1.3 - Authorization Grant

OAuth 2.0 protokolünde, istemcinin (client) bir **Access Token** almak için kullandığı kimlik belgesidir. Bu belge, **Resource Owner**'ın (kaynak sahibinin) korunan kaynaklara erişim iznini temsil eder.

Bu bağlamda, OAuth 2.0 dört temel **Authorization Grant** türünü tanımlar:

1. **Authorization Code Grant**: Güvenli ve iki adımlı bir süreçtir. İstemci, önce bir "authorization code" alır, ardından bu kodu kullanarak **Access Token** talep eder.
2. **Implicit Grant**: Daha hızlıdır ve genellikle tarayıcı tabanlı uygulamalarda kullanılır. Ancak güvenlik açısından daha az güçlüdür, çünkü **Access Token** doğrudan istemciye sunulur.
3. **Resource Owner Password Credentials Grant**: İstemcinin, kullanıcının kullanıcı adı ve şifresini doğrudan **Authorization Server**'a gönderdiği bir yöntemdir. Güvenilir istemciler için uygundur, ancak genellikle önerilmez.
4. **Client Credentials Grant**: İstemci, kendine ait kimlik bilgilerini kullanarak **Authorization Server**'dan bir **Access Token** talep eder. Genellikle kullanıcıya bağlı olmayan istemci-istemci iletişimleri için kullanılır.

Ayrıca, OAuth 2.0, özel gereksinimler için yeni **Grant** türlerinin tanımlanabilmesine olanak tanıyan bir genişletilebilirlik mekanizması sunar.

1.3.1 - Authorization Code

OAuth 2.0 protokolünde, istemcinin **Access Token** alması için kullanılan bir yetkilendirme türüdür ve en güvenli yöntemlerden biri olarak kabul edilir. Bu mekanizma, istemci ile kaynak sahibi (resource owner) arasında doğrudan bir bağlantı kurmak yerine, bir **Authorization Server**'ın aracı olarak kullanılması prensibine dayanır. İşleyişi şu şekildedir:

1. **Authorization Server Aracılığıyla Yetkilendirme**:
    - İstemci, kaynak sahibinden doğrudan yetkilendirme istemek yerine, kaynak sahibini **Authorization Server**'a yönlendirir.
    - Bu yönlendirme, genellikle kaynak sahibinin kullanıcı arayüzü veya tarayıcı (user-agent) aracılığıyla yapılır.
2. **Kaynak Sahibinin Kimlik Doğrulaması ve Yetkilendirilmesi**:
    - **Authorization Server**, kaynak sahibinin kimlik bilgilerini doğrular ve erişim yetkisini onaylar.
    - Bu süreçte kaynak sahibinin kullanıcı adı veya şifresi gibi kimlik bilgileri, istemciyle paylaşılmaz. Bu, istemciyle hassas bilgiler arasındaki doğrudan temas riskini ortadan kaldırır.
3. **Authorization Code'un İstemciye Dönmesi**:
    - **Authorization Server**, kaynak sahibini istemciye yönlendirirken bir **Authorization Code** sağlar.
    - Bu kod, istemcinin daha sonra bir **Access Token** talep etmek için kullanacağı bir geçici kimlik belgesidir.
4. **Access Token Talebi**:

- İstemci, aldığı **Authorization Code** ile **Authorization Server**'a bir **Access Token** talebinde bulunur.
- Bu adımda istemci kendini **Authorization Server**'a kimlik bilgileriyle doğrulayarak ek bir güvenlik katmanı sağlar.

Güvenlik Faydaları:

- **Kimlik Doğrulama Güvencesi**: **Authorization Server**, istemcinin kimliğini doğrular.
- **Access Token'ın Doğrudan İletilmesi**: **Access Token**, istemciye doğrudan iletilir ve kaynak sahibinin tarayıcısı üzerinden geçmediği için maruz kalma riski azalır.
- **Kaynak Sahibinin Gizliliği**: Kaynak sahibinin kimlik bilgileri (örneğin, kullanıcı adı ve şifre) istemciyle asla paylaşılmaz, yalnızca **Authorization Server** ile paylaşılır.

Bu yöntem, güvenliğin ve hassas bilgilerin korunmasının ön planda olduğu istemci-tarayıcı etkileşimlerinde yaygın olarak tercih edilir.

