서버리스 환경에서의 IAM 보안 메커니즘 심층 분석

신창희[1], 이승수[2]
[1]인천대학교 컴퓨터공학과 석사과정
[2]인천대학교 컴퓨터공학부 부교수
changhee9149@inu.ac.kr, seungsoo@inu.ac.kr

Deep Dive into IAM Security Mechanisms in Serverless Environments

Changhee Shin[1], Seungsoo Lee[2]
[1]Dept. of Computer Science and Engineering, Incheon National University
[2]Dept. of Computer Science and Engineering, Incheon National University

요 약

IAM은 서버리스 컴퓨팅 환경에서 보안을 구현하는 핵심 메커니즘이다. 서버리스 함수의 동적 실행 특성과 짧은 수명으로 인해 권한 관리 오류는 주요 보안 위협으로 직결된다. 이를 해결하기 위해 각 클라우드 제공업체는 상이한 정책 모델, 권한 단위, 실행 자격 증명, 보안 도구를 제공한다. 이러한 차이는 보안과 운영 측면에서 서로 다른 강점과 한계를 드러내며, 특히 최소 권한 원칙 적용과 함수 단위 격리 구현에서 중요한 영향을 미친다. 본 논문은 주요 클라우드 벤더의 IAM을 네 가지 측면에서 비교 분석하고, 실제 공격 사례를 통해 IAM의 방어 가능성과 한계를 평가한다. 또한 최근 제안된 보안 기법을 검토하여, 서버리스 환경에 적합한 차세대 IAM 프레임워크 설계 방향을 제시한다.

1. 서론

클라우드 컴퓨팅의 급속한 확산과 함께 AWS, Google Cloud, Microsoft Azure, Alibaba Cloud 등 주요 클라우드 서비스 제공자가 보편적으로 활용되고 있으며, Identity and Access Management(IAM)는 이러한 클라우드 보안의 핵심 요소로 부상하고 있다. 특히 서버리스 컴퓨팅에서는 수 초 단위로 생성되고 종료되는 특성을 가지므로, 권한 부여와 회수 과정이 빈번히 발생한다. 이로 인해 단일 설정 오류라도 전체 서비스에 광범위한 영향을 미칠 수 있으며, IAM의 정확성과 일관성이 더욱 중요하다. 따라서 서버리스 환경에서는 권한 관리의 작은 실수도 즉시 보안 취약점으로 이어질 수 있다. 실제로, 최근 보안 동향에서는 플랫폼 자체의 결함보다는 잘못된 IAM 설정이나 과도한 권한 부여와 같은 관리적 오류가 주요 침해 사고의 원인으로 지적되고 있다.

하지만, 보안 관점에서 볼 때, 각 클라우드 서비스 제공자마다 서로 다른 IAM 모델과 정책 구조를 채택하고 있으며, IAM은 시스템 전체의 모든 위협을 단독으로 포괄할 수 없기 때문에, 보안 관리에 상당한 어려움이 존재한다. 예를 들어, AWS는 JSON 기반 정책을 통한 세밀한 권한 제어를, Google Cloud는 계층적 바인딩을 통한 상속 기반 권한 관리를, Azure는 RBAC 기반의 역할 중심 접근 제어를, Alibaba Cloud는 AWS 유사 구조의 RAM을 각각 채택하고 있다. 이러한 이질적인 IAM 시스템은 관리자에게 상당한 학습 비용과 관리 복잡성을 야기하며, 잘못된 권한 설정으로 인한 보안 사고의 위험을 증가시킨다. 특히 서버리스 환경에서는 동적 실행 특성으로 인해 더욱 증대한다.

클라우드 관리자의 관점에서 이러한 IAM 시스템을 효과적으로 활용하는 데는 다음과 같은 구체적인 어려움이 있다. 첫째, 관리자는 종종 각 서버리스 환경을 위한 클라우드별 IAM 정책 모델의 근본적인 차이점과 보안 특성에 대한 이해가 부족하다. 둘째, 관리자가 서버리스 실행 환경에서 발생할 수 있는 IAM 취약점에 대한 종합적인 인사이트가 부족한 경우가 많다. 셋째, 각 플랫폼이 제공하는 보안 도구들은 복잡한 기능과 제한사항을 가지고 있어 전문 지식 없이는 효과적인 활용이 어렵다.

