# OAuth 2.0 개요 및 보안 고려사항

(보안연구부 보안기술팀 / 2015.12.14.)

## □ 개요

○ 핀테크 활성화의 일환으로 금융권의 오픈 API 제공을 위한 핀테크 오픈플랫폼 구축이 추진됨에 따라, 금융정보 제공 시 안전한 인증 방법으로 OAuth 2.0 및 관련 보안기술에 대한 관심 급증

○ 이에 OAuth 2.0의 개념을 알아보고 금융권에서 도입 시 고려해야 할 보안 사항을 조사

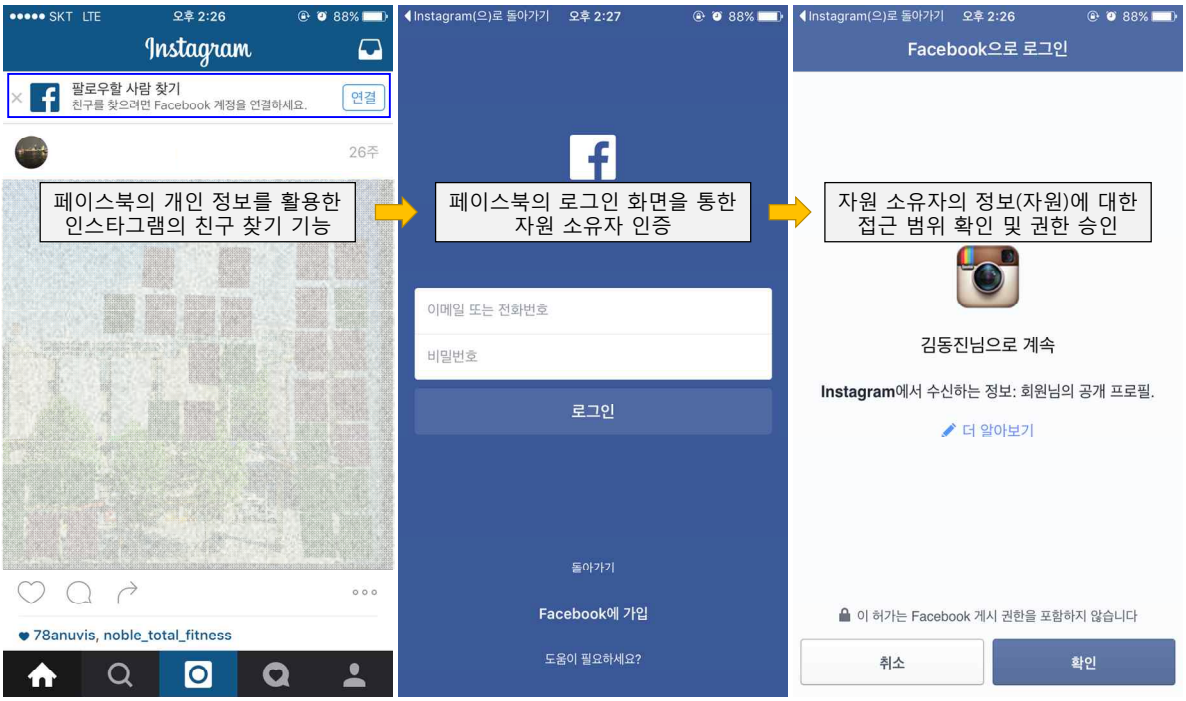
※ OAuth 2.0의 기술 상세 내용은 [붙임] 참조

## □ OAuth(Open Authorization) 2.0의 개념

○ **(정의)** 제3의 앱이 자원의 소유자인 서비스 이용자를 대신하여 서비스를 요청할 수 있도록 자원 접근 권한을 위임하는 방법

예시) 인스타그램은 SNS 친구 찾기를 위해 페이스북의 오픈 API를 사용하여 자원 소유자의 친구 목록에 접근하며, 이때 OAuth 2.0 기반으로 인증 및 권한 승인

