통 합 구 현

REST 보안

한국기술교육대학교
온라인평생교육원

학습내용

- 인증(Authentication)
- 인가(Authorization)

학습목표

- REST API에 적용되는 인증 방법을 사용할 수 있다.
- REST API에 적용되는 인가 방법을 설명할 수 있다.

1. 인증(Authentication)

1) API 보안 관점 및 인증

(1) API 보안 관점

① 정의

인증(Authentication)

- API를 호출하는 대상(단말, 서버, 사용자 등)을 확인하는 과정

인가(Authorization)

- 해당 리소스에 대하 사용자가 그 리소스를 사용할 권한이 있는지 확인하는 과정

네트워크레벨 암호화

- 네트워크 프로토콜 수준에서 암호화 진행 (예 : HTTPS)

메시지 무결성 보장

- 외부 요인에 의해서 중간에 변조가 되지 않게 방지하는 방법으로 메시지 시그니처(Signature)를 생성해서 메시지와 같이 전송

메시지 본문 암호화

- 네트워크 레벨 암호화를 사용할 수 없거나 신뢰할 수 없는 상황에서 메시지 자체를 암호화 하는 방법

- API를 호출하는 대상을 확인하는 절차

② 구현 방식

API 키 방식 | API 토큰 방식 | Claim 기반 토큰 방식 (JWT- JSON Web Token)

1. 인증(Authentication)

2) API 키 방식

(1) 개요

① API 키의 정의

API 키

- **특정 사용자만 알 수 있는 일종의 문자열**

② 사용방식

- 개발자가 API 사용할 시 API 제공사의 포탈 사이트에서 API 키를 발급받고 API 호출할 때 API 키를 메시지 안에 넣어 호출

③ 한계점

- 모든 클라이언트가 같은 API 키를 공유하기 때문에 높은 보안 인증이 필요할 시에는 권장하지 않음

1. 인증(Authentication)

3) API 토큰 방식

(1) 정의

① API 토큰 방식의 정의

API 토큰 방식

- **ID, 비밀번호 등으로 사용자 인증 후, 유효 기간이 있는 API 토큰을 발급해 API 토큰으로 사용자를 인증하는 방식**

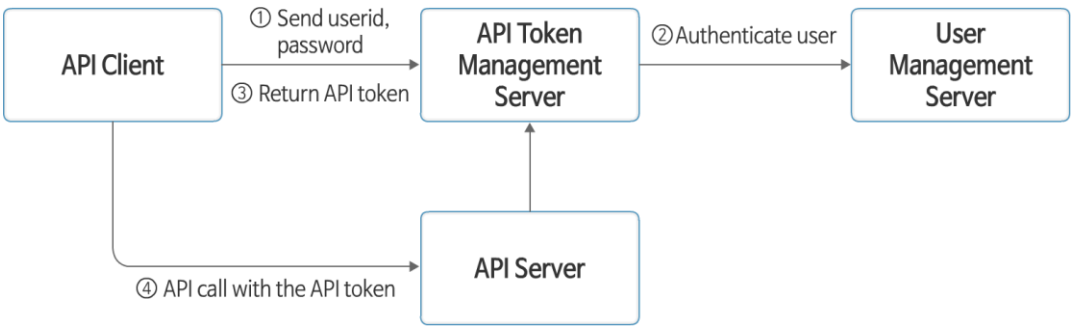
- 사용자 ID와 비밀번호를 대신해 API 토큰을 보내면서, 보안상 사용자 계정 정보를 탈취당할 가능성을 줄임

1. 인증(Authentication)

3) API 토큰 방식

(2) 일반적인 토큰 생성 과정 및 사용 흐름

- API 클라이언트가 사용자 ID, 비밀번호를 보내 API 호출을 위한 API 토큰을 요청
- API 인증 서버는 사용자 ID, 비밀번호를 바탕으로 사용자를 인증
- 인증된 사용자에 대해 API 토큰을 발급 (유효 기간을 가짐)
- API 클라이언트가 해당 토큰으로 API를 호출하면, API 서버는 API 토큰이 유효한지를 서버에 문의해 유효할 시 API 호출을 받아들임



