Bambda: 서버리스 환경을 위한 우회 공격 차단 프레임워크

신창희[1], 이승수[2]
[1]인천대학교 컴퓨터공학부 학부생
[2]인천대학교 컴퓨터공학부 교수
changhee9149@inu.ac.kr, seungsoo@inu.ac.kr

Bambda: A Framework for Preventing Evasion Attacks 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

서버리스 컴퓨팅의 급속한 확산과 더불어 보안 위협도 증가하고 있다. AWS 서버리스에서 제공하는 여러 보안 방법 중, IAM 정책을 설정하여 함수와 자원 간 접근 권한을 설정할 수 있지만, 정적 보안 메커니즘 한계로 동적 우회 공격은 막기 어렵다는 한계를 가진다. 본 연구에서 제안하는 Bambda는 플러그인 방식의 Lambda와 CloudWatch를 통해 실시간 동적 검증을 수행하여 이러한 우회 공격 방어의 한계를 극복한다.

1. 서론

서버리스 컴퓨팅은 개발자가 서버 인프라 관리 없이 애플리케이션을 배포하고 사용량 기반으로 비용을 지불하는 모델로, 운영 복잡성 감소와 확장성 향상을 제공한다[1]. 이러한 이점으로 인해 서버리스 컴퓨팅의 채택이 급속히 확산되고 있으나, 새로운 보안 과제도 함께 대두되고 있다. 특히, IAM(Identity and Access Management)이 중요한 관심사로 부각되고 있다. IAM은 AWS Lambda와 같은 서버리스 환경에서 사용자 인증 및 권한 관리를 담당하는 핵심 보안 메커니즘이다. 하지만, Lambda에 대한 부적절한 IAM 구성은 무단 데이터 조작의 위험을 초래할 수 있으며, 연쇄적인 Lambda 호출을 통한 비용 증가를 야기할 수 있다. 2020년 DivvyCloud가 발표한 보고서에 따르면, IAM 구성 오류로 인한 기업의 경제적 손실이 약 5조 달러에 달하는 것으로 추정되었다[2]. AWS는 사전 IAM 정책 검증 도구는 제공하지만 배포된 이후 실시간으로 보안 위협을 감지하고 차단하는 서비스는 제공하지 않아 동적인 서버리스 환경에서의 보안 취약점이 존재한다.

이러한 한계를 극복하기 위해 본 연구에서는 Bambda를 제안한다. Bambda는 CloudWatch를 이용한 실시간 로그 분석과 전달받은 Event Value를 기반으로 우회 공격을 감지하고 차단한다.

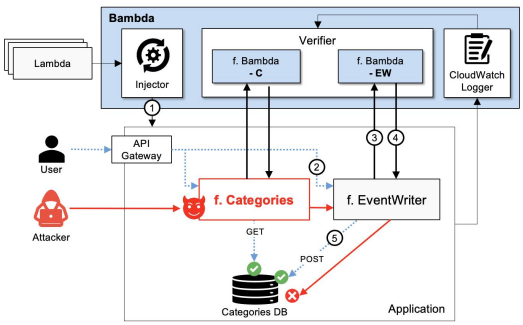
2. Bambda 디자인

2.1 문제상황

서버리스 환경에서 부적절한 권한을 가진 Lambda의 배포는 심각한 보안 위협을 초래할 수 있다. 예를 들어, 다른 Lambda를 호출할 수 있는 권한을 가졌지만 특정 리소스에 직접적인 접근을 할 수 없는 Lambda가 다른 Lambda를 호출하여 간접적으로 해당 리소스에 접근할 수 있는 경우가 발생할 수 있으며 이는 IAM 정책으로 차단할 수 없다. 그림 1에서 보이듯이, 공격자에 의해 배포된 'Categories' Lambda는 'EventWriter' Lambda를 호출함으로써 접근 권한 정책을 우회하여 Categories DB에 무단으로 접근하고 데이터를 조작할 수 있다. 따라서 이러한 우회 공격을 방어하기 위해 실행 시간에 동적으로 Lambda 호출 패턴을 모니터링하고 제어할 수 있는 메커니즘이 필요하다.