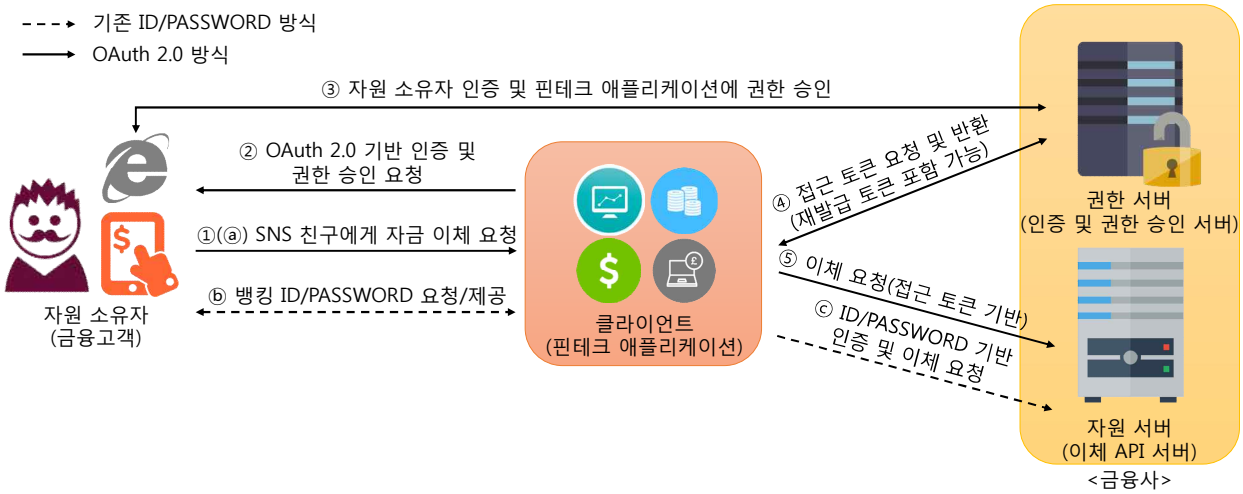
< OAuth 2.0 기반 오픈 API 사용 예 : 인스타그램 및 페이스북 >



○ **(금융거래 적용)** OAuth 2.0을 이용하여 금융고객(자원소유자)의 뱅킹 ID/PASSWORD를 핀테크 애플리케이션에 직접적으로 제공하지 않고, 접근 토큰(Access Token)*을 기반으로 계좌 이체 등에 대한 권한을 위임

* 자원에 대한 접근 권한을 자원 소유자가 인가하였음을 나타내는 자격증명(Credentials)

< OAuth 2.0의 권한 승인 과정 개요(예 : 자금 이체) >




※ OAuth 2.0 기반 권한 승인을 위해 클라이언트인 핀테크 애플리케이션을 권한 서버에 사전 등록 필요

## □ OAuth 2.0의 대표 보안 위협[1)]

### ○ 통신구간 보안위협

- **(중간자 공격)** 클라이언트의 자격증명*을 네트워크 단에서 도청 및 변조함으로써 정상 클라이언트로 위장하여 보호된 자원에 불법 접근

  * 권한 승인 방식에 따라 접근 토큰 외에도 재발급 토큰(Refresh token) 및 권한 코드(Authorization code)가 존재

  · 2014년 초에 발표된 은닉 리다이렉트(Covert redirect)[2)]는 중간자 공격 위협으로, 보호된 자원에 불법 접근 및 유출 가능함이 검증됨

### ○ 클라이언트 유형별 보안 위협

- **(웹서버)** 클라이언트가 웹 서버일 경우 모든 서비스 이용자의 접근 토큰을 저장 관리하고 있어 악의적인 공격에 의한 토큰의 탈취 위협

1) OAuth 2.0에 대한 세부 보안 위협, 대응 방안 및 보안 고려 사항은 RFC-6819 : OAuth 2.0 Threat Model and Security Considerations 참조

2) http://oauth.net/advisories/2014-1-covert-redirect/
http://www.tetraph.com/blog/covert-redirect/covert-redirect-vulnerability-related-to-oauth-2-0-and-openid/

- **(이용자 단말)** 악성코드 감염 단말기에서 자격증명이 공격자에게 노출 가능 및 권한서버와 자원 서버에 대한 서비스 거부 공격 가능

○ **자원 소유자 대상 보안 위협**

- **(파밍 등)** 파밍(Pharming) 또는 클릭재킹(Clickjacking)*을 통해, 자원 소유자의 인증 정보 탈취 및 권한 승인 유도