[일반적인 API 토큰 생성 요청 및 API 토큰 사용 흐름]

- 1단계의 사용자 인증 단계에서 보안 수준에 따라 여러 방식 사용

- HTTP Basic Auth
- Digest Access Authentication
- 클라이언트 인증 추가
- 제3자 인증 방식(Oauth 2.0 Authorization grant type)
- IP 화이트 리스트를 이용한 터널링
- Bi-directional Certification(Mutual SSL)

1. 인증(Authentication)

3) API 토큰 방식

(3) HTTP Basic Auth

HTTP Basic Auth

- 가장 기본적이고 단순한 형태의 인증 방식

- 사용자 ID와 비밀번호를 HTTP 헤더에 Base64 인코딩 형태로 넣어 인증 요청

```
Authorization: Basic VGVYcnk6aGVsbG8d29ybdDsq=
```

- 중간에 패킷을 가로채 Base64로 디코딩하면 그대로 ID와 비밀번호가 노출되기 때문에 반드시 HTTPS 프로토콜을 사용해야 함

1. 인증(Authentication)

3) API 토큰 방식

(4) Digest Access Authentication

Digest Access Authentication

- HTTP Basic Auth를 보강한 인증 프로토콜

- 클라이언트가 인증을 요청할 때 클라이언트가 서버로부터 nonce라는 일종의 난수 값을 받고, 사용자 ID와 비밀번호를 이 난수 값을 이용해 해시화하여 서버로 전송하는 방식
- 해시 알고리즘을 알고있다 하더라도 비밀번호를 역추적하기가 어려워 Basic Auth 방식보다 향상된 방식
  - 해시 알고리즘인 MD5는 보안등급이 낮아, 최소 SHA-1, SHA1-244, SHA1-256 이상의 해시 알고리즘 권장
  - MD5를 사용하는 경우 추가적인 보안(HTTPS) 로직을 겸비해 사용하거나, 다른 인증 메커니즘 사용 권장

① 흐름

- 클라이언트가 서버에 특정 리소스 /car/index.html 요청
- 서버는 해당 세션에 대한 nonce 값을 생성해 저장하고 클라이언트에 반환하며, 이때, realm을 같이 반환
- 클라이언트는 앞에서 서버로부터 받은 realm과 nonce값으로 해시(MD5) 값을 생성

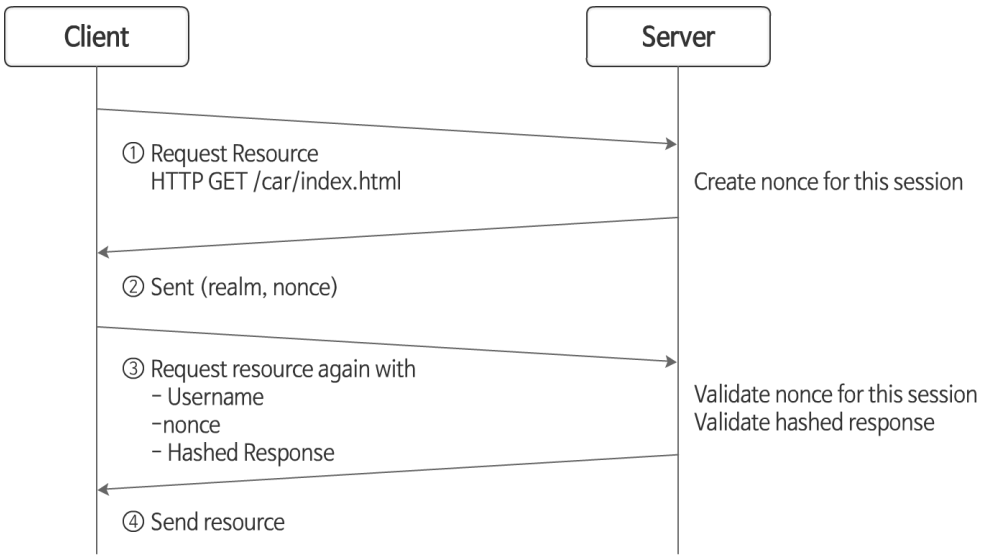
```
HA1 = MD5(사용자 이름 : realm : 비밀번호)
HA2 = MD5(HTTP method : HTTP URL)
  response hash = MD5(HA1 : nonce : HA2)
```

