AURORA: 클라우드 네이티브 환경을 위한 IAM 기반 리소스 접근 제어 자동화 프레임워크

김민경* ,이승수**

*,**인천대학교 (학부생, 교수)

AURORA: Automated Unified Resource-Oriented rule Alignment for Cloud-Native Environments

MinKyung Kim*, Seungsoo Lee**

*,**Incheon National University(Undergraduate student, Professor)

요 약

클라우드 네이티브 환경에서 워크로드가 클라우드 리소스에 접근하려면 IAM 권한과 VPC firewall rule이 모두 허용되어야 한다. 그러나 두 정책은 독립적으로 관리되며, 리소스별 네트워크 요구사항과 지속적인 환경 변화로 인해 일관성을 유지하기 어렵다. 본 논문에서는 이를 해결하기 위한 IAM 기반 리소스 접근 제어 자동화 프레임워크 AURORA를 제안한다. AURORA는 GCP 환경에서 IAM 권한과 LLM을 활용해 접근 가능한 리소스를 식별하고, 필요한 VPC firewall rule을 자동 구성한다. 또한 동적 변화에 따라 정책을 재구성하고 적용 순서를 보장한다. 평가 결과, LLM 기반 리소스 접근 판별에서 높은 성능을 보였고, Pod 이동 시 firewall rule을 자동 재구성해 정상 접근을 유지하였다. 또한, Pod 삭제 처리 시간은 약 68% 증가했으며, 이는 기존의 수동 분석과 정책 구성을 자동화하는 과정에서 발생한 추가 처리 시간이다.

I. Introduction

클라우드 환경에서 워크로드가 Cloud Service Provider(CSP)가 제공하는 리소스에 접근하려면 Identity and Access Management(IAM)과 Cloud firewall을 통한 제어가 필요하다. IAM은 워크로드가 접근 가능한 리소스에 대한 권한을 결정하고, Cloud firewall은 해당 리소스로의 네트워크 접근을 제어한다. 따라서 정상적인 클라우드 리소스 접근을 위해서는 두 정책 계층이 모두 적절히 구성되어야 한다.

그러나 IAM과 Cloud firewall은 독립적으로 관리되며, 클라우드 리소스마다 필요한 네트워크 요구사항도 다르다. 또한 클라우드 네이티브 환경에서는 리소스가 지속적으로 변화하므로, 이러한 특성과 변화를 firewall rule에 적절히 반영하지 않으면 필요한 접근이 차단되거나 불필요한 접근이 허용될 수 있다.

본 연구는 이러한 한계를 해결하기 위해 IAM 기반 리소스 접근 제어 자동화 프레임워크 AURORA를 제안한다. 본 프레임워크는 Large Language Model(LLM)을 활용하여 Google Cloud Platform(GCP) 환경에서 접근 가능한 클라우드 리소스를 식별하고, 리소스별 네트워크 요구사항을 Virtual Private Cloud(VPC) firewall rule으로 자동 변환한다. 또한 클라우드 네이티브 환경의 동적 변화를 반영하여 IAM과 VPC firewall 간 정책 일관성을 효과적으로 유지한다.

본 논문의 주요 기여는 다음과 같다.

- **IAM 기반 firewall rule 생성 자동화:** 워크로드의 IAM 권한으로부터 접근 가능한 Cloud Resource와 네트워크 요구사항을 식별하여 VPC firewall rule을 자동 구성한다.
- **정책 적용 순서 보장:** 동적 변화 시 IAM과 VPC firewall rule의 적용 순서를 제어하여 일시적인 접근 차단과 불필요한 네트워크 허용으로 인한 보안 공백을 최소화한다.
- **LLM 기반 리소스 식별:** IAM 정책만으로 확정하기 어려운 클라우드 리소스의 접근 가능 여부를 LLM을 활용해 판단하며, 평가에서 높은 판별 성능을 보였다.