* 인터넷 이용자에게 투명한 악성 웹 페이지와 정상 웹 페이지를 겹쳐서 보여줌으로써 공격자가 의도한대로 마우스 클릭을 유도하는 공격

## □ 결론 및 보안 고려사항

○ 오픈 API의 기본이 되는 OAuth 2.0에 대한 다양한 보안 위협이 존재하므로, 도입 시 보안성 검토 및 보안대책의 수립 필요

- **(보안성 강화)** 권한 및 자원 서버에서 클라이언트의 요청을 처리 시, OAuth 2.0에 정의된 클라이언트에 대한 검증 절차 강화
  - 등록된 정보와 접근 토큰 요청 시 제출된 클라이언트 정보 비교, 클라이언트의 IP 주소 및 기타 시스템 정보 비교
  - 클라이언트 유형 및 권한 승인 방식 중 비교적 안전한 웹 애플리케이션 및 권한 코드 승인 방식에 대한 우선 적용 고려
  - 접근 토큰의 불법 유출 및 악용에 대비하여 접근 토큰의 유효 기간을 짧게 설정(또는 일회용)하고, 재발급 토큰과 분리 저장하는 방안 고려
- **(관리대책 수립)** 클라이언트에 대한 신뢰성 확보를 위한 등록 절차를 마련하여 관련 서류나 현장 실사 등 적정성 검증

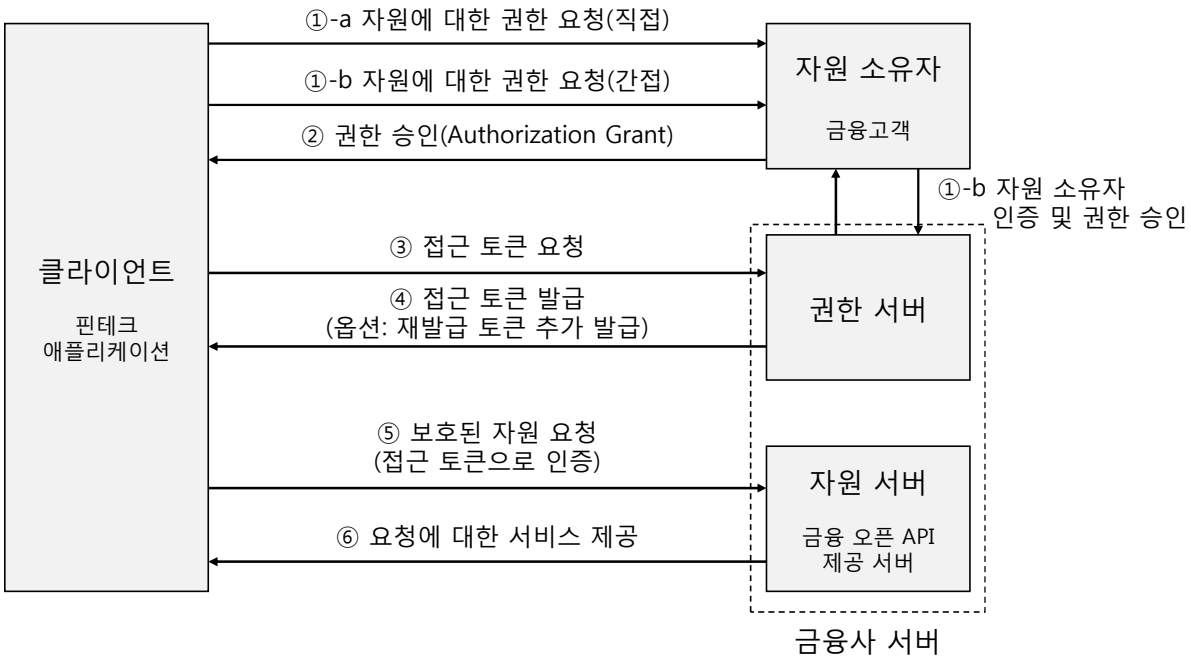
[붙임]

# OAuth 2.0 개요 및 보안 고려사항 (세부 내용)

## □ OAuth(Open Authorization) 2.0의 권한 승인 개요

○ OAuth 2.0의 권한 승인 과정은 6단계로 요약됨

< OAuth 2.0의 권한 승인 과정 요약 >




| 단계 | 세부 내용 |
|---|---|
| **① 단계** | 클라이언트가 자원 소유자에게 자원에 대한 접근 권한을 직/간접 요청<br>· **①-a** : 자원 소유자에게 직접 요청<br>· **①-b** : 권한 서버를 중개자로 간접적으로 요청(권장)*<br>* 리다이렉트(Redirect) URI를 통해 자원 소유자를 권한 서버로 리다이렉션 시키며, 권한 서버가 클라이언트를 식별할 수 있도록 식별 정보를 함께 전달 |
| **② 단계** | 클라이언트에게 자원에 대한 접근 권한을 승인(4가지 권한 승인 방법 정의) |
| **③ 단계** | 클라이언트가 권한 서버에게 접근 토큰을 요청 |
| **④ 단계** | 권한 서버는 클라이언트를 인증 및 부여 받은 권한을 검증하고, 검증된 경우 클라이언트에게 접근 토큰을 발급<br>· 권한 승인 방법에 따라, 선택적으로 접근 토큰의 유효기간 만료 시 접근 토큰 재발급을 위한 **재발급 토큰(Refresh Token)** 추가 발급 |
| **⑤ 단계** | 클라이언트가 자원 서버에게 접근 토큰으로 인증 및 자원을 요청 |
| **⑥ 단계** | 자원 서버는 접근 토큰을 검증하고, 검증된 경우 서비스 제공 |