1.3.2 - Implicit grant

Tarayıcı tabanlı uygulamalar (örneğin JavaScript ile yazılmış istemciler) için optimize edilmiş ve OAuth 2.0 sürecini basitleştiren bir yetkilendirme türüdür. Bu tür, daha az işlem gerektirdiği için **Authorization Code** yöntemine kıyasla daha hızlıdır, ancak bazı güvenlik zafiyetleri içerir. İşleyişi ve özellikleri şu şekilde açıklanabilir:

İşleyiş:

1. **Erişim Tokenı Doğrudan Verilir**:
    - **Authorization Server**, istemciye bir **Authorization Code** yerine doğrudan bir **Access Token** sağlar.
    - **Access Token**, kaynak sahibinin yetkilendirme süreci tamamlandıktan hemen sonra istemciye iletilir.
    - Arada bir **Authorization Code** alma ve bu kodu kullanarak token isteme adımları atlanır.
2. **İstemci Kimlik Doğrulaması Olmaz**:
    - Bu akışta **Authorization Server**, istemciyi kimlik doğrulamasından geçirmez.
    - Bunun yerine, istemcinin kimliğini doğrulamak için genellikle istemcinin kullandığı **Redirect URI** gibi faktörlere güvenilir.
3. **Tarayıcı Üzerinden İşleme**:
    - Token, kaynak sahibinin tarayıcısı (user-agent) üzerinden istemciye iletilir.
    - Bu, tokenın tarayıcıya veya başka bir uygulamaya maruz kalma riskini artırır.

Avantajlar:

- **Hız ve Verimlilik**:

    - İstemcinin **Access Token** alması için gereken işlem sayısı azaltıldığı için daha hızlıdır.
    - Özellikle tek sayfa uygulamalar (SPA) gibi tarayıcı tabanlı istemcilerde performansı artırır.
- **Basitleştirilmiş Akış**:
    - Kullanıcı deneyimini iyileştiren ve daha az adım gerektiren bir yapı sağlar.

Güvenlik Riskleri:

1. **Tokenın Maruz Kalma Riski**:
    - **Access Token**, kaynak sahibinin tarayıcısı üzerinden istemciye iletildiğinden, tarayıcıya veya başka bir uygulamaya maruz kalabilir.
    - Eğer tarayıcıda açıklar veya kötü amaçlı yazılımlar varsa, tokenın ele geçirilme riski yüksektir.
2. **İstemcinin Doğrulanmaması**:
    - İstemci kimlik doğrulaması yapılmadığı için kötü niyetli istemcilerin **Access Token** alması kolaylaşabilir.
3. **Token Süresinin Kısa Olması Gerekliliği**:
    - Bu akışta güvenlik risklerini en aza indirmek için token süreleri çok kısa tutulmalıdır.

Nerelerde Kullanılır?

- **Tek Sayfa Uygulamalar (Single Page Applications - SPA)**:
    - React, Angular gibi tarayıcı tabanlı çerçevelerle yazılmış uygulamalarda sıkça tercih edilir.
- **Sunucusuz (Serverless) Uygulamalar**:
    - İstemcinin yalnızca tarayıcıda çalıştığı ve backend sunucusunun olmadığı durumlarda uygundur.

Güvenlik Açısından Değerlendirme:

Authorization Code Grant, güvenlik açısından daha güçlüdür ve mümkünse tercih edilmelidir. Ancak, yalnızca tarayıcı tabanlı bir uygulama kullanılıyorsa ve hızlı bir akış gerekiyorsa, **Implicit Grant** seçeneği kullanılabilir. Kullanım sırasında, güvenlik risklerini azaltmak için şunlara dikkat edilmelidir:

- Token süresi kısa tutulmalı.
- **Redirect URI** kesinlikle doğrulanmalı.
- Tarayıcı güvenliğinin sağlandığından emin olunmalı.

Sonuç olarak, **Implicit Grant**, belirli durumlarda performans avantajı sağlarken güvenlik açıkları barındırdığı için dikkatli bir şekilde değerlendirilmesi gereken bir yöntemdir.