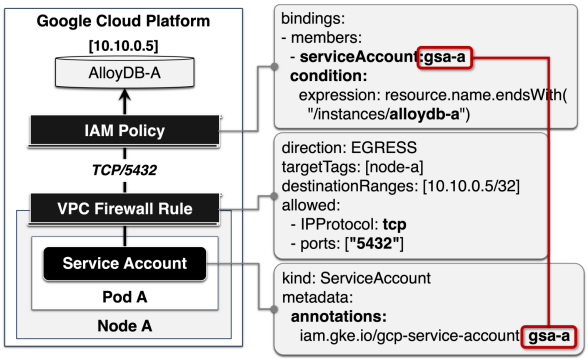


Fig 1. Google Cloud의 클라우드 리소스 접근 구조

II. Background

클라우드 네이티브 워크로드는 다양한 클라우드 리소스를 활용한다. GCP에서는 이러한 리소스 접근을 IAM과 VPC firewall을 통해 각각 접근 권한 및 네트워크 수준에서 제어한다. Fig 1은 이와 같은 클라우드 리소스 접근 구조와 관련 정책의 예시를 나타낸다.

2.1. Cloud Resource 접근 구조

Google Kubernetes Engine(GKE)에서 Pod는 Service Account의 annotation을 통해 Google Service Account(GSA)의 권한을 사용하며, 이에 따라 접근 가능한 리소스가 결정된다. 한편 실제 통신은 대상 리소스의 IP, protocol, port로 구성된 endpoint를 통해 수행된다. 따라서 네트워크 접근 또한 정상적인 리소스 접근을 위해서 허용되어야 한다.

2.2. IAM과 Cloud firewall 기반 접근제어

IAM은 접근 주체와 권한의 binding을 통해 클라우드 리소스 접근을 관리하며, condition 필드를 통해 특정 리소스나 요청 조건에서만 해당 권한이 적용되도록 제한할 수 있다. 반면 VPC firewall은 IP, protocol, port를 기준으로 네트워크 통신을 제어한다. 예를 들어 Fig 1에서는 gsa-a의 IAM 권한이 alloydb-a에만 적용되도록 condition을 설정하고, 해당 리소스의 endpoint인 10.10.0.5:5432/TCP에 대해서만 VPC firewall의 네트워크 접근을 허용한다.

III. Problem Statement

3.1. IAM과 VPC firewall의 불일치

GCP의 기본적으로 생성되는 VPC firewall은 모든 트래픽을 허용하므로 불필요한 네트워크 접근이 가능하다[2]. 반면 Allow-List 방식은 필요한 통신만 허용해 공격 표면을 줄일 수 있지만, IAM이 허용한 리소스에 필요한 firewall rule이 누락되면 정상적인 접근까지 차단될 수 있다. 따라서 두 정책의 허용 범위를 일관되게 유지하는 것이 중요하다.

3.2. 리소스별 네트워크 요구사항의 이질성

한편 클라우드 리소스는 종류와 구성에 따라 필요한 IP, protocol, port와 같은 네트워크 스펙이 서로 다르다. 따라서 IAM으로 접근 가능한 리소스를 식별한 후, 각 endpoint 정보를 파악하여 적절한 firewall rule으로 변환해야 한다. 요구사항이 누락되거나 잘못 반영되면 정상적인 리소스 접근이 제한될 수 있다.

3.3. 클라우드 네이티브 환경의 동적 변화

또한 클라우드 네이티브 환경에서는 Pod의 생성, 삭제 및 스케줄링, IAM binding 변경, 클라우드 리소스와 endpoint 변화가 지속적으로 발생한다. 이에 따라 접근 가능한 리소스와 node별 firewall rule도 함께 변경되어야한다. 이를 적절히 반영하지 않으면 필요한 접근이 차단되거나 불필요한 네트워크 접근이 허용될 수 있다.

IV. Design

4.1. Threat Model

본 연구는 워크로드가 Allow-List 기반 firewall 환경에서 운영되며, 모든 IAM binding 변경은 관리 노드의 전용 Command Line Interface (CLI)를 통해 수행된다고 가정한다. 또한 Pod가 Service Account를 통해 GSA의 권한을 위임받아 사용하는 GSA Impersonation 환경을 대상으로 한다. 클라우드 리소스는 endpoint를 명확히 식별할 수 있는 리소스로 범위를 제한한다.