- 위에서 전달된 nonce값과 저장된 nonce 값이 같은지 비교 후, 응답 해시 값을 계산해 같은지 비교하고 같으면 해당 리소스 반환
  - 두번째에서 저장을 위한 HTTP 세션을 사용하거나, 서버간에 공유 메모리 (Memcached, Redis 등)을 넣어 서버 간 상태 정보 유지

1. 인증(Authentication)

3) API 토큰 방식

(4) Digest Access Authentication



[Digest Access Authentication 흐름]

(5) 클라이언트 인증 추가

- 추가적인 보안 강화 위해 사용자 인증에 더해 클라이언트 인증 방식 추가
- 사용자 ID, 비밀번호 외에 Client ID, Client Secret을 같이 입력 받도록 함
  - Client ID : 특정 앱에 대한 등록 ID
  - Client Secret : 특정 앱에 대한 비밀번호

1. 인증(Authentication)

3) API 토큰 방식

(6) 제3자 인증 방식

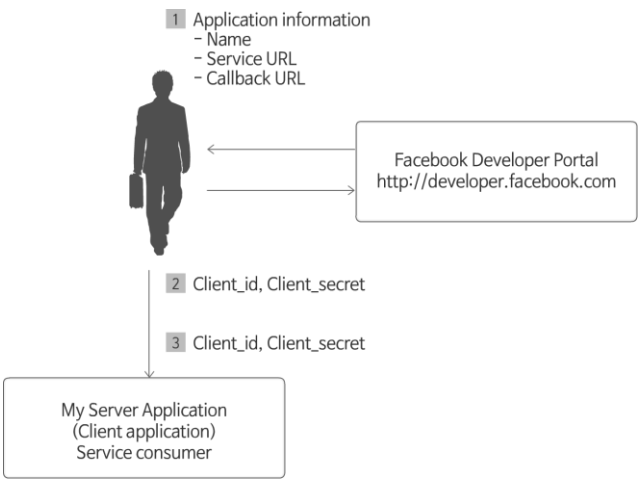
제3자 인증 방식

- **페이스북이나 트위터와 같은 API 서비스 제공자들이 파트너 애플리케이션에 적용하는 방법으로, 개발자가 만든 애플리케이션 서비스를 페이스북이나 트위터 계정을 이용해 인증하는 방법 (사용자인증을 타 서비스 사용자 계정 인증으로 대체)**

- 애플리케이션 서비스에 대해 해당 사용자가 페이스북 서비스 사용자라는 것을 인증하지만, 페이스북 사용자의 비밀번호는 노출되지 않음
- OAuth 2.0에서 제3자 인증, 직접인증 등 다양한 시나리오를 제공하고 있음

① 페이스북 기반으로 3자 인증과정

- 토큰 발급 과정
  - 페이스북 개발자 포털에 접속해, 인증을 사용하고자 하는 애플리케이션 정보를 등록(서비스명, 서비스URL, 인증 성공 시 성공 정보 받을 콜백 URL)
  - 해당 애플리케이션에 대한 client_id, client_secret 발급(클라이언트 인증)
  - 개발하고자 하는 애플리케이션에 client_id와 client_secret을 넣고 인증 페이지 정보를 넣어 애플리케이션 개발 진행



[페이스북 제3자 인증 방식 API 토큰 요청 흐름]

