LamGuard: 서버리스 환경을 위한 LLM 기반 동적 검증 프레임워크

신창희[1], 이승수[2]

*인천대학교 (학부생[1], 교수[2])

LamGuard: LLM-based dynamic verification framework for serverless environments

Changhee Shin[1], Seungsoo Lee[2]

*Incheon National University (Undergraduate student[1], Professor[2])

요 약

서버리스 컴퓨팅 환경에서는 개발자가 인프라스트럭처를 직접 관리할 필요 없이 비즈니스 로직에 집중할 수 있으나, 함수에 과도한 권한이 부여되거나 호출 검증이 누락되는 등 보안 취약점이 발생할 수 있다. 본 논문에서는 대규모 언어 모델(LLM)을 활용하여 서버리스 함수의 최소 권한을 자동으로 추출하고, 보안 로직을 삽입하는 프레임워크 LamGuard를 제안한다. LamGuard는 함수 코드와 Config 파일을 분석해 AWS SDK 호출 및 트리거 유형을 식별하고, 이를 기반으로 최소 권한과 호출 검증 로직을 생성한다. 생성된 보안 로직은 함수 코드에 자동 삽입되어 허용되지 않은 API 호출이나 비정상적인 트리거 조건을 차단한다. AWS Lambda 환경에서의 실험을 통해 LamGuard의 적용 가능성과 보안 자동화 효과를 확인하였다.

I. Introduction

서버리스 컴퓨팅은 개발자가 인프라스트럭처를 직접 관리하지 않고 비즈니스 로직에 집중할 수 있게 하여 클라우드 환경에서 널리 사용되고 있다. 개발자가 작성한 코드를 함수 단위로 배포하면 클라우드 제공자가 이를 자동 실행하고 리소스를 동적으로 할당한다. AWS Lambda, Google Cloud Functions, Azure Functions 등은 이벤트 기반 트리거와 높은 확장성으로 다양한 애플리케이션에 활용된다. 그러나 IAM(Identity and Access Management) 기반 권한 관리와 이벤트 기반 실행 메커니즘은 과도한 권한 부여, 악성 이벤트 유입 등 보안 취약점을 야기할 수 있다[1]. 본 논문에서는 서버리스 컴퓨팅의 구조적 한계와 보안 취약점을 해결하기 위해 LamGuard를 제안한다. LamGuard는 LLM 기반 코드 분석을 통해 함수 로직을 자동 분석하고, 최소 권한을 추출해 적절한 보안 로직을 삽입함으로써 함수의 보안성과 신뢰성을 향상시킨다.

본 논문의 주요 기여는 다음과 같다:

1. 최소 권한 추출 및 보안 로직 생성: 함수 로직 분석을 통해 필요한 최소 권한을 정확히 추출하고, 권한 남용을 방지하기 위한 보안 로직을 삽입함으로써 서버리스 함수의 신뢰성과 안정성을 강화한다.

2. 권한 및 보안 시스템 자동화: 개발자가 작성한 함수 코드를 기반으로 최소 권한 설정과 보안 로직 삽입 과정을 자동화하여, 수작업 개입을 최소화하고 서버리스 환경에서의 운영 효율성을 극대화한다.

II. Background

2.1 서버리스 함수

서버리스 함수는 클라우드 환경에서 이벤트에 따라 실행되는 경량화된 독립 실행 단위로, 상태를 유지하지 않는(stateless) 특성과 단일

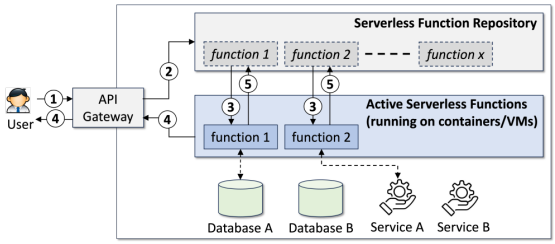

Fig 1. The workflow of a serverless function execution

목적의 작업 수행을 특징으로 한다. Fig 1.과 같이 함수는 요청이 발생할 때만 활성화되며, 클라우드 제공자가 자동으로 필요한 리소스를 할당하고 실행 환경을 구성한다. 실행되지 않는 동안에는 자원을 점유하지 않아 비용 효율성이 높으며, 요청 수에 따라 리소스가 자동으로 확장 또는 축소된다[2].