4.2. Architecture Overview

AURORA의 아키텍처는 Fig 2와 같다. Initialization 단계에서는 Google Cloud 문서를 기반으로 리소스별 네트워크 정보를 수집하여 Endpoint DB를 구성한다. Runtime 단계에서 Pod 및 IAM binding 변화는 AURORA Scheduler와 Controller가 감지하고, Cloud Resource 변화는 별도 경로로 감지한다. Pod는 GSA Annotation 유무에 따라 AURORA Scheduler 또는 Default Scheduler에서 처리된다. 이후 AURORA는 관련 GSA를 식별하고, Google Cloud의 Policy Analyzer[1]를 이용해 IAM condition을 평가하여 접근 가능한 Resource Set을 구성한다. 이때 condition을 확정적으로 판단하지 못해 Conditional로 반환된 리소스는 LLM으로 추가 평가한다. 최종 Resource Set은 Endpoint Resolver가 네트워크 요구사항을 조회하여 node별 목표 상태를 AURORA CRD에 반영하고, Converter가 현재 상태와 비교하여 필요한 VPC firewall rule을 재구성한다.

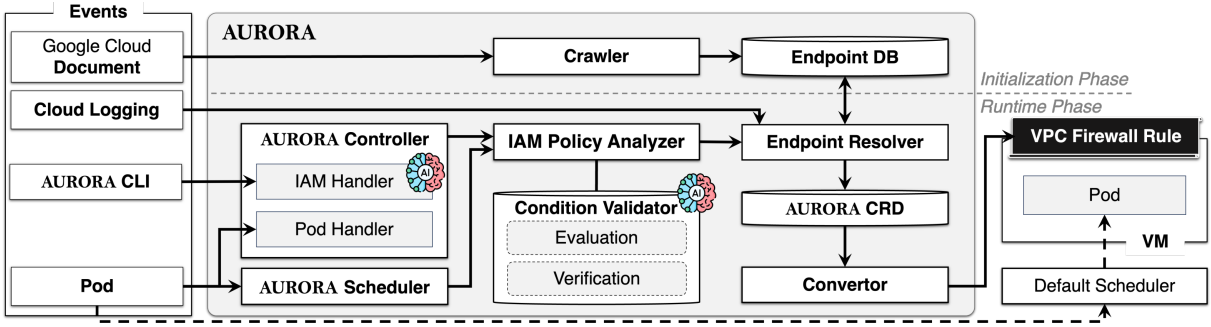


Fig 2. Overall Architecture

V. Detail

5.1. IAM 기반 firewall rule 생성

AURORA는 Policy Analyzer의 분석 결과를 기반으로 GSA별 Resource Set을 구성한다. 접근 가능 여부가 확정된 리소스는 바로 포함하고, Conditional로 반환된 리소스는 Fig 3과 같이 condition 평가에 필요한 context를 수집해 LLM으로 추가 검증한다. LLM은 전체 condition을 직접 판단하는 DIRECT와 개별 절을 평가하는 CLAUSE를 수행한다. 두 결과가 일치하면 해당 결과를 포함하고, 불일치하면 COUNTEREXAMPLE을 추가로 수행한 뒤 세 결과의 다수결로 접근 가능 여부를 결정한다. TRUE로 판단된 리소스를 최종 Resource Set에 포함한다.

이후 각 리소스에 대해 Endpoint DB를 조회해 네트워크 요구사항을 식별하고, 이를 기반으로 node별 목표 상태를 구성하여 AURORA CRD에 반영한다. Converter는 목표 상태와 현재 firewall 상태를 비교해 수정할 규칙을 도출하고, 규칙 정보의 hash 값으로 고유한 규칙명을 생성한다. 이후 변경사항을 VPC firewall에 반영하여 실제 정책을 목표 상태와 일치시킨다.

5.2. 네트워크 접근 요구사항 구성

Google Cloud 문서를 크롤링하여 리소스별 port, protocol 정보를 분석하여 Endpoint DB를 구축한다. 이후 리소스가 식별되면 Endpoint Resolver가 실제 해당 리소스의 endpoint IP를 확인하고, 이를 IP를 포함한 형태의 네트워크 요구사항으로 변환한다.