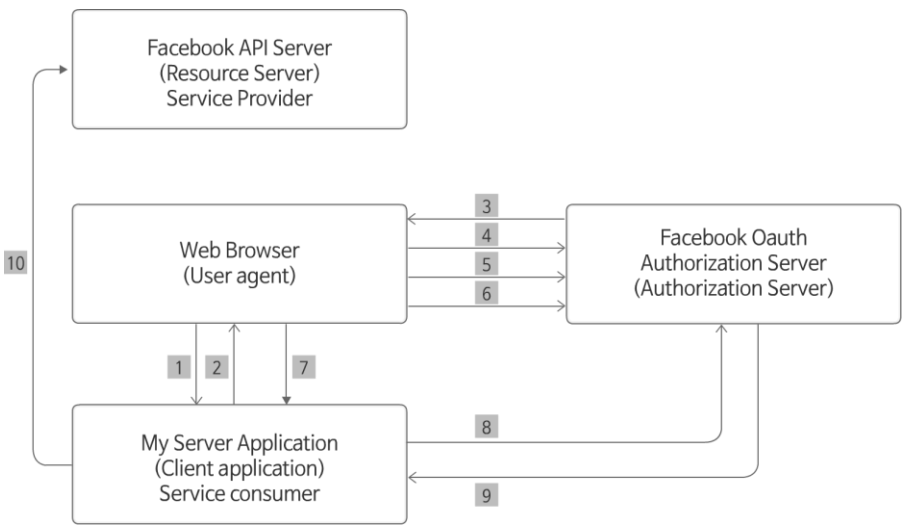
1. 인증(Authentication)

3) API 토큰 방식

(6) 제3자 인증 방식

① 페이스북 기반으로 3자 인증과정

• 사용자 인증 진행 과정
- 사용자가 애플리케이션 서비스 접근 요청
- 사용자 인증 되지 않았기 때문에 페이스북 로그인 URL을 HTTP 리디렉션으로 브라우저에 보냄
  → 이 때, URL에는 해당 애플리케이션에 대한 인증 요청임을 알려주고자 client_id 등의 추가 정보와 정보 접근 권한을 scope라는 필드를 통해 요청
- 페이스북 로그인 페이지로 이동해 ②에서 받은 추가 정보들과 함께 로그인 요청
- 페이스북이 사용자에게 로그인 창 전송
- 사용자가 페이스북의 ID와 비밀번호 입력
- 페이스북이 사용자 인증하고, 인증 관련 정보와 함께 브라우저로 해당 애플리케이션의 로그인 완료 페이지로 리디렉션을 요청
- 애플리케이션이 인증 관련 정보를 받음
- 애플리케이션은 해당 정보를 가지고 페이스북에 제대로 인증 받은 | 사용자인지 문의
- 페이스북은 인증된 사용자임을 확인 후 API Access Token 발급
- 애플리케이션은 해당 토큰으로 페이스북 API에 접근



[제3자 인증 방식을 이용한 API 인증 흐름]

1. 인증(Authentication)

3) API 토큰 방식

(7) IP 화이트 리스트를 이용한 터널링

- 서버 간의 통신이나 자사 서버 간의 통신에서 특정 API URL에 대해 들어오는 IP 주소를 화이트 리스트로 유지
- API 서버 앞단에 HAProxy나 아파치와 같은 웹 서버를 배치해 특정 URL로 접근할 수 있는 IP 목록을 제한하거나, 방화벽 자체에 들어올 수 있는 IP 제한

(7) Bi-directional Certification(Mutual SSL(Secure Socket Layer))

- 보통은 HTTPS 통신 사용시 서버에 공인 인증서를 놓고 단방향의 SSL을 제공하는 방식이지만, 이 방식은 클라이언트에도 인증서를 두고 양방향으로 SSL을 제공
- 일반 서비스에는 사용되지 않으며, 높은 인증 수준이 요구될 시 사용
  - [참고사이트]
    http://www.codeproject.com/Articles/326574/An-Introduction-to-Mutual-SSL-Authentication

1. 인증(Authentication)

4) Claim 기반 토큰 방식

(1) 정의

Claim 기반 토큰 방식

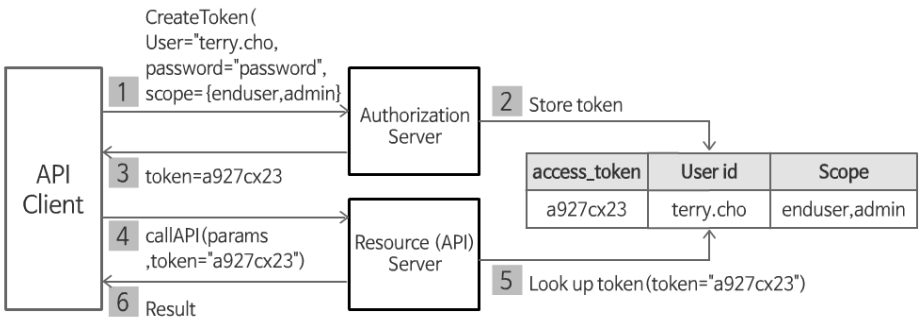
- **Claim이라는 사용자에 대한 프로퍼티나 속성을 말하며, 토큰 자체가 정보를 가지는 방식**