## □ OAuth 2.0의 역할 구성

< OAuth 2.0의 역할 및 금융권 예 >

| 역할 | 설명 | 금융권 예 |
|---|---|---|
| 자원 소유자 (Resource owner) | 보호 자원에 대한 접근 권한을 부여할 수 있는 개체(일반적으로 이용자) | 핀테크 업체의 금융 서비스를 사용하는 금융고객 |
| 클라이언트 (Client) | 자원 서버에 보호 자원을 요청하고 관련 서비스를 제공하는 애플리케이션 | 금융 오픈 API를 사용하여 금융 서비스를 제공하는 핀테크 애플리케이션 |
| 자원 서버 (Resource server) | 보호된 자원에 대한 서비스 API를 제공하는 서버 | 금융 오픈 API(예금 조회/이체, 결제 등)를 제공하는 금융사의 서버 |
| 권한 서버 (Authorization server) | 클라이언트가 보호된 자원에 대한 제한된 접근할 수 있도록 자원 접근 권한을 위임 및 관리하는 서버 | 금융 서비스에 대한 제한된 접근 권한을 핀테크 애플리케이션에게 위임 및 관리하는 금융사의 서버 |

○ **(권한/자원 서버)** 권한 서버와 자원 서버는 구축 및 운영 방식에 따라, 동일 또는 별도 서버로 존재 가능하며, 하나의 권한 서버가 발급한 접근 토큰이 여러 자원 서버에 유효할 수 있음(1:N 구성 가능)*

* 권한 서버와 자원 서버간의 관계 및 상호운영방식은 OAuth 2.0 표준의 범위를 벗어남

○ **(클라이언트 유형)** 클라이언트는 자격증명을 안전하게 보관 및 관리할 수 있는지 여부에 따라 크게 비밀(Confidentiality) 및 공개(Public)로 구분

< OAuth 2.0의 클라이언트 분류 >

| 유형 | 특징 | 클라이언트 종류 |
|---|---|---|
| 비밀 | 안전한 서버에서 실행되며, 클라이언트의 자격증명이 안전하게 보호 관리되기 때문에 보안성이 높음 | - 웹 애플리케이션 |
| 공개 | 자원 소유자의 단말에서 실행되기 때문에 공격자가 클라이언트 자격증명에 접근 가능하며 자격증명의 기밀성을 보장하기 어려움 | - 이용자 에이전트 애플리케이션<br>- 네이티브 애플리케이션 |

○ 클라이언트는 구축 방식에 따라 웹 애플리케이션(Web application), 이용자 에이전트 애플리케이션(User-agent-based application), 네이티브 애플리케이션(Native application)으로 분류

< OAuth 2.0의 클라이언트 >

| 클라이언트 유형 | 설명 |
|---|---|
| 웹 애플리케이션 | 웹 서버 상에서 실행되는 애플리케이션으로 자원 소유자는 HTML 및 웹 브라우저 또는 별도의 웹 클라이언트를 통해 접속 |
| 이용자 에이전트 애플리케이션 | 웹 브라우저 등 이용자의 에이전트 상에서 실행되는 애플리케이션으로 클라이언트 측 스크립트(자바 스크립트 등) 및 웹 브라우저 확장 프로그램 형태 등으로 배포 |
| 네이티브 애플리케이션 | PC, 스마트폰 등과 같은 이용자 단말에 직접 설치 및 실행되는 애플리케이션 |

○ **(클라이언트 등록)** OAuth 2.0에서는 API 요청을 한 클라이언트를 식별 및 검증하기 위해, 먼저 권한 서버에 클라이언트를 등록하도록 정의

- 일반적으로 권한 서버(또는 API 개발자 사이트)에서 제공하는 HTML 양식을 통해 개발자(사)가 직접 등록함
- 등록이 완료되면, 권한 서버는 클라이언트 자격증명 정보 중 하나인 client_id, client_secret*를 발급하며, 클라이언트 개발자는 이 자격증명을 애플리케이션 개발 시 포함시킴