5.3. 동적 변화에 따른 firewall 재구성

클라우드 네이티브 환경에서의 동적 변화를 event로 감지하고, 변경 유형별로 VPC firewall rule의 적용 순서를 보장한다. 예를 들어 Pod 생성 전 필요한 firewall rule을 먼저 적용하고, Pod 삭제 후 불필요한 정책을 제거한다. 또한 IAM binding 변경에서도 전환 과정에서 발생할 수 있는 보안 공백을 최소화한다.

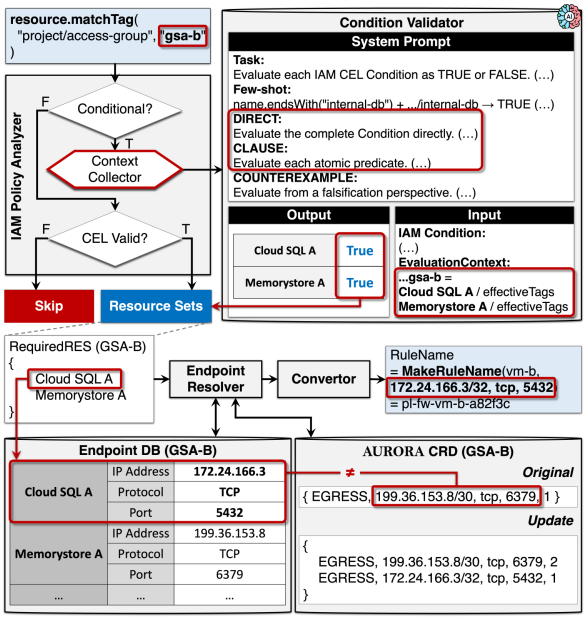


Fig 3. LLM 기반 VPC firewall 생성 과정

VI.Evaluation

6.1. Implementation

평가는 Google Online Boutique 기반의 1개 master node와 2개 GKE worker node 환경에서 수행했으며, AURORA는 Go로 구현하였다. 시나리오별로 GSA와 IAM binding 구성을 달리하고, 클라우드 리소스로는 GitHub에서 제공하는 AlloyDB cluster와 Memorystore for Redis cluster를 사용하였다. LLM 기반 Conditional 평가에는 gpt-5.6-terra를 사용했으며, 28개의 GSA와 6개의 Cloud SQL 리소스를 별도로 구성해 복합 IAM condition을 적용하였다.

6.1. 변환 정확성

IAM-to-SG 변환. GSA에 AlloyDB 접근을 위한 IAM binding을 추가한 결과, Fig 4와 같이 AURORA가 없는 환경에서는 firewall rule이 반영되지 않아 접근에 실패하였고(A), 반면, AURORA는 필요한 규칙이 반영되어 정상적으로 Query 결과를 반환하였다(B).