- Claim 기반의 토큰 방식인 JWT(JSON Web Token) 표준이 많이 사용
- XML 기반의 SAML 2.0도 Claim 기반 토큰 방식이나 크기가 크고 무거움 (표준화가 잘 되어 있어 현재에도 사용되기는 함)
- 토큰 생성하는 단계에서 토큰을 별도로 서버에 저장할 필요가 없고, 사용자 정보를 별도로 계정 시스템 등에서 조회할 필요가 없어 구현이 단순해짐

1. 인증(Authentication)

4) Claim 기반 토큰 방식

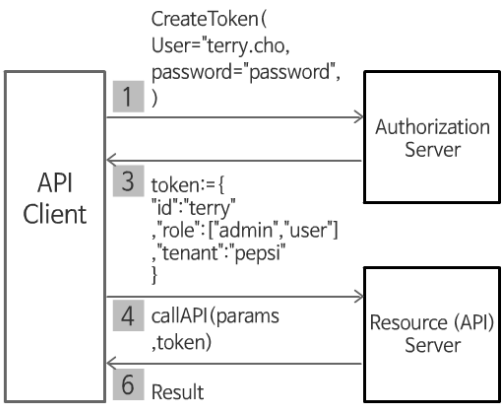
(2) Oauth 토큰 방식과의 비교



[String 토큰에 의한 API 권한 인증 흐름]

Oauth

- Access_token이라는 랜덤 문자열을 발급해 사용
- 서버에 토큰을 저장해두고 정보를 검색하며 사용



[Claim 기반의 토큰에 의한 API 권한 인증 흐름]

Claim

- Claim이라는 사용자에 대한 프로퍼티나 속성 사용
- 토큰 자체가 정보를 가짐

1. 인증(Authentication)

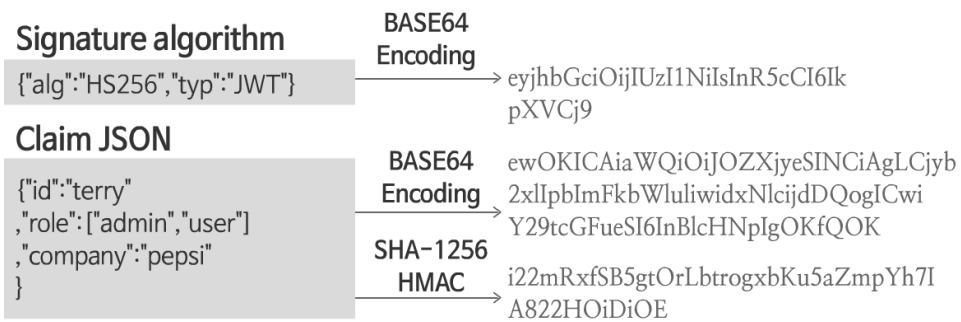
4) Claim 기반 토큰 방식

(3) JWT 구성

① 메시지 포맷은 다음과 같으며, 각각의 요소를 ‘ . ‘으로 이어 붙여 토큰을 생성함

```
{서명 방식을 정의한 JSON을 이용한 Base64 인코딩}.{JSON
Claim을 Base64 인코딩}.{JSON Claim에 대한 서명}
```

- Claim(메시지) 정의 : Claim JSON 문자열을 Base64 인코딩을 통해 하나의 문자열로 변환
- 변조 방지 : 무결성을 보장하는 방법으로 서명
- 서명 생성 방식 : SHA1-256, 384, 512, 공인 인증서 이용한 RS256 등 다양한 서명 알고리즘 지원



