AhnLab Cyber Threat Intelligence Report

TLP : CLEAR

분석 보고서

JWT 인증의 명과 암

이커머스 플랫폼 보안 붕괴, 미흡한 JWT 인증 관리

안랩 시큐리티 인텔리전스 센터(ASEC)

2026. 01

AhnLab

목차

개요

2025년말, 국내 최대 이커머스 플랫폼 개인정보 유출 사태가 불거졌다. 2025년 12월 기준, 유출 규모는 3천만 건이 넘는 것으로 알려졌다. 이번 보안 사고의 핵심은 해당 기업의 JWT(JSON Web Token) 인증 시스템이 취약하게 관리되었다는 점이다. 당사에서 근무했던 개발자의 JWT 서명키가 퇴사 후에도 리셋되지 않았고, 공격자는 유효한 서명키를 활용해 고객 정보를 제한 없이 조회할 수 있었다.

이번 사건의 중심에 있는 JWT 인증은 현대 웹/모바일 애플리케이션의 표준으로 자리잡고 있다. 무상태(stateless) 인증의 편리함을 제공하지만, 관리가 제대로 되지 않으면 인증 체계 전체를 붕괴시킬 수 있는 단일 실패 지점(Single Point of Failure) 이 될 수도 있다.

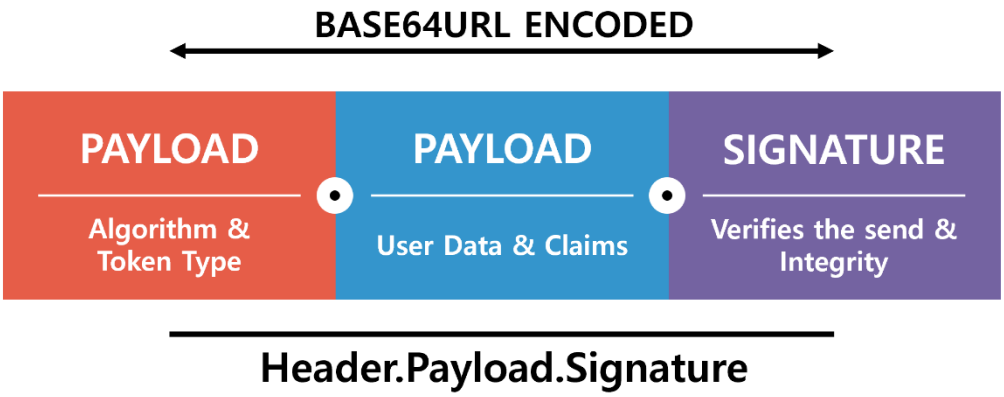
본 문서에서는 JWT의 개념과 인증 방식을 소개한 뒤, CVE 사례 중심으로 주요 취약점을 분석하고 이를 예방 및 완화하기 위한 실질적인 보안 전략을 제시한다.

JWT에 대하여

JWT는 웹 표준(RFC 7519)으로, JSON 객체를 사용해 두 개체 간 정보를 안전하게 전송하는 가볍고 '자가 수용적인(compact and self-contained)' 인증 방식이다. JWT 인증 토큰은 모든 필수 정보를 자체적으로 담고 있어, 서버에서 데이터베이스 조회 없이 사용자 인증 및 권한 확인이 가능하다. 또한, 무상태(stateless) 구조로 되어 있어 서버 측 세션의 저장 및 동기화 부담을 줄이고, HTTP 헤더나 URL 파라미터로 쉽게 전달 가능하다. 분산된 환경에서 확장과 전달이 용이하다는 장점을 바탕으로 현대 웹/마이크로서비스 환경의 핵심 인증 방식으로 자리 잡았다.

1. JWT의 구조 및 구성 요소

JWT는 Base64Url로 인코딩 된 세 부분(Header, Payload, Signature)이 마침표(.)로 구분되어 결합된 형태로 구성된다.