1.3.3 - Resource Owner Password Credentials (ROPC)

Kaynak sahibinin kullanıcı adı ve şifre bilgilerini doğrudan bir yetkilendirme belgesi (grant) olarak kullanarak istemciye erişim tokenı sağlama yöntemidir. Bu yöntem yalnızca kaynak sahibi ile istemci arasında yüksek güven ilişkisi olduğu durumlarda kullanılmalıdır. Örneğin, istemci işletim sisteminin bir parçası olduğunda veya çok yetkili bir uygulama olduğunda tercih edilebilir. İşleyişi ve özellikleri şu şekilde özetlenebilir:

İşleyiş:

1. **Kullanıcı Adı ve Şifre Kullanımı**:
    - Kaynak sahibi (örneğin bir kullanıcı), istemciye kullanıcı adı ve şifresini doğrudan verir.
    - İstemci, bu kimlik bilgilerini **Authorization Server**'a gönderir.
2. **Erişim Tokenı Sağlanması**:
    - **Authorization Server**, bu kimlik bilgilerini doğrular.
    - Eğer doğrulama başarılı olursa, istemciye bir **Access Token** sağlar.
    - Token, kaynak sahibinin korunan kaynaklarına erişim için kullanılır.
3. **Kimlik Bilgilerinin Tek Seferlik Kullanımı**:
    - Kaynak sahibinin kullanıcı adı ve şifre bilgileri, yalnızca token almak için kullanılır ve başka bir amaçla saklanmaz.
    - Uzun süreli bir **Access Token** veya **Refresh Token** ile kimlik bilgilerini saklama ihtiyacı ortadan kalkar.

Avantajlar:

- **Basitlik**:
    - Diğer grant türleri (örneğin Authorization Code veya Implicit) yerine daha doğrudan ve basit bir yöntem sunar.
- **Kimlik Bilgilerini Saklama Zorunluluğunun Azalması**:
    - İstemci, kullanıcı adı ve şifre gibi hassas bilgileri uzun süreli saklamak yerine bunları bir token ile değiştirir.
- **Yetkilendirme Türünün Alternatif Olmadığı Durumlar İçin Uygun**:
    - Örneğin, bir cihazın işletim sistemi parçası olan istemcilerde ya da başka grant türlerinin uygulanamayacağı durumlarda kullanışlıdır.

Güvenlik Riskleri ve Dikkat Edilmesi Gerekenler:

1. **Kimlik Bilgilerinin Ele Geçirilme Riski**:
    - Kullanıcı adı ve şifre bilgilerinin istemciye doğrudan verilmesi, büyük bir güvenlik riski doğurabilir.

    - İstemcinin güvenilir ve güvenli bir şekilde çalıştığından emin olunmalıdır.
2. **Sınırlı Kullanım Alanı**:
    - Yalnızca istemci ile kaynak sahibi arasında yüksek güven ilişkisi olduğu durumlarda kullanılmalıdır.
    - Düşük güven seviyesine sahip istemciler için uygun değildir.
3. **Alternatif Grant Türlerinin Tercihi**:
    - Mümkünse **Authorization Code Grant** gibi daha güvenli grant türleri kullanılmalıdır.
    - ROPC, yalnızca başka bir seçeneğin olmadığı durumlarda bir "son çare" olarak değerlendirilmelidir.

Nerelerde Kullanılır?

- **Cihaz Tabanlı Uygulamalar**:
    - Örneğin, işletim sistemi tarafından entegre edilmiş bir istemci uygulamasında kullanılabilir.
- **Yüksek Güven Gerektiren Uygulamalar**:
    - Uygulamanın doğrudan kontrol edilen bir ortamda çalıştığı ve güvenliğin tamamen sağlandığı senaryolarda uygundur.

Değerlendirme ve Sonuç:

Resource Owner Password Credentials Grant, güvenliğin ve gizliliğin kritik olduğu durumlarda dikkatli bir şekilde kullanılmalıdır. Bu grant türü, istemcinin kaynak sahibinin kullanıcı adı ve şifre bilgilerine doğrudan erişim gerektirdiği için, kötüye kullanım riskini beraberinde getirir. Bu nedenle:

- Uygulama, yalnızca yüksek güven ortamlarında çalıştırılmalı.
- Kullanıcı adı ve şifre bilgileri yalnızca bir kere kullanılmalı ve hiçbir şekilde saklanmamalı.
- Mümkünse, daha güvenli grant türleri tercih edilmelidir.

Bu yöntem, yalnızca belirli ve sınırlı kullanım senaryolarında, güvenilir istemcilerle uygulanmalıdır.