* 접근 토큰 요청 및 오픈 API 호출 시 등록된 정상 클라이언트인지 검증하는데 사용됨

< 구글의 클라이언트 등록 정보 및 화면 >

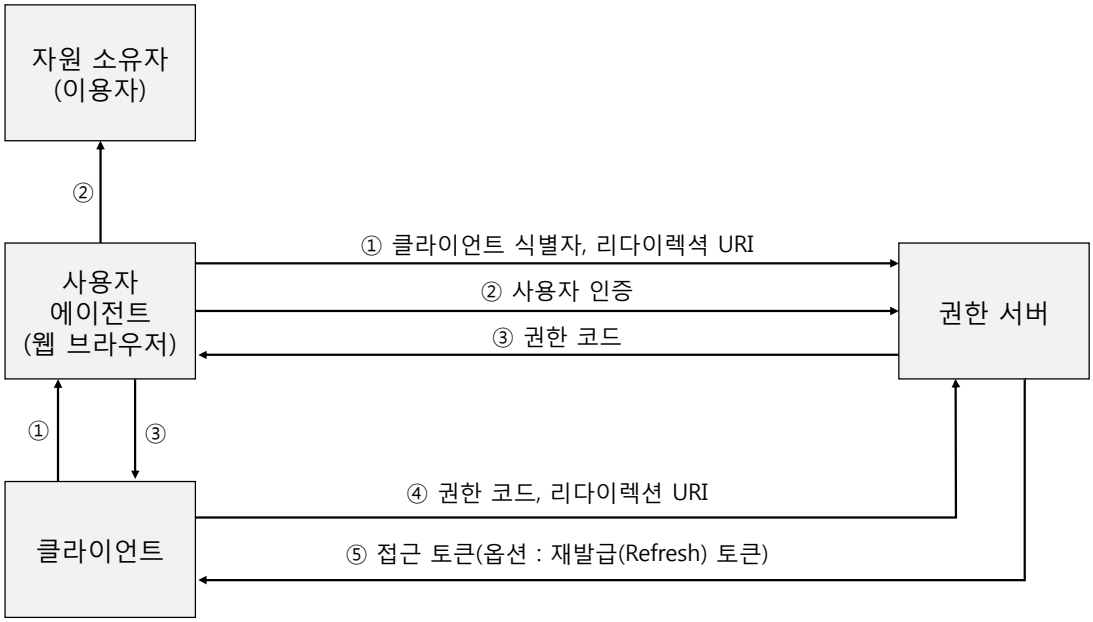
## □ OAuth 2.0의 권한 승인 방법

○ 클라이언트 유형, 자원에 대한 장기적 접근 여부 및 제공 서비스의 특성 등에 따라, 총 4가지 권한 승인 방법을 정의

< OAuth 2.0의 권한 승인 방법 비교 >

| 권한 코드 승인<br>(Authorization Code Grant) | 암묵적 승인<br>(Implicit Grant) |
|---|---|
| ○ **클라이언트 유형**<br>· 서버 웹 애플리케이션<br>· 네이티브 애플리케이션(백엔드 서버 사용)<br>○ **장기적 접근**<br>· 재발급 토큰을 통해 지원<br>○ **보안성**<br>· 이용자 웹 브라우저를 통해 접근 토큰이 노출되지 않도록 **권한 코드** 사용<br>· 클라이언트는 권한 코드를 사용하여 권한 서버에게 직접 접근 토큰을 요청<br>· 중요 자격증명 정보가 서버에 저장되기 때문에 다른 권한 승인 방법에 비해 안전함 | ○ **클라이언트 유형**<br>· 이용자 에이전트 애플리케이션<br>○ **장기적 접근**<br>· 재발급 토큰 미지원으로 인해 접근 토큰의 유효 기간이 짧으면 장기적 접근 불가<br>○ **보안성**<br>· 접근 토큰이 웹 브라우저에 전달 및 저장되기 때문에 웹 브라우저, 클라이언트, 이용자에 대한 신뢰가 높은 경우에 적합<br>· 접근 토큰 유출을 고려하여, 접근 토큰의 유효 기간에 대한 적절한 조절이 필요함 |
| **자원 소유자 비밀번호 자격증명 승인<br>(Resource Owner Password Credentials Grant)** | **클라이언트 자격증명 승인<br>(Client Credentials Grant)** |
| ○ **클라이언트 유형**<br>· 모든 클라이언트 유형 가능<br>· 일반적으로 API 제공 업체가 배포한 애플리케이션<br>○ **장기적 접근**<br>· 재발급 토큰을 통해 지원<br>○ **보안성**<br>· 자원 소유자의 비밀번호가 애플리케이션에 노출되고, 피싱 위험이 존재<br>· 접근 토큰 발급 요청 시에만 비밀번호가 사용되기 때문에 클라이언트에 인증정보를 저장할 필요 없음 | ○ **클라이언트 유형**<br>· 클라이언트가 데이터를 소유하고 있거나, 접근 위임이 이미 허용된 경우<br>○ **장기적 접근**<br>· 재발급 토큰 미지원<br>· 클라이언트 인증만으로 즉시 접근 토큰 재발급 가능<br>○ **보안성**<br>· 클라이언트 인증만으로 접근 토큰을 발급하기 때문에 안전한 인증 방식 사용 및 인증 정보의 규칙적인 변경이 필요함 |

