AWS Lambda 환경에서 Amazon S3 에서의 평문 노출을 막기 위한 스트리밍 기반 파일 암·복호화 API 설계

김예찬, 김시환, 안승민, 김형지, 조재문, 배채은, 권다은, 김동찬*
국민대학교

{tomking0820, sihwan031008, bryan0126, aloa2012, henryjm, co5eun, ekdms3809, dckim*}@kookmin.ac.kr

Streaming-Based File Encryption/Decryption API Design to Prevent Plaintext Exposure in Amazon S3 within AWS Lambda Environments

Kim Yae-Chan, Kim Si-Hwan, Ahn Seung-Min, Kim Hyeong-Ji, Jo Jae-Moon,
Bae Chae-Eun, Kwon Da-Eun, Kim Dong-Chan*
Kookmin Univ.

요 약

본 논문은 공격자가 해당 서비스를 운영하는 조직의 운영자 수준의 권한을 확보한 상태(assumed-breach)를 전제로, AWS Lambda-Amazon S3 기반 파일 처리에서 S3에 평문이 존재할 때 스토리지 접근으로 평문을 획득할 수 있는 위험을 줄이기 위한 스트리밍 파일 암·복호화 API 설계를 제안한다. 제안한 API는 업로드/다운로드 전 과정에서 데이터를 스트리밍 방식으로 처리하고, S3에는 암호문만 저장되도록 구성한다. 또한 MITRE ATT&CK 기반 위협 모델에서 정의한 평문이 수집·집결을 거쳐 공격자 소유의 클라우드 계정으로 전달되는 유출 과정을 기준으로, S3에 암호문만 저장하는 구조가 스토리지 직접 접근을 통한 평문 획득을 제한함을 분석한다. 다만 키 또는 키 관리 로직이 실행 코드에 포함되어 있어 역공학에 의한 키 노출 가능성을 배제하기 어렵다는 점을 논의한다.

Ⅰ. 서론

클라우드 환경에서 평문 객체가 스토리지에 저장된 상태로 운영되는 구성은 데이터 유출 공격의 주요 원인으로 지적되어 왔다. 이에 Onyia 등은 S3 버킷 설정에 대한 실증 연구를 통해 평문의 장기 잔존 위험을 분석하였으며, Cloud Storage Security의 기술보고서에서도 동일한 구성으로 인한 실제 데이터 유출 사례를 보고하였다[1,2].

이와 같은 평문 객체가 존재하는 환경에서는 계정 탈취 또는 클라우드 인프라 자체의 침해로 인해, 공격자가 해당 서비스를 운영하는 조직의 운영자 수준의 권한을 획득하는 상황이 발생할 수 있다. 본 논문에서는 Microsoft의 Zero Trust 모델에 따라 침해가 이미 발생한 상황(assumed-breach)을 전제로 한다[3].

AWS는 SSE-S3, SSE-KMS 등 서버 측 암호화를 제공한다[4]. 그러나 SSE는 저장 시 객체가 암호화되더라도, 권한을 가진 호출자는 GetObject 응답으로 복호화된 데이터를 수신하므로, assumed-breach 상황에서 IAM 권한이 탈취되면 평문 노출을 방지하기 어렵다. 본 논문은 S3에 암호문만 저장되도록 하는 Lambda 기반 암호화 방식을 제안하며, 기여는 다음과 같다. 먼저 assumed-breach 가정을 전제로 MITRE ATT&CK 기반 위협 모델을 정의하고 대응 범위를 분석하였다. 이를 근간으로 Lambda Function URL의 요청·응답 크기 제약을 고려한 스트리밍 기반 파일 암·복호화 API를 설계하였다.

논문의 구성은 다음과 같다. Ⅱ절에서는 위협 모델을, Ⅲ절에서는 시스템 설계를, Ⅳ절에서는 성능 평가를 기술한다.

Ⅱ. MITRE ATT&CK 기반 위협 모델

본 절에서는 서버리스 환경에서 파일 암·복호화를 수행하는 애플리케이션을 대상으로 위협 모델을 정의한다.

공격자는 계정 탈취 또는 클라우드 인프라 침해를 통해 정상적인 인증 정보와 접근 권한을 확보한 상태이며, 이를 기반으로 클라우드 스토리지 및 서버리스 실행 환경에 대해 운영자 수준의 접근이 가능하다고 가정한다. 구체적으로, 공격자는 S3 버킷에 대한 읽기·쓰기 권한을 보유하고, Lambda 함수의 실행 로그 및 환경 변수에 접근할 수 있으며, API Gateway 또는 Lambda Function URL을 통한 요청·응답을 관찰할 수 있다.

본 논문에서 고려하는 공격자의 주요 전술 및 기술은 MITRE ATT&CK 프레임워크를 기반으로 하며, [표 1]에 해당하는 기법을 포함한다[5]. 공격자의 목표는 클라우드 저장소에 직접 접근하여 자동화된 방식으로 데이터를 수집·집결한 뒤, 공격자 소유의 클라우드 계정으로 전송하여 조직의 중요 정보를 외부로 유출하는 것이다.