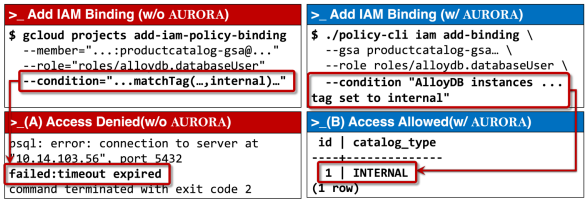


Fig 4. IAM binding 추가 후 리소스 접근 결과 비교

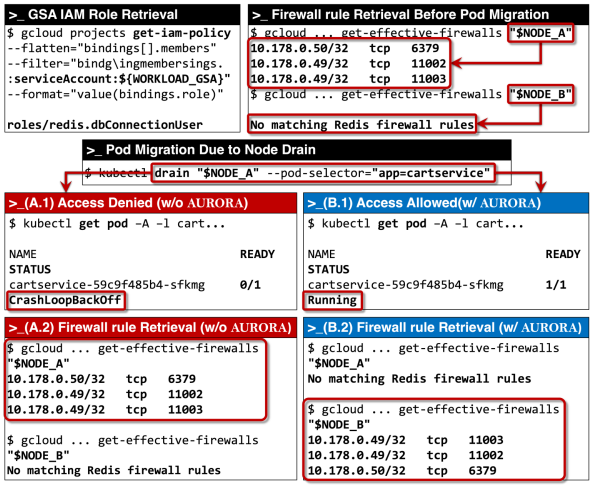


Fig 5. Pod 스케줄링 후 리소스 접근 결과 비교

Pod 이동에 따른 firewall 재구성. Pod 이동 시 firewall rule이 올바르게 재구성되는지 평가하기 위해, Redis 접근이 필요한 Pod가 실행 중인 node A를 drain하여 node B로 재스케줄링하였다. Fig 5와 같이 AURORA가 없는 경우에는 이동 후 firewall rule이 반영되지 않아 (A.2) Redis 접근에 실패하였다(A.1). 반면 AURORA는 node B에 필요한 규칙을 추가하고 Node A의 불필요한 규칙을 제거하여(B.2) 정상적인 접근을 유지하였다(B.1).

6.2. LLM 성능

Conditional binding 처리에서 LLM 적용 효과를 평가하였다. Policy Analyzer가 Conditional로 반환한 60개의 GSA-Resource 조합에 대해 ground truth를 구성하고, Google Cloud의 기본 Policy Analyzer를 사용하는 baseline과 LLM 기반 condition 평가를 적용한 AURORA의 결과를 비교하였다.

Table 1과 같이, baseline에서는 Policy Analyzer가 Conditional로 반환한 리소스를 모두 접근 가능한 것으로 포함하여 37개의 오탐이 발생한 반면, AURORA는 이를 모두 제거하여 Precision, Recall, F1-score 및 Accuracy에서 100%를 기록하였다. 이를 통해 Conditional로 반환된 리소스의 condition 충족 여부에 대한 식별 정확도가 향상됨을 확인하였다.

	w/o AURORA	w/ AURORA
TP/TN/FP/FN	23 / 0 / 37 / 0	23 / 37 / 0 / 0
Accuracy(%)	38.33	100
Precision(%)	38.33	100
Recall(%)	100	100
F1(%)	55.42	100

Table 1. Conditional binding 처리 성능 비교

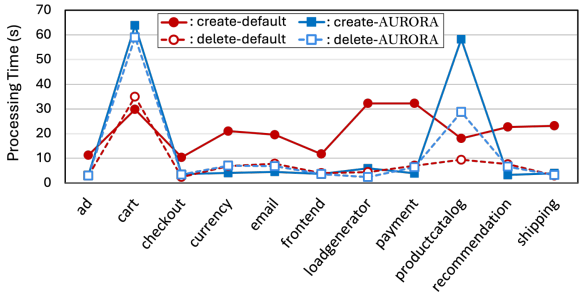


Fig 6. Pod별 생성 및 삭제 처리 시간 비교

6.3. overhead

Pod 생성 및 삭제의 처리 비용을 평가하고자 default scheduler와 AURORA의 처리 시간을 비교하였다. Fig 6과 같이 리소스 접근 Pod 삭제 시 AURORA는 baseline 대비 약 68%의 오버헤드를 보였다. 이는 firewall 요구사항이 사전에 주어진 Baseline과 달리, AURORA가 G-SA 분석과 네트워크 요구사항 도출을 자동화하기 때문이다. 즉, 해당 오버헤드는 관리자의 사전 분석을 자동화하며 발생한 처리 비용이다.

VII. Conclusion

본 논문에서는 IAM을 기반으로 VPC firewall rule을 자동 구성하고, 동적 변화에 따른 정책 재구성 순서를 보장하는 방법을 제안하였다. 또한, 실험을 통해 Conditional binding 처리의 정확성 및 firewall 재구성 효과를 검증하였다.

Acknowledgement

이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. RS-2025-16069415).

[참고문헌]

[1]Google Cloud. 2026. Analyze allow policies. Google Cloud Documentation. https://docs.cloud.google.com/policy-intelligence/docs/analyze-iam-policies

[2]Google Cloud. 2026. Virtual Private Cloud (VPC) overview. Google Cloud Documentation. https://docs.cloud.google.com/vpc/docs/overview