○ 비교적 안전한 권한 코드 승인 방식은 세부 5단계로 구성

<table>
<tr><th>구분</th><th>권한 코드 승인(Authorization Code Grant) 세부 내용</th></tr>
<tr><td>세부<br>단계</td><td>


<table>
<tr><th>단계</th><th>세부 내용</th></tr>
<tr><td>① 단계</td><td>클라이언트가 웹 브라우저에게 리다이렉트 URI를 전달하여 권한 서버의 인증 및 권한 확인 웹 페이지로 리다이렉션 시킴<br>· 권한 서버가 클라이언트를 식별할 수 있도록 client_id를 함께 전달</td></tr>
<tr><td>② 단계</td><td>권한 서버의 자원 소유자에 대한 인증 방식을 통해 인증 및 자원에 대한 접근 권한 승인 요청<br>· 일반적인 웹 서비스에서는 ID/PASSWORD 방식을 통해 자원 소유자 인증</td></tr>
<tr><td>③ 단계</td><td>권한 서버는 웹 브라우저를 다시 클라이언트에게로 리다이렉션* 시키며, 이때 자원 소유자로부터 자원에 대한 접근 권한을 승인 받았음을 나타내는 권한 코드를 클라이언트에게 전달<br>* 클라이언트 등록 시, 권한 서버에 제출한 URI, 또는 ① 단계에서 전달한 리다이렉트 URI 사용</td></tr>
<tr><td>④ 단계</td><td>클라이언트는 접근 토큰을 요청하기 위해, 권한 코드와 함께 클라이언트 인증을 위한 client_id 및 client_secret을 권한 서버에 전달</td></tr>
<tr><td>⑤ 단계</td><td>권한 서버는 접근 토큰을 클라이언트에게 반환<br>· 접근 토큰에 유효 기간이 존재하는 경우, 기간 만료까지 남은 시간(expires_in)과 재발급 토큰(refresh_token) 정보도 함께 반환</td></tr>
<tr><td>접근 토큰 재발급</td><td>접근 토큰의 유효 기간이 만료되어 재발급 받을 경우, 위 단계의 반복 없이 재발급 토큰과 client_id 및 client_secret으로 권한 서버에 재발급 요청</td></tr>
</table>
</td></tr>
</table>

※ 기타 권한 승인 방식의 세부 단계는 RFC-6749 : The OAuth 2.0 Authorization Framework 참조

## □ OAuth 1.0과 2.0 비교

< OAuth 1.0과 2.0의 주요 차이점 비교 >

| 특징 | OAuth 1.0 | OAuth 2.0 |
|---|---|---|
| 역할<br>(역할 명칭 변경 및 세분화) | - 이용자(User) | - 자원 소유자(Resource Owner) |
| | - 소비자(Consumer) | - 클라이언트(Client) |
| | - 서비스 제공자(Service Provider) | - 자원 서버(Resource Server)<br>- 권한 서버(Authorization Server) |
| API 호출 인증 및 보안 | - 서명 | - HTTPS(SSL/TLS) 기본<br>- 서명 : 자원 서버가 별도 서명을 요구하는 경우 |
| 클라이언트 지원 유형 | - 웹 애플리케이션 | - 웹 애플리케이션<br>- 이용자 에이전트 애플리케이션<br>- 네이티브 애플리케이션 |

○ OAuth 2.0은 1.0의 알려진 보안 문제 등을 개선한 버전으로 1.0을 대체(하위 호환성 미지원)