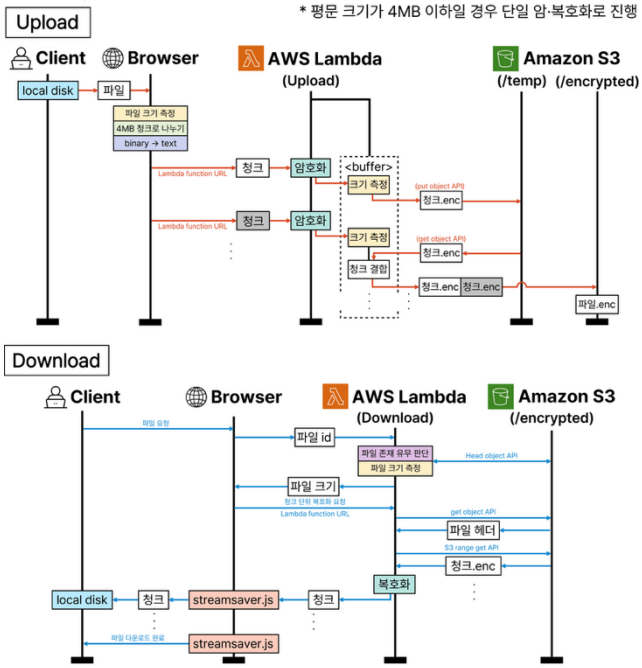
[표 1] MITRE ATT&CK 기반 위협 모델 포함 주요 기법

ID	기법명	설명
T1530	Data from Cloud Storage	클라우드 스토리지 직접 접근
T1537	Transfer Data to Cloud Account	공격자 계정으로 데이터 전송
T1567.002	Exfiltration to Cloud Storage	클라우드로 데이터 유출
T1119	Automated Collection	자동화된 데이터 수집
T1074.002	Remote Data Staging	유출 전 데이터 집결
T1213	Data from Information Repositories	내부자 위협 / 정보

Ⅲ. 스트리밍 기반 파일 암·복호화 API 설계

[그림 1]은 AWS Lambda- Amazon S3 기반 서버리스 환경에서 제안하는 스트리밍 암·복호화 API의 전체 구성과 데이터 흐름을 나타낸다. 사용자는 브라우저를 통해 업로드/다운로드를 수행하며, 브라우저는 Lambda Function URL을 통해 파일을 일정 단위로 분할하여 Lambda로 전송하거나 복호화를 요청한다. Lambda는 업로드 시 암호문 형태로 S3에 저장하고, 다운로드 시 S3에 저장된 암호문 객체(file.enc)를 조회하여 스트리밍 방식으로 복호화 처리 결과를 브라우저로 반환한다. Amazon S3는 처리 중간 산출물을 위한

임시 영역(/temp)과 암호문 저장을 위한 영역(/encrypted)으로 구분되어, 업로드 및 다운로드 처리 단계에 따라 저장 영역을 분리한다.



[그림 1] 파일 암·복호화 드라이브 API 전체 설계 구조

본 시스템은 AES-128-CTR 모드와 HMAC-SHA256을 결합하여 기밀성과 무결성을 제공한다[6]. CTR 모드는 임의 위치에서의 복호화가 가능하므로, 분할된 데이터 단위의 스트리밍 처리에 적합하다. 각 파일에 대해 128비트 IV를 무작위로 생성하고, 시그니처, 버전 정보, 원본 크기, IV, HMAC 값을 포함하는 72바이트 헤더를 파일 선두에 부가한다.

데이터 처리 단위(이하 청크) 크기는 Lambda Function URL의 응답 본문 제한(6MB)과 Base64 인코딩 오버헤드(약 33%)를 고려하여 4MB로 설정하였다. 4MB 원본 데이터는 인코딩 시 약 5.3MB가 되어 제한 범위 내에서 처리 가능하다. S3 Multipart Upload의 최소 파트 크기(5MB) 충족을 위해 Lambda는 데이터 처리 단위를 버퍼에 누적한 후 조건 충족 시 업로드를 수행한다.

i. 스트리밍 방식 업로드 API 처리 절차

Client는 로컬 디스크의 파일을 브라우저로 전달한다. 브라우저는 파일 크기 측정 한 뒤, 파일을 청크 단위로 분할하고 전송을 위해 binary→ text 변환을 수행한다. 이후 브라우저는 Lambda Function URL을 통해 분할된 파일 데이터를 Lambda로 전송한다.

Lambda는 수신된 각 청크를 암호화하여 내부 임시 버퍼에 적재하고, 암호문의 크기가 S3 Multipart Upload API의 업로드 조건을 충족하는지 여부를 판단한다. 해당 조건을 충족하지 않는 경우, Lambda는 해당 암호문을 /temp에 단일 객체로 저장한다. 이후 추가적인 청크가 수신되면, Lambda는 새로 암호화된 청크와 S3 임시 폴더(/temp)에 저장되어 있던 기존 암호문을 Get Object API로 조회하여 버퍼에서 결합하고, 누적 크기를 측정한다.

