement detection** using fuzzy logic	Not a full static verifier for RBAC policies	Good example of nuanced security analysis beyond simple confict detection

Key Learnings from Literature Survey

From the detailed study of RBAC standards, IAM tools, and recent research papers published in IEEE, ACM, and Springer venues, the following key learnings were obtained.

• RBAC is a "mature and widely adopted access control model", with strong standardization through the NIST RBAC framework.

• Existing RBAC standards primarily focus on "policy specification and runtime enforcement", not on early verification of policy correctness.

• Most enterprise IAM systems:

- Enforce access control at runtime
- Lack automated mechanisms for detecting design-time misconfigurations
- Depend heavily on manual audits for security validation

• Research work has explored formal, graph-based, and constraint-based techniques for:

- RBAC policy verification
- Role conflict detection
- Privilege escalation analysis

• Privilege escalation is often caused by:

- Improper role inheritance
- Transitive permission propagation
- Complex role hierarchies that are difficult to analyze manually

• Existing research approaches:

- Are often theoretical or domain-specific
- Require complex formal models
- Are not easily usable by policy designers or system administrators

• Very few research works treat RBAC policies as “formal, analyzable specifications”, similar to programs.

• There is “no unified solution” that simultaneously provides:

- A Domain-Specific Language (DSL) for RBAC policy specification
- Compile-time static security analysis
- Automated detection of conflicts, redundancies, and privilege escalation
- Explainable and human-readable security diagnostics

• These learnings clearly indicate that compiler-based static analysis techniques, widely used in software engineering, can be effectively applied to RBAC policy verification.

• The literature survey strongly motivates the proposed project to develop a compiler-based RBAC policy analysis framework that ensures security by design rather than security after deployment.

5. Comparative Analysis of Existing Approaches

Aspect	**RBAC Standards (e.g., NIST)**	**RBAC / IAM Tools**	**Research-Based Approaches**	**Proposed Compiler-Based Approach**
Policy Representation	Conceptual RBAC model	Configuration rules	Formal or mathematical models	Domain-Specific Language (DSL)
Stage of Analysis	Runtime enforcement	Runtime enforcement	Mostly post-deployment	Compile-time

Static Security Analysis	Not supported	Very limited	Partially supported	Fully supported
Role Conflict Detection	Manual	Limited	Supported	Fully automated
Redundant Permission Detection	Not supported	Rarely supported	Partially supported	Fully supported
Privilege Escalation Detection	Not addressed	Limited	Supported (often post-deployment)	Compile-time detection (direct and indirect)
DSL-Level Policy Verification	Not available	Not available	Rarely available	Fully supported
Explainability of Results	Not applicable	Minimal	Limited	Human-readable explanations
Scalability for Large Policies	Conceptual only	High (for enforcement)	Limited	High
Ease of Policy Authoring	Moderate	Moderate	Low (complex formalisms)	High (structured DSL)

Key Observations from Comparative Analysis

- Existing RBAC standards focus on policy definition, not verification
- IAM tools prioritize runtime enforcement, not early error detection
- Research approaches provide theoretical analysis but lack practical integration
- None of the existing approaches combine:
    - DSL-based policy specification
    - Compiler-style static analysis
    - Explainable security reporting
- The proposed compiler-based approach addresses all these limitations in a unified framework

6. Identification of Research Gaps

Based on the detailed study of RBAC standards, existing IAM systems, and recent research work, several critical research gaps have been identified. These gaps highlight the limitations of current approaches and clearly motivate the need for the proposed compiler-based static analysis framework.

6.1 Lack of DSL-Based RBAC Policy Specification

- Existing RBAC standards and IAM tools:
    - Rely on configuration files or administrative rules
    - Do not provide a formal "Domain-Specific Language (DSL)" for RBAC policy definition
- Absence of a DSL leads to:
    - Ambiguous and error-prone policy specifications
    - Difficulty in performing syntax and structural validation
- Limited or no support for:
    - Early syntax checking
    - Structural consistency validation of RBAC policies

6.2 Limited Compile-Time Security Analysis

- Most existing approaches focus on:
    - Runtime access control enforcement
    - Post-deployment security audits
- Compile-time analysis of RBAC policies is:
    - Rarely supported
    - Not systematically implemented
- As a result:
    - Security misconfigurations remain undetected until deployment
    - Logical errors propagate into production systems

6.3 Poor Explainability of Detected Security Issues

- Existing tools and research approaches often:
    - Report security violations as low-level warnings
    - Provide limited contextual information
- Lack of:
    - Clear explanation of why a violation occurs

    - Guidance for correcting RBAC misconfigurations
- This makes:
    - Debugging complex RBAC policies difficult
    - Manual correction time-consuming and error-prone

6.4 Inadequate Privilege Escalation Detection

- Many existing approaches detect:
    - Only direct privilege escalation scenarios
- Limited support for:
    - Indirect or transitive privilege escalation
    - Multi-level role inheritance analysis
- Graph-based analysis techniques are:
    - Used inconsistently
    - Rarely automated in practical tools

6.5 Fragmented and Non-Integrated Solutions

- Existing research solutions:
    - Address individual RBAC security problems in isolation
    - Lack end-to-end integration
- No unified framework exists that combines:
    - Policy specification
    - Parsing and validation
    - Static security analysis
    - Explainable reporting

7. Problem Gap Statement

The literature survey indicates that although Role-Based Access Control (RBAC) is widely adopted and extensively studied, existing standards, tools, and research approaches do not adequately support early detection of RBAC policy misconfigurations. Most RBAC standards focus on policy definition and runtime enforcement, without providing mechanisms for verifying policy correctness before deployment.

Current RBAC and IAM tools primarily enforce access decisions at runtime and offer limited support for detecting logical errors such as role conflicts, redundant permissions, and unintended privilege escalation in advance. As a result, security vulnerabilities often remain unnoticed until after deployment or during manual audits.

While research-based approaches propose formal and graph-based techniques for RBAC security analysis, they are often complex, lack practical integration, and provide limited explainability for detected issues. Very few approaches treat RBAC policies as analyzable specifications or support compile-time security verification.

Therefore, there exists a clear problem gap for a unified approach that enables DSL-based RBAC policy specification, compiler-style static analysis, and explainable security reporting to detect RBAC misconfigurations before deployment.

8. Justification for Compiler-Based Static Analysis

8.1 Why a Compiler-Based Approach is Suitable

- RBAC policies have structured components:
    - Roles, permissions, users, and inheritance
- These components follow well-defined rules and constraints
- Hence, RBAC policies can be treated as program-like specifications
- Compiler techniques naturally fit this structured analysis

8.2 Benefits of Compile-Time Verification

- Detects security misconfigurations before deployment
- Prevents insecure RBAC policies from entering production
- Provides early feedback during policy design
- Reduces dependence on runtime monitoring and audits

8.3 Role of Static Security Analysis

- Analyzes RBAC policies without execution
- Enables detection of:
    - Role conflicts
    - Redundant permissions
    - Direct and indirect privilege escalation
- Graph-based analysis supports:
    - Role hierarchy modeling

    - Permission propagation analysis using DFS/BFS

8.4 Advantages Over Existing Approaches

- Unlike runtime-only systems:
    - Identifies logical errors at design time
- Unlike isolated research tools:
    - Integrates DSL design, parsing, analysis, and reporting
- Produces human-readable and explainable security diagnostics
- Supports secure-by-design RBAC policy development