2.2 Bambda 아키텍처 개요

Bambda는 AWS Lambda에 플러그인 형태로 적용되는 우회 공격 차단 프레임워크이다. 이 프레임워크는 AWS CloudWatch Log를 활용하여 실시간 모니터링 및 검증을 수행한다. 그림 1은 Bambda가 적용된 아키텍처를 보여주며, 그 실행 흐름은 다음과 같다. (1) Injector를 사용하여 기존 Lambda에 실

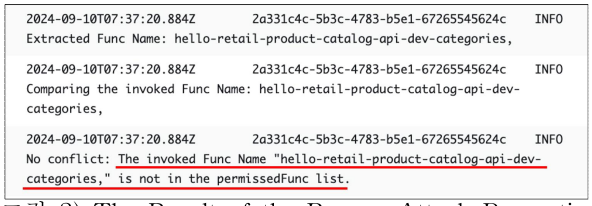


(그림 1) Motivating Example and Architecture

시간 로그 기능을 포함하고, 리소스에 접근하는 Lambda인 경우, Bambda의 검증 기능이 포함된 형태로 변환한다. (2) 변환된 Lambda는 호출 요청이 API Gateway를 통해 왔는지, 또는 aws-sdk를 이용한 직접 호출인지를 구분한다. (3) 직접 호출로 식별된 경우, Lambda는 받은 event 값을 Event Value로 하여 Bambda을 호출한다. (4) Bambda는 CloudWatch Logger를 통해 Lambda의 실행 기록과 전달받은 Event Value를 분석한다. 이를 기반으로 해당 호출이 허용된 접근인지 우회 공격인지를 판단한다. (5) Bambda의 판단 결과에 따라, Lambda에 주입된 코드는 리소스 접근을 허용하거나 차단한다.

2.3 실시간 로깅 및 검증 메커니즘

Bambda의 핵심 기능인 실시간 로깅 및 검증 메커니즘은 CloudWatch Log의 한계를 극복하고 동적인 Lambda 호출을 효과적으로 모니터링하기 위해 설계되었다. 기존 CloudWatch Log는 Lambda 실행 종료 후 일괄 로깅하는 방식이었으나, Bambda는 이를 개선하여 모든 Lambda가 공유하는 단일 CloudWatch Log Group을 생성하고, Injector를 통해 변환된 Lambda가 실행 즉시 이 Log Group에 정보를 기록하도록 했다. 검증 과정에서 Bambda는 먼저 리소스에 접근하는 Lambda와 해당 Lambda를 호출할 수 있는 Lambda 목록을 사전에 배열로 저장한다. Bambda 호출 시, Event Value로 전달받은 Lambda의 이름과 Request ID를 실시간 로그와 대조하여 존재 여부를 확인한다. 다음으로, 전달받은 Lambda 이름이 허용 목록에 포함되어 있는지 검증한다. 이 과정을 통과하면 정상적인 호출로 판단하여 리소스 접근을 허용하고, 그렇지 않으면 우회 공격으로 판단하여 차단한다. 이러한 다층적 검증 절차를 통해 동적인 서버리스 환경에서 발생할 수 있는 우회 공격 패턴을 효과적으로 탐지하고 차단할 수 있다.



(그림 2) The Result of the Bypass Attack Prevention.

3. 프로토타입 평가

실험은 AWS 환경에서 Serverless Framework 4.25 버전을 사용하여 Hello Retail! 애플리케이션을 배포하여 진행하였다[3]. 'Categories' Lambda가 악성 배포되어 내부적으로 'EventWriter' Lambda를 호출해 DynamoDB에 악의적인 값을 삽입하는 시나리오를 가정했다. 그림 2는 Bambda가 이러한 우회적 접근을 성공적으로 탐지하고 차단하는 로그를 보여준다. 이를 통해 Bambda의 효과적인 우회 공격 방지 기능을 확인할 수 있었다.

4. 결론

본 연구에서는 서버리스 컴퓨팅 환경의 보안 문제를 해결하고자 Bambda를 제안하였다. Bambda는 AWS Lambda에서 실시간 로깅과 동적 검증을 통해 우회적 Lambda 호출을 효과적으로 탐지 및 차단한다. 실험 결과, Bambda가 부적절한 권한을 가진 Lambda의 무단 리소스 우회 접근을 성공적으로 방지했으며, 서버리스 환경에서의 IAM 보안을 강화하고 복잡한 Lambda 간의 상호작용에서 발생할 수 있는 보안 위협에 효과적으로 대응함을 확인하였다.

ACKNOWLEDGEMENT

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 학석사연계 ICT핵심인재양성사업의 연구결과로 수행되었음(IITP-2024-RS-2024-00437024).

참고문헌

[1] Eismann, Simon, et al. "Serverless applications: Why, when, and how?." IEEE Software 38.1 (2020): 32-39.

[2] DivvyCloud, "2020 Cloud Misconfiguration Report", 2021. [Online]. Available: https://www.bankinfosecurity.com/whitepapers/2020-cloud-misconfigurations-report-w-6009

[3] SimonEismann. "Hello Retails! Application", 2021. [Online]. Available: https://github.com/SimonEismann/hello-retail