따라서 본 논문에서는 다음과 같은 관점에서 주요 클라우드 IAM 시스템을 분석한다:

- 각 클라우드의 IAM 구성 요소는 서버리스 보안 관점에서 어떤 특성과 차이를 보이는가?
- 각 플랫폼 IAM의 보안 한계가 실제 서버리스 실행 환경에서 어떻게 드러나는가?
- 각 벤더가 제공하는 보안 도구의 한계는 무엇이며, 이를 보완하기 위한 개선 방향은 무엇인가?

<표 1> 주요 클라우드 플랫폼의 IAM 구성요소 비교

	AWS	Google Cloud	Azure	Alibaba
정책 모델	JSON, Deny-first	Hierarchical Binding	RBAC(Built-in/Custom)	JSON, Deny-first
권한 단위	Execution Role	Service Account / Workload Identity	Managed Identity	RAM Role
실행 자격 증명	AssumeRole API	Metadata Server	Metadata Server	STS API
벤더별 보안 도구	Access Analyzer, Cloud Trail	Recommender, Security Command Center	PIM, Conditional Access	STS, ActionTrail

2. IAM 보안 기능 분석

본 섹션에서는 주요 클라우드 서비스 제공업체의 IAM 기능을 서버리스 보안 관점에서 비교한다. 모든 벤더가 IAM을 통해 서버리스 함수의 접근 제어를 수행하지만, 서로 다른 특성을 보인다. 이는 최소권한 원칙 준수, 함수 단위 권한 격리 수준, 그리고 실행 환경에서의 보안 가시성 확보에 직접적인 영향을 미친다. 따라서 본 섹션에서는 (i) 정책 모델, (ii) 권한 단위, (iii) 실행 자격 증명, (iv) 보안 도구를 중심으로 각 벤더를 분석하며, 기본 구조 비교 결과는 표 1에, 서버리스 보안 측면에서의 종합적 평가 결과는 표 2에 요약한다.

2.1 서버리스 IAM의 정책 모델

서버리스 환경의 IAM 정책 모델은 크게 직접 정책(Direct Policy Approach)과 계층적 권한 상속(Hierarchical Permission Inheritance)으로 구분된다. 두 모델은 모두 최소 권한 원칙을 지향하나, 정책의 정의와 적용 방식에서 근본적 차이를 보인다. 그림 1은 두 모델의 구조적 차이를 시각화한 것으로, 직접 정책은 개별 함수에 정책을 바로 부여하고, 계층적 상속은 상위 수준에서 정의된 권한이 하위 리소스로 전파되는 과정을 보여준다.

직접 정책 방식은 JSON 등 정책 문서에 권한 조건을 구체적으로 명시하는 구조로 AWS와 Alibaba Cloud가 대표적이다. 이 방식은 권한 결정을 단일 문서에서 추적할 수 있어 투명성과 예측성이 높다. 특히 AWS는 다양한 조건 연산자를 제공해 시간, 위치, MFA 등 실행 컨텍스트에 따른 세밀한 제약을 지원한다. 반면 Alibaba는 조건 연산자와 고급 제약 기능이 제한적이어서 복잡한 서버리스 시나리오에서 세밀한 제어에 제약이 있다.

계층적 권한 상속 방식은 조직, 폴더, 프로젝트 등 상위 계층에 정의된 바인딩이 하위 리소스로 전파되는 구조로, Google Cloud와 Azure가 대표적이다. Google Cloud의 계층적 바인딩은 대규모 환경에서 관리 일관성을 제공하지만, 상속된 불필요 권한을 하위에서 제거하기 어려워 함수 단위의 최소 권한 구현에 제약이 따른다. 반면 Azure의 RBAC는 계층별로 독립적인 역할 할당이 가능해 함수 단위 권한 격리에 더 유리하다.

정책 복잡성 관점에서는 직접 정책 방식이 단일 문서 기반으로 분석과 감사가 용이한 반면, 계층적 모델은 여러 수준의 정책 병합을 거쳐 최종 권한이 결정되므로 디버깅과 취약점 분석의 난이도가 상승한다. 실무적 관점에서 보면 직접 정책은 정책 오류를 조기에 포착하기 유리하나 관리 대상이 증가할수록 운영 부담이 커진다. 반대로 계층적 상속은 관리 효율성을 제공하지만 상속으로 인한 권한 과잉이 발생하기 쉽다.

2.2 서버리스 함수의 권한 단위 구조

서버리스 함수의 권한 단위 구조는 공통적으로 임시 자격 증명(Temporary Credential) 기반 접근을 지원하지만, 권한 할당 방식과 권한 조합의 유연성에서는 클라우드별로 상이한 특성을 보인다.