[그림 1] JWT 구조

1) 헤더: 메타 정보

헤더(Header)는 토큰의 메타 정보를 담고 있는 JSON 객체를 Base64Url로 인코딩한 부분이다. typ (Type, 일반적으로 JWT) 및 alg (Algorithm, 서명에 사용된 암호화 알고리즘, 예: HS256, RS256) 클레임을 필수적으로 포함한다.

```
{
  "alg": "HS256",   // 서명 알고리즘
```

```
  "typ": "JWT"      // 토큰 타입
}
```

[표 1] JWT의 헤더 구성

2) 페이로드: 클레임

페이로드(Payload)에는 전송하고자 하는 인증, 권한 및 기타 속성 정보가 JSON 객체 형태로 포함된다. 이를 클레임(Claim)이라 부른다. 페이로드 역시 헤더와 마찬가지로 Base64Url로 인코딩된다.

- **클레임 종류:**
  클레임은 역할에 따라 Registered(표준 정의), Public(공개), Private(사설) 세 종류로 나뉜다. Registered Claims는 토큰의 만료 시점(exp), 발급 시점(iat), 발급자(iss), 수신자(aud) 등 토큰의 유효성 검증에 활용되는 표준 필드를 포함한다.
- **보안 유의점:**
  헤더와 페이로드는 암호화가 아닌 단순 인코딩이므로, 페이로드를 확보하면 손쉽게 디코딩해 내용을 파악할 수 있다. 따라서, 페이로드에는 이메일, 결제 정보 등 민감한 개인 식별 정보(PII)가 포함되면 안 된다.

```
{
  "sub": "user123",              // 사용자 ID
  "name": "John Doe",            // 사용자 이름
  "role": "admin",               // 권한
  "iat": 1516239022,             // 발급 시각
  "exp": 1516242622              // 만료 시각
}
```

[표 2] JWT의 페이로드 구성

3) 서명: 무결성 보장

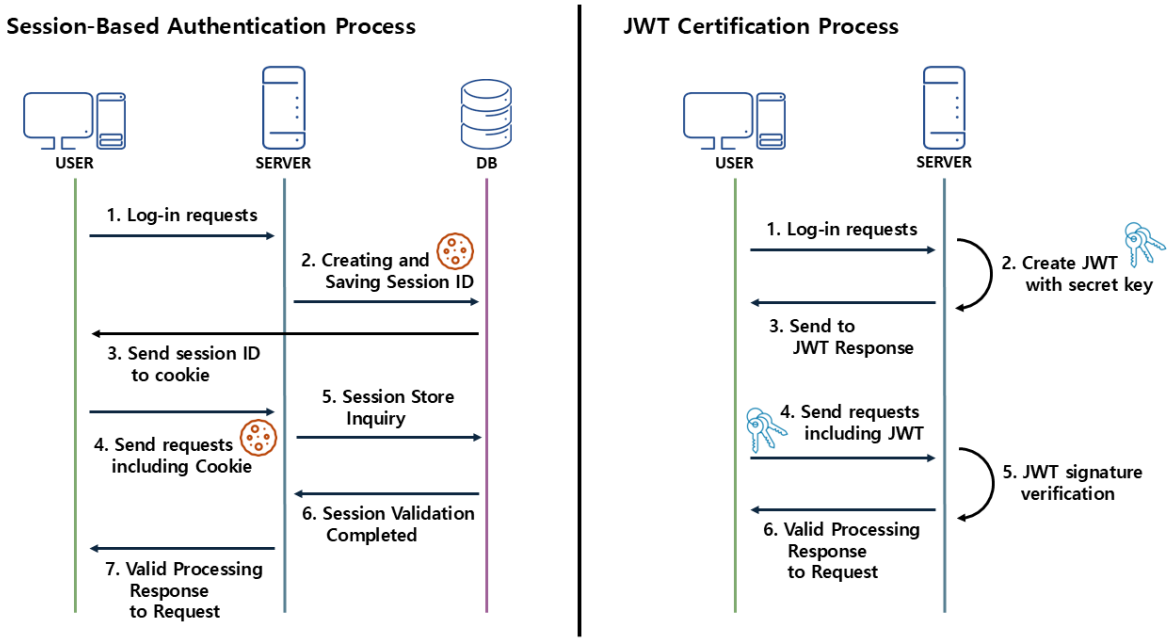
서명은 토큰의 무결성(Integrity) 및 신뢰성(Authenticity)을 보장하는 핵심 요소다. 서명은 인코딩된 헤더와 페이로드, 그리고 서버가 보관하는 비밀 키(Secret Key, HS256) 또는 사설 키(Private Key, RS256)를 결합해 지정된 알고리즘의 암호학적 서명 연산으로 계산된다. 서버는 서명을 재계산하여 수신된 서명과 일치하는지 확인하고, 이를 통해 토큰의 위변조 여부를 판단한다. 비밀 서명 키를 모르는 공격자는 유효한 서명을 생성할 수 없다.

```
HMACSHA256(
  base64UrlEncode(header) + "." + base64UrlEncode(payload),
  SECRET_KEY
)
```