[JWT 토큰 구조]

- 위의 예를 JWT 구성에 맞추어 만든 토큰의 결과

```
{서명 방식을 정의한 JSON을 이용한 Base64 인코딩}.{JSON
Claim을 Base64 인코딩}.{JSON Claim에 대한 서명(HMAC)}
```

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ew0KICAiaWQiOiJ0ZXJyeSINCiAgLCJy
b2xlIjpbImFkbWluIiwidXNlciJdDQogICwiY29tcGFueSI6InBlcHNpIg0KfQ0K.i22m
RxfSB5gt0rLbtrogxbKj5aZmpYh7lA82HO1Di0E
```

1. 인증(Authentication)

4) Claim 기반 토큰 방식

(4) JWT 라이브러리가 있는 프로그래밍 언어

- Java
- Python
- Node.js
- Ruby
- PHP
- .NET
- Haskell 등

(5) JWT 문제점

- 토큰의 길이가 김
  - Claim 에 넣는 데이터 양이 많아질수록 JWT 토큰의 길이가 길어져, 대역폭 낭비를 가져옴
- 발급된 토큰에 대한 수정 및 폐기 어려움
  - 토큰 내에 모든 정보를 가지고 있기 때문에 정보를 수정할 수 없음. 만료 시간(Expire time)을 명시적으로 두고 Refresh token 등을 이용해 중간마다 토큰을 재발행 해야 함
- 암호화를 하지 않아 토큰 노출에 취약
  - 기본적으로 Claim에 대한 정보를 암호화 하지 않아, 중간에 토큰을 취득하는 경우 사용자 정보가 누출될 수 있음
  - 특히, 자바스크립트 기반의 웹 클라이언트는 토큰 노출에 취약함. 보완하기 위해 토큰 자체를 암호화하는 JWE(JSON Web Encryption)을 사용함

2. 인가(Authorization)

1) 개요

(1) 정의

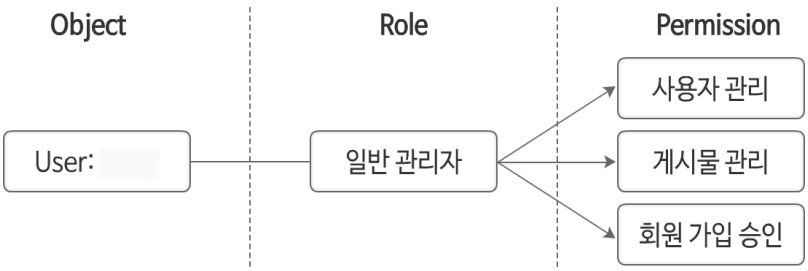
인가(Authorization)

- 사용자가 인증을 통해 로그인을 한 후, 해당 API에 대한 호출 권한이 있는지 확인하는 과정

2. 인가(Authorization)

2) 구현 방식 - RBAC/ACL

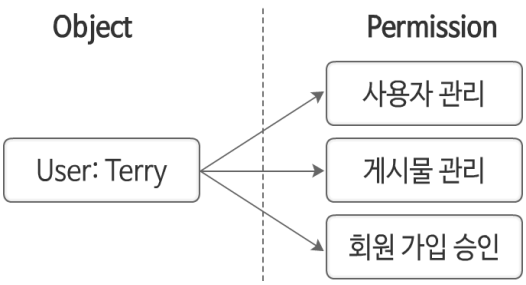
(1) RBAC 방식



[Object, Role, Permission의 상관 관계 예]

- Role Based Access Control
- 정해진 역할(Role)에 권한(Permission)을 연결해 두고, 해당 역할을 가진 사용자나 그룹에게 해당 권한 부여
- 사용자 역할을 기반으로 해 사용하기 쉬움
  - Object : 권한 부여의 대상이 되는 사용자나 그룹
  - Permission : 개별 권한
  - Role : 사용자 역할

(2) ACL 방식

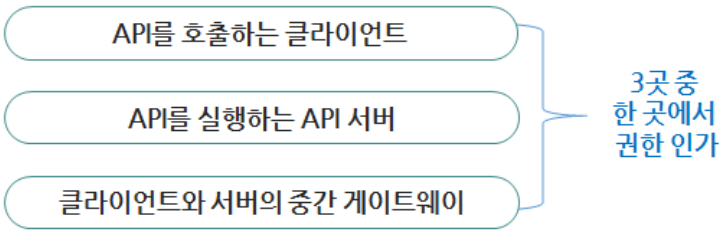


[Object와 Permission의 예]

- Access Control List
- 사용자에게 직접 권한 부여(RBAC의 role이라는 중간 매개체가 없음)