Azure는 상대적으로 유연한 권한 구조를 제공한다. System-assigned와 User-assigned Managed Identity를 각각 또는 조합하여 활용할 수 있어, 함수별 고유 권한 설정과 여러 함수 간 권한 공유를 선택적으로 구현할 수 있다. 특히 User-assigned Identity는 여러 함수에서 동일한 권한 세트를 재사용할 수 있어 권한 관리의 효율성을 높인다.

반면 AWS, Google Cloud, Alibaba Cloud는 함수당 단일 권한 엔티티 구조를 채택한다. AWS는 하나의 Execution Role을 할당받아 해당 역할에 연결된 모든 정책이 함수의 권한을 결정한다. Google Cloud는 기본 Compute Service Account 또는 커스텀 Service Account를 사용하며, 함수당 하나의 Service Account만 할당 가능하다. Alibaba Cloud의 RAM Role 역시 함수당 하나의 역할만 부여된다.

단일 권한 엔티티 구조는 권한 추적과 감사 측면에서 명확성을 제공한다. 각 함수가 하나의 권한 주체만을 가지므로 권한 흐름을 파악하기 용이하다. 그러나 기능별 권한을 세밀하게 분리하기는 어렵다. 예를 들어 데이터베이스 접근과 외부 API 호출이 모두 필요한 함수는 두 권한을 하나의 역할로 묶어 부여해야 하므로, 코드 취약점이 발생할 경우 공격자가 수행할 수 있는 행동 범위가 넓어지게 되고, 피해 범위도 증가하게 된다.

<표 2> 서버리스 보안 관점에서 IAM 효과성 비교
(O: 높음/양호, △: 중간, X: 낮음/미흡)

	AWS	Google Cloud	Azure	Alibaba
권한 과잉 위험도	o	o	△	o
취약점 영향 범위	o	o	△	o
실시간 보안 제어	x	x	x	x
서버리스 특화 지원	x	x	x	x

결과적으로, Azure의 다중 Identity 모델은 권한 재사용성과 관리 편의성 측면에서 장점을 제공하지만, 모든 주요 벤더의 단일 엔티티 모델은 단순한 시나리오에서는 관리가 용이하지만 권한 과잉을 구조적으로 유발한다. 이는 서버리스 보안에서 IAM 설정이 단순 편의성을 넘어 최소 권한 구현의 핵심 제약 요소임을 보여준다.

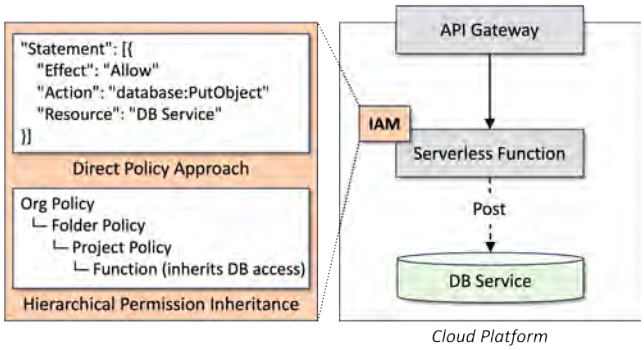
2.3 서버리스 함수의 실행 자격 증명

서버리스 함수 실행 시 자격 증명은 자동으로 부여되며, 별도의 인증 로직을 구현하지 않아도 되는 점에서 모든 플랫폼은 공통적인 편의성을 제공한다. 또한 토큰의 생성, 갱신, 폐기가 모두 자동화되어 운영 부담을 최소화한다는 점에서도 유사하다.

그러나 자격 증명 제공 방식에서는 차이가 존재한다. AWS는 Execution Role을 환경 변수로 주입하여 SDK가 이를 자동 인식하도록 하며, 이 방식은 표준적이고 투명하다. Google Cloud와 Azure는 Metadata Server를 통해 토큰을 발급하며, SDK가 이를 자동화하여 사실상 투명성을 제공한다. 필요 시 직접 HTTP 요청으로 제어할 수 있다는 점에서 유연성이 높다. Alibaba Cloud는 STS API 기반의 임시 자격 증명을 제공하며 서버리스 환경에서는 자동화되어 있으나, 세밀한 조건부 제어 기능은 상대적으로 제한적이다.