[표 2] JWT의 서명 구성

2. 세션 기반 인증 vs JWT 인증

[그림 2]는 세션 기반 인증과 JWT 인증 방식을 구조화해 비교한 것이다.



[그림 2] 세션 기반 인증과 JWT 인증 비교

JWT 인증이 보편화된 이유는 세션 기반 인증의 단점을 해소하고 편리함을 제공하기 때문이다.

세션 기반 인증 방식

세션 기반 인증에는 크게 네 가지 단점이 있다.

- 서버가 세션 상태를 직접 관리해야 하는 상태 유지(Stateful) 구조
- 서버 수평 확장 시 세션 공유(공동 세션 스토어, Sticky Session 등) 필요
- 세션 저장소(Database, Redis 등)에 대한 추가 부하 및 운영 비용 발생
- 중앙 세션 저장소에 의존하기 때문에 마이크로서비스 및 분산 아키텍처 유연성 저하

JWT 인증 방식

반면, JWT 인증은 세션 기반 인증의 단점을 상당 부분 해소한다.

- 무상태(Stateless) 구조로 별도 세션 저장소 없이 서명 검증만으로 기본 인증 가능
- 서버/인스턴스 간 세션 동기화가 필요 없어 수평 확장에 유리
- 각 서비스가 토큰을 직접 검증할 수 있어, 마이크로서비스 아키텍처에 친화적
- 토큰에 사용자 식별자, 권한, 만료 시간 등을 포함하여 서버 상태에 대한 의존도를 줄임

주요 JWT 보안 취약점

JWT의 취약점은 주로 서명 검증 로직 구현에 오류가 발생하거나, 사용자가 조작 가능한 헤더(Header) 파라미터를 서버가 무분별하게 신뢰할 때 발생한다. JWT의 주요 장점인 무상태(Stateless) 특성은 운영 설계가 미흡할 경우 역설적으로 치명적인 약점이 된다.

전통적인 세션 기반 인증은 서버에서 세션을 삭제할 경우 강제 로그아웃이 즉시 반영된다. 하지만, JWT는 클라이언트가 토큰을 보관하므로 서버가 토큰을 즉시 회수하기 어렵다. 별도 철회 메커니즘이 없다면 토큰이 만료 시점까지 유효하게 유지될 수 있다. 특히, 직원의 권한이 변경되거나 계정이 비활성화되더라도 이전에 발급된 JWT가 만료 전이라면 API 접근이 가능할 수 있다. 계정 탈취를 탐지해 비밀번호를 재설정하더라도 공격자가 이미 확보한 JWT는 만료 전까지 유효하게 남을 수 있다.