누적 크기가 S3 Multipart Upload API 기준에 도달하면, Lambda는 업로드 절차를 수행하여 암호화된 데이터를 S3(/encrypted)에 저장한다. 이후 모든 데이터 처리 단위에 대해 동일한 스트리밍 처리 흐름을 반복함으로써, 파일 전체가 암호화된 상태로 S3에 업로드 되도록 구성된다.

ii. 다운로드 API 처리 절차

Client는 브라우저를 통해 복호화 파일을 요청하며, 브라우저는 Lambda Function URL을 통해 Lambda에 파일 식별자(file id)를 전달한다. Lambda는 Head Object API를 통해 S3의 /encrypted에서 대상 파일의 존재 유무 여부와 크기를 확인한 후, 파일 크기 정보를 브라우저로 전달한다. 브라우저는 파일 크기에 따라 단일 요청 또는 데이터 처리 단위 복호화를 요청하며, Lambda는 복호화에 필요한 헤더 정보를 확보하기 위해 Get Object API를 통해 암호화된 파일의 선두 구간을 수신한다. 이후 각 요청에 대해 S3 Range Get API를 통해 청크 단위 만큼 수신하고, 복호화하여 브라우저로 반환한다. 브라우저는 반환된 평문 데이터를 streamsaver.js를 통해 로컬 디스크에 순차적으로 기록한다. 이 과정을 반복함으로써, 파일 전체가 로컬 디스크에 저장되도록 구성된다. 모든 구간에 대한 복호화 데이터 수신이 종료되면, 브라우저는 스트림을 종료하고 streamsaver.js를 통해 파일 저장을 완료한 뒤 다운로드 완료 메세지를 Client에게 전송한다.

IV. 결론 및 향후 연구

본 논문은 assumed-breach 상황을 전제로, 클라우드 객체 스토리지에 평문이 저장되는 구성에서 발생하는 직접 접근 기반 유출 위험을 완화하기 위해 AWS Lambda 환경의 AES 기반 스트리밍 파일 암·복호화 API를 설계하였다. 제안한 API는 업로드·다운로드 전 과정에서 데이터가 평문 형태로 S3에 저장되지 않도록 구성하여, 저장소 직접 접근을 통한 데이터 획득 가능성을 구조적으로 낮춘다.

공격자가 S3 및 실행 환경에 내부자 수준 권한을 확보하더라도 S3에 암호문만 남도록 구성되므로, 스토리지 직접 접근으로 평문을 확보하기 어렵다. 이에 따라 데이터 획득·수집·집결·유출(T1530, T1119, T1074.002, T1567.002, T1537)로 이어지는 유출 과정의 실효성이 저하될 수 있다. 즉, 본 연구의 핵심 효과는 assumed- breach에서도 유출의 전제가 되는 저장 계층에 평문 상태의 파일이 잔존하지 않게 함으로써 공격 비용을 증가시키는 데 있다.

한편, 현재 설계는 layer가 포함된 Lambda 실행코드에 암호화 키 또는 키 관리 로직이 포함되어 있다. 따라서 공격자가 해당 바이너리에 접근할 수 있는 경우, 정적 분석 또는 동적 계측을 통해 키 관련 정보가 노출될 가능성을 완전히 배제하기 어렵다. 이는 저장 계층에서의 평문 제거만으로는 키 보호까지 달성되지 않으며, 서버리스 환경에서 실행 코드와 키 관리가 보안 경계의 일부가 됨을 의미한다.

향후 연구에서는 키 소재를 코드로부터 분리하거나 노출 난도를 높이기 위해 키 분할·난독화 등 정적 분석 저항 기법을 적용하고, AWS KMS 기반 envelope encryption을 도입하여 키 관리·회전·접근 통제를 체계화함으로써, 실행 환경 침해를 가정한 상황에서도 키 추출 및 오·남용 가능성을 추가로 완화하는 방안을 검토할 예정이다. 이를 통해 본 연구가 제시한 스토리지 계층의 기밀성 확보를 기반으로, 키 보호까지 포함하는 종합적 방어 범위로 확장하는 것을 목표로 한다.

ACKNOWLEDGMENT

이 논문은 국민대학교 2025학년도 동계방학 학부생 연구참여 프로그램(UROP)의 지원과 2025년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(No. RS-2024-00397105, KCMVP 보안수준 3 암호모듈 제작을 위한 핵심기술 개발).

참 고 문 헌

[1] Davies R. W." The Data Encryption standard in perspective, "Computer Security and the Data Encryption Standard, pp. 129-132.

[2] Cloud Storage Security / Casmer Labs,"Public S3 Bucket Exposure: Misconfiguration Risks in 2025," Technical Report, 2025.

[3] Microsoft, "Zero Trust implementation guidance," Microsoft Security Documentation, 2024.

[4] AWS,"Protecting data by using client-side encryption," Amazon S3 User Guide, AWS Doc-umentation.

[5] MITRE Corporation,"MITRE ATT&CK Framework for Enterprise and Cloud," MITRE ATT&CK Knowledge Base, accessed Jan. 2026.

[6] NIST,"FIPS 197: Advanced Encryption Standard(AES)," Federal Information Processing Stan-dards Publication 197, 2001 (updated 2023).