보안성 측면에서도 차이가 있다. Azure의 Managed Identity와 Google Cloud의 Workload Identity는 완전한 키리스 환경을 제공하여 자격 증명 노출 위험을 줄인다. AWS 역시 임시 자격 증명 방식을 사용해 유사한 보안을 제공하지만, Role 설정과 위임 구조의 복잡성으로 인해 구성 오류 위험이 존재한다. Alibaba Cloud는 기본적인 임시 자격 증명을 지원하나 고급 제어 기능은 부족하다.

결과적으로, 모든 플랫폼이 자동화된 자격 증명을 통해 기본적인 편의성과 보안성을 확보하고 있다. AWS는 표준적이고 투명한 제공 방식에서 강점을, Azure와 Google Cloud는 키리스 환경 제공에서 보안상의 우위를, Alibaba Cloud는 기본 기능 제공에 머무르는 특징을 가진다. 이는 함수의 실행 자격 증명이 제공 여부가 아니라 투명성, 유연성, 보안성의 균형 속에서 평가되어야 함을 시사한다.



<그림 1> 서버리스 함수에 적용되는 IAM 정책 모델 비교

2.4 벤더별 보안 도구와 한계

클라우드 벤더들은 IAM 정책 검증과 권한 최적화를 지원하기 위해 다양한 보안 도구를 제공한다. AWS는 Access Analyzer와 CloudTrail, Google Cloud는 Recommender와 Security Command Center, Azure는 Privileged Identity Management(PIM)과 Conditional Access, Alibaba Cloud는 STS와 ActionTrail을 중심으로 구성되어 있다. 이러한 도구들은 정책의 외부 노출 가능성 분석, 최소 권한 권장, API 호출 로깅 등 전통적 환경에서는 효과적이다.

그러나 서버리스 환경에서는 이러한 도구들의 효용성이 제한적이다. 첫째, 모든 플랫폼의 도구들이 정적 정책 분석 또는 사후 로그 기반 감사에 의존하기 때문에, 실행 시간이 수 초에 불과한 함수의 실시간 보안 상태를 포착하지 못한다. 둘째, 권한 추천 도구들은 계정 단위의 활동 패턴을 기반으로 작동하므로, 개별 함수 단위의 동적 권한 사용을 제대로 반영하지 못한다. 셋째, 서버리스 특화 메트릭과 알림 체계가 부족해, 함수 수준에서의 세밀한 가시성 확보가 어렵다.

결과적으로, 네이티브 보안 도구들은 서버리스 보안의 기초적 역할(정책 검증, 권한 축소, 로그 수집)은 수행할 수 있으나, 함수 실행 시점의 동적 권한 제어와 실시간 위협 탐지에는 직접적인 도움을 주지 못한다. 따라서 이러한 한계를 보완하기 위해, 서드파티 솔루션이나 서버리스 특화 보안 연구의 필요성이 불가피하다.

3. 서버리스 공격 사례와 IAM 효과성 분석

최근 공개된 서버리스 공격 사례들은 IAM의 방어 가능 영역과 한계를 드러낸다. 과도한 권한 부여와 함수 코드 취약점이 결합된 CloudGoat[1] 유형은 본질적으로 정책 설정 오류이므로 IAM 수준의 대응이 가능하다. 함수별 최소 권한을 반영한 Custom Role 정의, 권한 상승 작업의 분리, 정책 변경 전의 정적 시뮬레이션 및 Access Analyzer를 통한 검증, 변경 이후의 CloudTrail 기반 감사와 수동 대응 절차가 핵심 대응책이다. 최근에는 LLM을 보조적 분석

<표 3> 서버리스 환경의 공격 사례와 IAM 대응 효과

사례	취약점	IAM 효과성	보완 기술
CloudGoat[1]	권한 과다, 코드 취약	O	CustomRole
Warmonger[2]	공유 egress IP	X	네트워크 격리
LeakLess[3]	Transient, Side-channel	X	TEE, WASM

도구로 활용해 대규모 정책을 검토하거나 과도한 권한을 식별하는 시도도 있다.

Warmonger[2]와 같이 공유 egress IP 풀을 악용하는 공격은 IAM의 관할 밖이며 네트워크와 인프라 차원의 보완이 필요하다. 함수 단위 egress IP 분리, API Gateway 수준의 rate limiting과 WAF 적용, 네트워크 세그멘테이션과 함께 IP 기반 신뢰에서 벗어난 TLS 상호 인증 또는 토큰 기반 인증으로의 전환이 권장된다. 이 과정에서 LLM은 대규모 네트워크 로그를 요약해 이상 패턴을 관리자에게 전달하는 보조 도구로 활용될 수 있다.