2. 인가(Authorization)

3) API 권한 인가 처리 위치



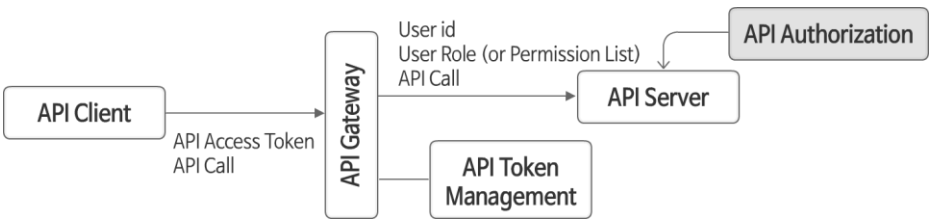
- 모바일 클라이언트, 자바스크립트 기반의 웹 클라이언트 등 다양한 클라이언트가 지원되며 점차 서버 쪽에서 권한 인가가 주로 이루어지게 됨
- API 서버의 비즈니스 로직에서 권한 처리를 하는 것이 일반적임

2. 인가(Authorization)

4) 서버에 의한 권한 인가 처리

(1) API 서버의 비즈니스 로직 단에서 권한 처리

- 각 비즈니스 로직에서 API 메시지를 각각 파싱하기 때문에 API 별로 권한 인가 로직을 구현하기 용이
- API Access Token을 통해 API를 호출할 경우 API Gateway가 이 Token을 권한 인가에 필요한 사용자 ID, Role 등으로 변환해 서버에 전달하면 헤더의 내용만을 이용해 API 권한 인가 처리가 가능
- 별도의 데이터베이스를 검색하지 않아도 됨



[API Server를 이용한 권한 인가 처리]

핵심정리

1. 인증(Authentication)

- API 보안 관점
  - 인증(Authentication)
  - 인가(Authorization)
  - 네트워크레벨 암호화
  - 메시지 무결성 보장
  - 메시지 본문 암호화

- 인증 정의
  - API를 호출하는 대상을 확인하는 절차

- API Key 방식
  - API Key : 특정 사용자만 알 수 있는 일종의 문자열
  - 개발자가 API 사용할 시 API 제공사의 포탈 사이트에서 API 키를 발급받고 API 호출할 때 API 키를 메시지 안에 넣어 호출

- API Token 방식
  - ID, 비밀번호 등으로 사용자 인증 후, 유효 기간이 있는 API 토큰을 발급해 API 토큰으로 사용자를 인증하는 방식
  - 구현 방식 : HTTP Basic Auth, Digest Access Authentication, 클라이언트 인증 추가, 제3자 인증 방식(Oauth 2.0 Authorization grant type), IP 화이트 리스트를 이용한 터널링, Bi-directional Certification(Mutual SSL)

- Claim 기반 Token 방식(JWT)
  - Claim이라는 사용자에 대한 프로퍼티나 속성을 말하며, 토큰 자체가 정보를 가지는 방식
  - 토큰 구조 : {서명 방식을 정의한 JSON을 이용한 Base64 인코딩}.{JSON Claim을 Base64 인코딩}.{JSON Claim에 대한 서명(HMAC)}

핵심정리

2. 인가(Authorization)

- 인가 정의
  - 사용자가 인증을 통해 로그인을 한 후, 해당 API에 대한 호출 권한이 있는지 확인하는 과정

- 구현 방식
  - RBAC(Role Based Access Control)
  - ACL(Access Control List)

- 서버에 의한 권한 인가
  - 비즈니스 로직에서 API 메시지를 각각 파싱하여 처리