서명 키가 유출되면 HS256과 같은 대칭키 방식이든 RS256과 같은 공개키 기반 서명 방식이든, 공격자는 서버와 동일한 서명 키 값을 사용해 형식상 완전히 정상적인 JWT를 직접 생성할 수 있다. 이렇게 위조된 토큰이 서명 검증을 그대로 통과하면 서버와 탐지 시스템은 해당 요청을 정상 트래픽으로 오인해 처리하게 되고, 그 결과 공격 행위에 대한 탐지가 실제 대량 정보 유출이 발생한 이후에야 뒤늦게 이루어질 수 있다. 이처럼, 제대로 관리되지 않은 JWT의 보안 취약점은 이커머스 플랫폼 개인정보 유출 사태에서도 비슷한 양상으로 나타남을 알 수 있다.

다음으로, JWT와 관련된 주요 취약점들을 살펴본다.

1. 서명 무결성 검증 우회 취약점

1) HS/RS 알고리즘 혼동 취약점 (CVE-2015-9235)

jsonwebtoken Node.js 4.2.2 미만 버전은 비대칭 알고리즘(RS/ES 계열)으로 검증해야 하는 상황에서도 대칭 알고리즘(HS 계열)으로 서명된 토큰을 허용하는 알고리즘 혼동 문제가 있다. 공격자는 서버가 검증에 사용하는 공개 키를 HMAC 비밀키처럼 악용해 임의의 페이로드로 토큰을 위조하고 인증을 우회하거나 권한을 상승시킬 수 있다.

2) alg=none 허용 취약점 (CVE-2021-22160 / CVE-2022-23540 / CVE-2025-61152)

JWT 구현 과정 중 일부는 alg: "none"이 포함된 토큰에 대해 서명을 검증하지 않거나, 검증 옵션 미지정 시 none을 사실상 허용하는 방식으로 동작해 서명 검증 우회를 유발한다. Apache Pulsar는 alg=none일 때 서명을 검증하지 않는 문제가 보고되었다. jsonwebtoken은 jwt.verify()에서 알고리즘

을 명시하지 않을 때 none 기본 처리로 검증 우회가 가능했으며, python-jose는 alg=none 토큰을 서명 검증 없이 허용하는 문제가 있었다.

3) JWT 클레임 위변조 취약점 (CVE-2022-39227)

python-jwt 3.3.4 이전 버전은 JWS Compact 직렬화와 JWS JSON 직렬화 처리 과정의 불일치로 인해 서명 검증을 통과한 값과 애플리케이션이 실제로 사용하는 클레임이 달라질 수 있다. 공격자는 비밀키(Secret Key)를 모르더라도 기존 토큰의 서명을 재사용하는 방식으로 클레임을 변조해 인증을 우회할 수 있다. 해당 취약점은 CVSS 9.1점(Critical)으로 평가되었다.

2. 알고리즘 혼동 공격

1) json-web-token 라이브러리 취약점 (CVE-2023-48238)

Node.js용 json-web-token 3.1.1 이전 버전은 검증에 사용할 알고리즘을 JWT 헤더의 alg 필드에서 추출하는 설계 상 결함이 있다. 서버가 RS256을 사용 중이고 공격자가 공개 키를 확보한 경우, alg를 HS256으로 설정하고 공개키를 HMAC 키처럼 사용하게 하여 개인 키 없이도 토큰을 위조할 수 있다.

2. cjwt 라이브러리 취약점 (CVE-2024-54150)

C 기반 JWT 라이브러리 cjwt의 v2.2.0은 토큰 검증 과정에서 서명 방식(HMAC vs RS/EC/PS)을 엄격히 구분하지 못해 알고리즘 혼동(algorithm confusion)이 발생할 수 있다. 공격자는 검증 로직이 토큰의 alg와 키 타입을 안전하게 강제하지 않는 구성에서, 공개 키를 HMAC 키처럼 취급하게 만들어 개인 키 없이도 위조 토큰이 검증을 통과하도록 유도할 수 있다. 해당 취약점은 v2.3.0에서 수정됐다.

