Product Requirements Document
Version: 1.0 Date: September 2025 Author: Lawrence Stakeholder: LLM Security Team, working with the LLM Data team
Executive Summary
Integrate with the People Analytics team's org-chart graph database to enable dynamic, role-based privacy decisions within our AI Privacy Firewall, improving the accuracy of Contextual Integrity.
Problem Statement
Our privacy firewall currently relies on static role definitions (e.g., "manager", "employee", "contractor") that fail to capture complex organizational relationships. This leads to:
● False positives: legitimate information sharing between team members is blocked
● False negatives: Inappropriate cross-department data exposure is missed
● Manual maintenance: 40 hours/month updating role mappings
Proposed Solution
Leverage the existing Neo4j org-chart graph database maintained by People Analytics to dynamically determine:
● Reporting relationships (direct, skip-level, dotted-line)
● Department boundaries and data domains
● Project team memberships
● Role-based access permissions
● Temporal role changes (acting roles, leave coverage)
Key Use Cases
Use Case
Current State
Future State
Manager accessing team member's PTO
Static rule checking
Graph traversal validates reporting relationship
Cross-functional project data sharing
Manual whitelist maintenance
Auto-detect shared project membership
Contractor data access
Binary yes/no permissions
Scoped access based on contract terms in graph
Acting role permissions
Manual temporary rules
Time-bounded permissions from graph metadata
Integration Requirements
Required Graph Queries
cypher
```
// 1. Check direct reporting relationship
MATCH (employee:Person)-[:REPORTS_TO]->(manager:Person)
WHERE employee.id = {employeeId} AND manager.id = {managerId}
RETURN exists((employee)-[:REPORTS_TO]->(manager))
// 2. Check same department
MATCH (p1:Person)-[:BELONGS_TO]->(d:Department)<-[:BELONGS_TO]-(p2:Person)
WHERE p1.id = {senderId} AND p2.id = {recipientId}
RETURN d.name, d.data_classification
```

```
// 3. Check project membership
MATCH (p1:Person)-[:MEMBER_OF]->(proj:Project)<-[:MEMBER_OF]-(p2:Person)
WHERE p1.id = {senderId} AND p2.id = {recipientId}
AND proj.status = 'active'
RETURN proj.name, proj.data_scope
```