- 기존 서비스 제공자(Service Provider)를 자원 및 권한 서버로 분리하여 다수의 서비스 제공자(서버)로 구성 웹 서비스에서 발생 가능한 권한 동기화 문제 개선
- 오픈 API 요청 시 클라이언트 인증 방법으로 서명 대신, HTTPS를 의무화하여 서버 및 클라이언트 개발 편의성 개선
- 다양한 유형의 클라이언트와 이를 고려한 권한 승인 방법을 정의하여 유형별 클라이언트들에 대한 일관된 구현 가능
- 접근 토큰 재발급을 위한 재발급 토큰(Refresh Token)을 도입함으로써 접근 토큰의 유효 기간 단축 및 보안성 개선
  - 접근 토큰의 유효 기간이 과도하게 긴 경우, 접근 토큰이 유출된 경우 공격자에 의해 장시간 악용 가능
- 기타 다양한 확장성 지원

## □ OAuth 2.0 도입에 따른 보안적인 이점

○ 자원 소유자가 클라이언트에 비밀번호를 제공할 필요가 없으며, 피싱 등의 위협 감소

○ 클라이언트 개발자는 자원 소유자의 비밀번호에 대한 안전한 저장 및 노출을 고려하지 않아도 됨

○ 자원 소유자의 모든 권한이 아닌, 클라이언트에게 반드시 필요한 권한만 제한적으로 제공 가능

○ 자원에 대한 클라이언트의 접근을 각 클라이언트별로 차단 및 권한 취소가 가능하며, 권한의 범위를 제어 가능

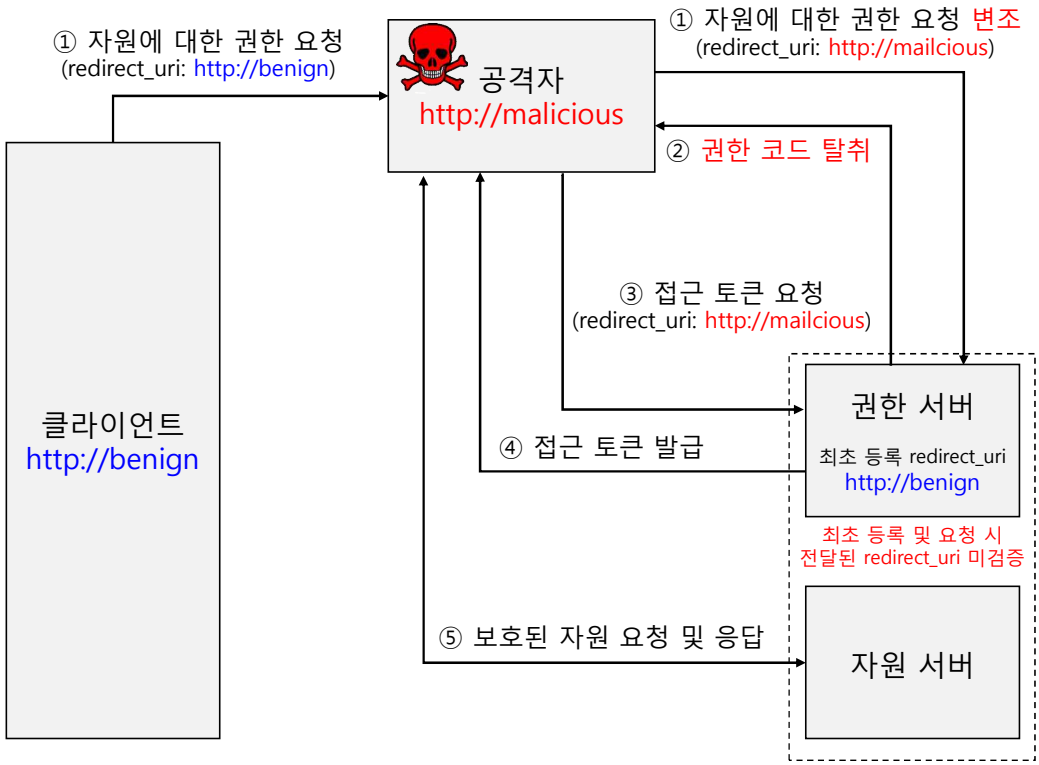
- 비밀번호 방식의 경우, 클라이언트의 권한 취소를 위해서는 비밀번호를 변경해야하며, 접근 권한을 재부여할 클라이언트에게 비밀번호 제공 필요
- 클라이언트별 API 호출 가능 횟수 제한, 일부 자원에 대한 접근만 허용하는 등의 권한 범위 제어 가능