2.2 Large Language Model(LLM)

대규모 언어 모델(Large Language Model, LLM)은 대량의 텍스트 데이터를 기반으로 학습된 신경망 모델로, 자연어 처리의 핵심 기술로 자리 잡고 있으며, 문장 생성, 텍스트 분류, 질의응답 등 다양한 언어 처리 작업에서 높은 성능을 보인다. 최근에는 복잡한 문맥 이해 능력을 바탕으로 소프트웨어 코드와 같은 구조적 데이터에도 효과적으로 적용되고 있다. 사전 훈련된 LLM은 비교적 소량의 도메인 특화 데이터로 파인튜닝함으로써, 특정 과업에 대한 성능을 효율적으로 향상시킬 수 있다.

III. System Architecture

3.1 LamGuard Design

LamGuard는 서버리스 함수의 보안성을 향상시키기 위해, 함수 코드와 Config 파일을 분석하여 최소 권한과 트리거 조건을 식별하고, 이 정보를 기반으로 통합된 보안 로직을 자동 삽입하도록 설계되었다. Fig 2.는 LamGuard의 전체 동작 흐름을 나타낸다. 생성된 보안 로직은 함수 진입 지점에 삽입되며, 최종 함수 코드는 배포 가능한 형태로 반환된다. 이를 통해 개발자는 최소 권한 정책이 적용된 보안 강화 함수를 생성할 수 있다.

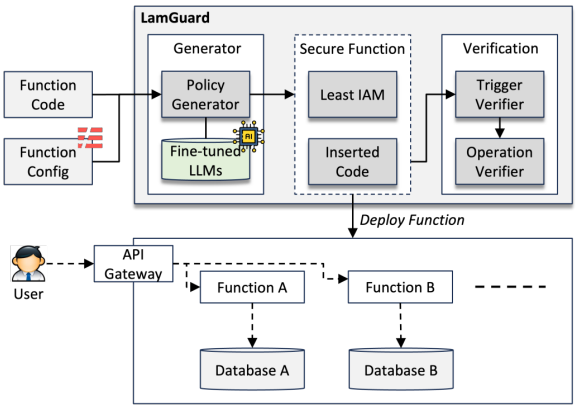

Fig 2. The architecture of LamGuard

3.2 Least Privilege 추출

LamGuard는 DeepSeek 기반 LLM을 활용해 서버리스 함수에 필요한 최소 권한을 자동으로 추출한다. 각 함수의 소스코드와 Config 파일은 입력 프롬프트로 구성되며, 출력은 해당 함수에 부여할 permissions와 보안 로직이 삽입된 전체 코드(Result)를 JSON 형식으로 생성하도록 모델을 학습시켰다. prompt-completion 형태의 학습 데이터에는 permissions 및 Result 필드를 포함하여, IAM 권한 요구사항을 반영하도록 구성되었다. 모델은 코드 내 AWS SDK 호출과 리소스 접근 패턴을 기반으로 실제 필요한 최소 권한만을 정확히 식별해 출력한다.

3.3 Dynamic Verification

LamGuard의 보안 로직은 AWS SDK 호출 감시와 함수 트리거 유형 검증 기능으로 구성되며, Config 파일과 최소 권한 정보에 기반해 통합적으로 생성된다. 모니터링 로직은 permissions를 바탕으로 허용 리스트를 정의하고, 실행 중 외부 호출이 이 범위를 벗어날 경우 즉시 차단하고 로그로 기록한다. 트리거 유형 검증은 Config의 events 필드를 분석해 직접 호출, HTTP 호출, 이벤트 기반 호출로 분류하고, 이 결과를 기반으로 런타임 시 트리거 조건을 검증하는 코드를 삽입한다. 이 두 기능은 하나의 보안 로직으로 함수 진입 지점에 삽입되며, 실행 조건을 벗어난 호출을 효과적으로 차단한다.