3. 키 검색 매개변수 주입 공격

1) kid 파라미터 경로 순회 및 SQL 인젝션

서버가 kid 값을 파일 경로나 데이터베이스 조회 쿼리에 직접 연결해 사용하는 경우, 공격자는 ../../dev/null 같은 경로 순회 문자열이나 SQL 구문을 주입할 수 있다. 이를 통해, 임의의 키를 선택하게 만들거나 키 조회 로직을 교란할 수 있다. 결과적으로 잘못된 키로 검증이 진행되거나 키 저장소 정보가 노출될 수 있다.

4. 키 관리 실패 취약점

1) 하드코딩된 비밀 키 (CVE-2025-7079 / CVE-2025-6950)

서명 키가 코드나 펌웨어에 하드코딩된 경우 공격자는 해당 값을 확보해 임의의 JWT를 위조하고 인증을 우회할 수 있다. CVE-2025-7079는 bluebell-plus의 jwt.go 파일에 'bluebell-plus' 문자열이 하드코딩된 사례다. CVE-2025-6950는 Moxa 네트워크 보안 장비 및 라우터에서 JWT 서명용 하드코딩 키 사용이 보고된 사례다.

2) iss 클레임 형식 위반 (CVE-2025-30144)

fast-jwt 5.0.6 이전 RFC 7519 관점에서 문자열이어야 하는 iss 클레임에 문자열 배열을 허용하는 검증 결함이 존재한다. 공격자는 정상 발행자와 악성 발행자를 혼합한 iss 배열을 구성해 검증 로직을 우회할 수 있다.

3) 서명 키 로테이션 및 폐기 실패에 따른 내부자 악용

조직이 JWT 서명 키를 장기간 운영하는 가운데, 담당자 변경 및 권한 조정 시 키 로테이션 또는 폐기 절차를 수행하지 않으면, 유출된 키로 유효한 JWT가 지속적으로 생성될 수 있다. 서명이 정상 검증되는 한 서버와 관제는 이를 정상 트래픽으로 오인할 수 있다. 이로 인해, 탐지가 지연되고 사고가 외부 신고나 조사 이후에 식별되는 상황이 발생할 수 있다.

탐지 및 보안 전략

조직들이 내부 키 유출 포함 모든 JWT 공격 벡터에 대응하기 위해서는 서버 측 검증 로직을 강화하고, 실시간 침해 탐지 체계를 구축해야 한다.

1. 안전한 키 관리 시스템 구축 및 HS256 대체 전략

1) 하드웨어 보안 모듈 (HSM) 또는 클라우드 KMS 활용

내부자 위협 대응의 핵심은 서명 키 자료를 애플리케이션 서버의 파일 시스템이나 메모리에서 격리하는 것이다. 서명 키(비밀 키/개인 키)는 하드웨어 보안 모듈(Hardware Security Module, Module) 또는 클라우드 기반 키 관리 서비스(Key Management Service)에 저장하고, 애플리케이션 서버는 키 값을 직접 취급하지 않고 서명 연산 API를 호출하는 방식으로 사용해야 한다. 이러한 방식은 내부자가 서버에 접근하더라도 키 자료를 획득하는 것을 물리적으로 차단한다

.

2) 비대칭키(RS256)로의 전환 및 키 로테이션

내부자 위협과 키 노출 리스크를 낮추기 위해 대칭키(HS256) 단일 공유 구조보다 개인 키/공개 키 구조의 비대칭키(RS256) 사용을 우선시해야 한다. 개인 키는 HSM/KMS 내부에 보관하고, 공개 키는 외부에 배포하더라도 개인 키의 기밀성이 유지되는 한 공격자가 유효한 토큰을 위조할 수 없다. 또한, 침해 가능성을 전제로 키 수명과 교체 주기를 정의하고, 주기적으로 키를 로테이션 및 폐기하는 운영 절차를 마련해야 한다.

2. 서버 측 엄격한 토큰 검증 구현

개발자는 JWT 라이브러리를 사용할 때, 표준 스펙에서 허용하는 모든 잠재적 위험을 명시적으로 차단해야 한다.