< 자원 소유자의 자원에 대한 클라이언트별 접근 차단 예 : 페이스북 >

## □ OAuth 2.0의 보안 위협 및 대응 방안

○ **(중간자 공격)** 2014년 초에 공개된 은닉 리다이렉트(Covert Redirect)[3] 는 중요 자격증명 정보인 권한 코드 및 접근 토큰 등의 유출로 이어질 수 있는 보안 위협

| 구분 | 상세 설명 |
|---|---|
| 공격 방법 | 공격자가 클라이언트와 권한 서버 간의 권한 승인 요청을 가로채서, 전달 인자 중 redirect_uri* 값을 공격자의 서버로 변조하여, 권한 코드 및 접근 토큰 탈취<br>* 권한 서버가 클라이언트에게 발급하는 권한 코드, 접근 토큰 등이 전달될 주소 |
| 위협 원인 | URI 값은 최초 클라이언트 등록 시 권한 서버에 제출되지만, 최초 제출된 URI와 요청 시 전달된 URI의 미검증 및 정상 클라이언트 여부 미검증이 원인 |
| 파급 효과 | 발표자에 의하면, 위협 공개 당시 페이스북, 구글 등 주요 오픈 API 제공사 대부분이 위협에 노출되어 있는 상태였음 |
| 대응 방안 | 최초 등록된 URI와 요청 시 함께 전달된 URI 비교 및 클라이언트의 IP 주소, 기타 시스템 정보 비교를 통한 클라이언트 검증 절차 준수 |




※ 은닉 리다이렉트 위협은 RFC-6819에 4.2.4. Threat: Open Redirector 로 정의되어있음

3) 리다이렉트 목적지 주소를 변조하여, 어떤 정보가 공격자에게 전달되도록 하는 방법(출처 : http://oauth.net/advisories/2014-1-covert-redirect/, http://www.tetraph.com/blog/covert-redirect/covert-redirect-vulnerability-related-to-oauth-2-0-and-openid/)

○ RFC-6819에는 권한 코드 승인 방법과 관련된 12개의 보안 위협을 정의 및 대응 방안을 제시하고 있으며, 금융권 OAuth 2.0 적용 시 검토 필요

| 분류 | 보안 위협 |
|---|---|
| 권한 코드 유출 | · 권한 코드 도청 및 유출<br>· 권한 서버의 DB로부터 권한 코드 유출<br>· 권한 코드 피싱<br>· 위조 클라이언트로 인한 권한 코드 유출<br>· redirect_uri 에 대한 CSRF(Cross Site Request Forgery)[4] 공격을 통해 권한 코드 획득 |
| 서비스 거부 (DoS) | · 반복된 권한 요청을 통해 권한 서버에 대한 서비스 거부 공격<br>· 악의적으로 생성된 권한 코드를 통해 권한 서버에 서비스 거부 공격 |
| 불법 권한 획득 | · 권한 코드 추측(guessing)을 통한 권한 획득<br>· 악의적인 클라이언트의 권한 획득<br>· 클릭재킹(Clickjacking) 공격을 통한 권한 획득 |
| 신분/세션 도용 | · 자원 소유자 신분 도용<br>· 이용자 세션 도용 |

| 분류 | 대응 방안 |
|---|---|
| 권한 코드 유출 | · SSL/TLS 기반의 HTTPS를 통해 클라이언트와 권한/자원 서버 간 통신 구간 보호<br>· 권한 코드 발급 시 클라이언트 인증 절차 준수<br>· 권한 코드의 유효 기간을 짧게 설정<br>· SQL 인젝션에 대한 대응<br>· redirect_uri 검증 |
| 서비스 거부 (DoS) | · 이용자별 발급 가능한 접근 토큰 개수 제한<br>· 비정상 권한 코드를 일정 횟수 이상 권한 서버에 전송하는 경우, 해당 클라이언트와의 연결 차단 |
| 불법 권한 획득 | · 권한 코드에 메시지 서명 적용<br>· 권한 코드에 1회성 임의 값을 추가<br>· 인증 및 권한 부여 웹 페이지에서는 iFrame을 비활성화 하도록 서버 옵션 추가 |
| 신분/세션 도용 | · CAPTCHA* 사용<br>* 변형된 이미지를 사용하여 사용자가 실제 사람인지를 확인하는 방법<br>· 1회성 인증 정보를 SMS 등의 별도 망으로 발급 |

○ OAuth 2.0에서 다루지 않는 이용자 단말과 통신 구간에 대한 보안 위협 및 대책 고려 필요

4) 이용자가 자신의 의지와는 무관하게 공격자가 의도한 악성 행위를 특정 웹 사이트에 요청하게 하는 공격(출처 : https://ko.wikipedia.org/wiki/사이트 간 요청 위조)