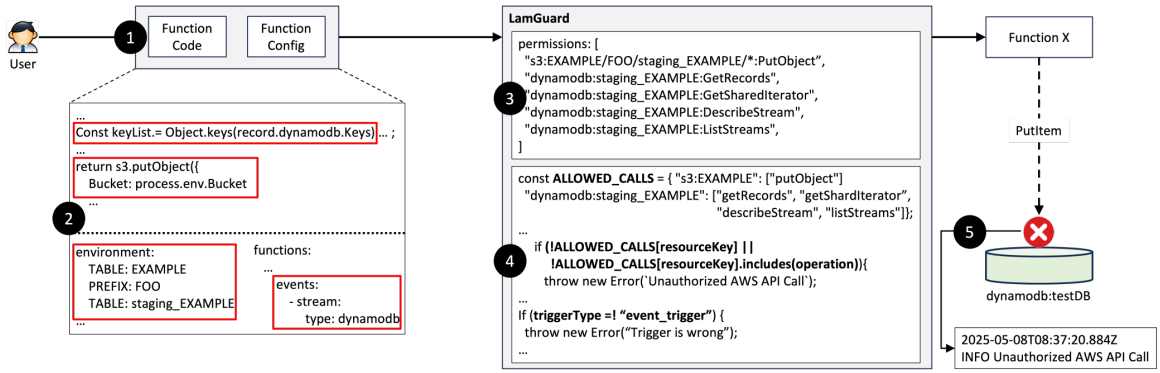

Fig 3. The workflow of LamGurad

IV. Evaluation

4.1 Test Environments

본 실험은 AWS Lambda의 ap-northeast-2 리전에서 128MB 메모리 환경으로 구성하고, 함수 배포 및 관리는 Serverless Framework v4.25를 통해 수행하였다. LamGuard는 약 10만 개의 학습 샘플로 구성된 데이터셋을 기반으로 학습되었으며, LLM 파인튜닝은 NVIDIA H100 GPU 서버에서 진행되었다. 본 실험은 LamGuard의 실제 서버리스 환경에서의 적용 가능성과 효율성을 평가하기 위한 목적으로 수행되었다.

4.2 Use Case

Fig 3.은 LamGuard를 활용하여 실제 서버리스 함수에 적용하는 시나리오를 나타낸다. 먼저, 사용자는 함수의 소스코드와 Config 파일을 시스템에 입력한다 (①). 시스템은 이를 분석하여 AWS SDK 호출과 실행 트리거 유형을 식별하고, 명시되지 않은 리소스 정보는 Config 파일로부터 보완한다 (②). 분석 결과를 바탕으로 최소 권한(permissions) 목록이 생성되며, 실시간 모니터링 및 이벤트 검증을 위한 보안 로직이 구성된다 (③). 해당 로직은 함수 실행 흐름을 방해하지 않도록 자동으로 삽입된다 (④). 이후, 보안 로직이 삽입된 함수가 의도되지 않은 외부 서비스에 접근할 경우, 시스템은 이를 감지하고 “Unauthorized AWS API Call” 오류를 반환한다 (⑤). 이러한 흐름은 LamGuard가 실행 중 발생할 수 있는 권한 초과 접근을 효과적으로 차단하며, 서버리스 환경에서 실시간 정책 강제와 동작 검증이 가능함을 보여준다.

V. Conclusion

LamGuard는 서버리스 함수의 최소 권한 생성과 보안 로직 삽입을 자동화함으로써, 운영 편의성과 시스템 보안성을 동시에 향상시킨다. 본 실험을 통해 LLM 기반 보안 자동화의 실효성과 실제 서버리스 환경에서의 적용 가능성을 입증하였다. 하지만, 클라우드 벤더 및 프로그래밍 언어의 확장성에서의 한계가 존재한다. 이를 위해 향후 연구에서는 더 다양한 데이터셋을 구축하고, 이를 기반으로 클라우드 벤더 및 프로그래밍 언어에 대한 적용 범위를 확장함으로써 시스템의 범용성과 확장성을 강화할 예정이다.

ACKNOWLEDGEMENT

이 논문은 2022년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (RS-2022-II220740, 다크웹 은닉서비스 식별 및 근원지 추적기술 개발).

[참고문헌]

[1] Shafiei, H., Khonsari, A., & Mousavi, P. (2022). Serverless computing: a survey of opportunities, challenges, and applications. ACM Computing Surveys, 54(11s), 1-32.

[2] Eismann, S., Scheuner, J., Van Eyk, E., Schwinger, M., Grohmann, J., Herbst, N., ... & Iosup, A. (2020). Serverless applications: Why, when, and how?. IEEE software, 38(1), 32-39.