1) 알고리즘 화이트리스트 강제화

JWT 검증 시, 서버가 허용하는 알고리즘(예: RS256)을 라이브러리에 명시적으로 고정해야 한다. 이를 통해, alg=none 같은 안전하지 않은 알고리즘을 허용하지 않고, 알고리즘 혼동 공격 시도를 무력화한다.

2) 핵심 클레임에 대한 엄격한 검증

exp(만료 시간), iss(발급자), aud(수신자) 등 핵심 클레임을 철저하게 검증해야 한다. 접근 토큰(Access Token)을 수신할 때 exp 유효성 검증을 통해 만료된 토큰 사용을 차단해야 한다.

3) 키 참조 매개변수 방어

kid, jku, jwk 같은 동적 키 참조 메커니즘은 주입 공격의 빌미가 될 수 있어 사용을 지양해야 한다. 불가피하게 사용하는 경우, kid 값에 Path Traversal 패턴(../) 및 SQL Injection 패턴 등 강력한 화이트리스트 기반 입력 필터링을 적용해야 한다.

3. 토큰 수명 관리 및 무효화 전략

1) 접근 토큰(Access Token) 수명 관리

접근 토큰은 탈취 시 피해 범위 최소화를 위해 만료 시점(exp)을 수 분 이내로 짧게 해야 한다. 장기적인 세션 유지는 리프레시 토큰(Refresh Token)으로 분리한다. 리프레시 토큰은 재사용 방지를 위한로테이션, 일회성 사용 등의 정책 적용이 필요하다.

2) 실시간 토큰 무효화(Revocation) 시스템

키 유출이나 강제 로그아웃 발생 시, 즉각적인 토큰 무효화를 위한 중앙 집중식 무효화 리스트(Revocation List)를 구축한다. 각 API 요청은 토큰의 jti 클레임을 기준으로 리스트를 조회해, 목록에 있으면 요청을 차단한다. 무효화 리스트 항목은 TTL(Time To Live) 형태로 자동 만료되도록 관리한다.

4. 침해 탐지 및 모니터링 체계 구축

1. 상세 로깅 및 SIEM 연동

JWT 검증 실패 이벤트(Invalid Signature, Algorithm Mismatch, Claim Errors 등)를 상세히 기록하고, 이를 중앙 집중형 SIEM 시스템에 연동해 관리 및 대응한다.

2. 이상 징후 자동 탐지 시스템 구축

단기간 내 동일한 토큰의 과도한 API 호출, 지리적으로 비정상적인 접근, 토큰 만료 시간(exp)이 비정상적으로 긴 토큰 유입 등 다양한 이상 징후가 발생할 수 있다. 이러한 이상 징후를 즉각적으로 탐지하고 알림(alert)을 발생시키는 시스템을 구축해야 피해 확산을 막을 수 있다.

결론

대형 이커머스 플랫폼 사건과 같이 JWT 기반 인증 시스템의 보안이 제대로 관리되지 않으면, 기업의 핵심 자산이 광범위하게 침해될 수 있다. 특히, 서명 키 관리 실패는 인증 체계 전반을 무력화하는 단일 실패 지점(Single Point of Failure)이 되기도 한다.

따라서, 조직들은 서명 키를 HSM/KMS 내부에 격리하고, 비대칭키(RS256) 및 키 로테이션 정책을 우선적으로 적용해야 한다. 동시에 알고리즘 고정, 핵심 클레임 검증, 키 참조 입력 방어를 포함한 엄격한 검증 로직을 구현해 공격 성공 가능성을 최대한 낮춰야 한다. 마지막으로 로그 기반 탐지와 이상 징후 자동화를 통합한 보안 체계를 구축하여 침해 징후를 조기에 식별하고 피해를 최소화할 수 있어야 한다.

분석 보고서
JWT 인증의 명과 암

㈜안랩

경기도 성남시 분당구 판교역로 220 (우) 13493

홈페이지 : www.ahnlab.com

대표전화 : 031-722-8000 | 구매문의 : 1588-3096 | 팩스 : 031-722-8901

AhnLab