LeakLess[3]가 제기하는 transient execution 및 side-channel 공격은 실행 런타임의 취약점을 이용하기 때문에 IAM으로는 근본적 차단이 불가능하다. 대응으로는 민감 데이터의 상시 암호화, TEE 또는 Confidential Compute 적용, WASM 기반 샌드박싱 및 코드 하드닝 등 실행환경 보강이 필요하다. LLM 기반의 로그와 리포트 요약은 잠재 취약점의 식별과 대응 준비에 보조적으로 기여할 수 있다.

표 3은 위 세 가지 사례를 취약점, IAM 효과성, 권장 보완 기술 관점에서 요약한다. 즉, IAM은 정책과 설정 기반 취약점에는 효과적이지만 네트워크 자원 공유나 런타임 결함에 대해서는 별도의 네트워크 및 실행환경 보강이 필요하다. 따라서 정책 검증, 로그 분석과 조치, 네트워크 격리와 런타임 강화를 결합한 다층 방어가 요구되며, LLM은 이를 보완하는 도구로 활용될 수 있다.

4. IAM 연구 동향과 향후 과제

IAM은 정책 및 설정 기반 취약점에 대해서는 효과적인 방어 수단이지만, 네트워크 공유나 런타임 결함과 같은 영역에서는 한계를 가진다. 이를 보완하기 위해 최근 연구들은 네 가지 방향으로 진행되고 있다. 첫째, 실행 로그를 동적으로 분석하여 실제 사용되는 권한만을 추출하고 불필요한 권한을 자동으로 제거한다. 둘째, 서버리스 함수를 사전에 분석하여 잠재적 보안 위협을 식별한다. 셋째, 그래프 기반 분석을 통해 함수 간의 권한 흐름을 식별하고, 데이터 유출 경로나 권한 상승 가능성이 존재하는 경로를 사전에 탐지한다. 넷째, Allow List 기반 접근 제어를 적용하여 서버리스 함수가 명시적으로 허용된 자원에만 접근하도록 제한한다.

이러한 연구들은 공통적으로 IAM의 기능을 확장하고 강화하는 데 기여하지만, 특정 클라우드 벤더 환경에 종속되거나 사전에 정의된 조건에 크게 의존한다는 한계가 존재한다. 또한, IAM으로 해결할 수 없는 런타임 취약점이나 컨테이너 격리 문제에 대해서는 여전히 한계가 존재한다. 따라서 향후 연구에서는 실행 시점의 맥락을 반영한 동적 조정, 서로 다른 클라우드 환경에서의 일관된 정책 생성, 그리고 런타임 행위와 환경적 맥락을 통합적으로 고려하는 차세대 IAM 프레임워크가 요구된다.

5. 결론

본 연구는 서버리스 환경에서 핵심적인 보안 메커니즘으로 작동하는 IAM을 대상으로 심층 분석을 수행하였다. 기존 연구가 특정 플랫폼에 한정되거나 단순 비교에 집중한 것과 달리, 본 연구는 주요 클라우드 벤더의 IAM 구조와 보안 취약점을 체계적으로 비교 및 분석함으로써, 현재 IAM이 직면한 한계를 실증적으로 드러냈다. 이러한 분석 결과는 클라우드 보안 구성을 담당하는 실무자에게는 정책 설계와 운영상의 실질적 시사점을 제공하며, 클라우드 컴퓨팅 보안을 발전시키고자 하는 연구자에게도 의미 있는 기초 자료로 기여할 수 있을 것으로 기대된다.

Acknowledgements

이 논문은 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원 - 학·석사연계ICT핵심인재양성 지원을 받아 수행된 연구임(IITP-2025-RS-2024-00437024)

참고문헌

[1] RHINO, “ClodGoat goes Serverless: A walkthrough of Vulnerable Lambda Functions”, 2025. [Online]. Available: https://rhinosecuritylabs.com/cloud-security/cloudgoat-vulnerable-lambda-functions/

[2] Xiong, Junjie, et al. "Warmonger: Inflicting denial-of-service via serverless functions in the cloud." Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. 2021.

[3] Rostamipoor, Maryam, Seyedhamed Ghavamnia, and Michalis Polychronakis. "LeakLess: Selective Data Protection Against Memory Leakage Attacks for Serverless Platforms." Proceedings of the Network and Distributed System Security Symposium (NDSS), San Diego, CA. 2025.