1.3.4 - Client Credentials

İstemcinin kimliğini doğrulamak ve kaynaklara erişim izni almak için kullanılan bir yetkilendirme türüdür. Bu grant türü genellikle **istemcinin kendi adına** veya **önceden yetkilendirilmiş erişim** ile korunan kaynaklara erişim sağlamak amacıyla kullanılır. Başka bir deyişle, istemci, genellikle kaynak sahibiyle aynı kimliklere sahip değildir ve yalnızca **kendi kaynaklarına** erişim talep eder.

İşleyiş:

1. **Kimlik Doğrulama**:

    - İstemci, bir **access token** almak için **Authorization Server**'a kimlik bilgilerini (client ID ve client secret gibi) sunar.
    - Bu kimlik bilgileri istemcinin güvenli bir şekilde doğrulanmasını sağlar.
2. **Erişim Tokenı Alınması**:
    - İstemci, **client credentials**'ını kullanarak **authorization server** üzerinden **access token** talep eder.
    - Bu işlem, istemcinin kaynak sahibinin kimlik bilgilerine veya şifresine ihtiyacı olmadığı için daha güvenli ve hızlıdır.
3. **Erişim Sağlanması**:
    - Erişim tokenı alındıktan sonra istemci, bu token'ı kullanarak **resource server**'a (korunan kaynak sunucusu) erişebilir.

Ne Zaman Kullanılır?

- **İstemci Kendi Kaynaklarına Erişmek İstediğinde**:
    - Eğer istemci, sadece kendi kontrolündeki kaynaklara erişim talep ediyorsa (örneğin, kendi veritabanındaki verilere), client credentials grant türü uygundur.
- **Önceden Yetkilendirilmiş Erişim Senaryoları**:
    - Eğer bir istemci, belirli bir kaynak üzerinde önceden anlaşmaya varmışsa (örneğin, bir API üzerinden belirli verilere erişim için önceden yetkilendirilmişse), bu tür bir grant kullanılabilir.
- **Sunucu-Sunucu İletişimi**:
    - Client credentials grant türü, genellikle **sunucu-sunucu iletişimi** için kullanılır. Örneğin, bir microservice mimarisinde, mikroservislerin kendi aralarında güvenli bir şekilde iletişim kurması gerektiğinde bu tür bir grant kullanılır.

Avantajlar:

1. **Güvenli ve Basit**:
    - Kaynak sahibinin kimlik bilgilerini istemciyle paylaşma gereksinimi yoktur. İstemci yalnızca kendi kimlik bilgilerini (client ID ve client secret) kullanarak kimlik doğrulaması yapar.
2. **İstemcinin Kendi Kaynaklarıyla İşlem Yapması**:
    - İstemci, yalnızca kendi kaynaklarına veya önceden yetkilendirilmiş kaynaklara erişim talep ettiği için, diğer kullanıcıların kaynaklarıyla ilgili bir risk oluşmaz.
3. **Sunucu-Sunucu Erişimi İçin Uygun**:
    - Genellikle API'ler veya mikro servisler gibi sunucu tabanlı uygulamalar arasında kullanılır. Bu tür uygulamalar genellikle kullanıcı adı ve şifre gerektirmeden yalnızca istemci kimlik bilgileriyle kimlik doğrulaması yapar.

Sınırlamalar ve Güvenlik Riskleri:

1. **Kaynak Sahipliği**:
    - Bu grant türü, **istemcinin kaynak sahibiyle aynı olması** durumunda kullanılabilir. Yani istemci, yalnızca kendi kaynakları üzerinde işlem yapabilir. Başka bir deyişle, istemcinin, kaynak sahibi olmayan veriler üzerinde işlem yapması için başka bir grant türü gereklidir.
2. **Client Secret'ın Güvenliği**:
    - **Client Secret** gibi kimlik bilgileri, istemcinin güvenli bir ortamda tutulması gerektiği için doğru güvenlik önlemleri alınmalıdır. Eğer bu bilgiler ele geçirilirse, kötü niyetli kişiler istemci adına yetkisiz erişimler yapabilir.
3. **Kısıtlı Erişim**:
    - Bu grant türü, yalnızca istemcinin önceden belirlenmiş ve yetkilendirilmiş kaynaklarına erişim sağlar. Diğer kaynaklara erişim için farklı bir grant türü gereklidir.

Sonuç:

Client Credentials Grant, özellikle istemcinin kendi kontrolündeki kaynaklara erişim sağlamak için tasarlanmış güvenli bir yetkilendirme türüdür. Bu yöntem, istemcilerin kimlik doğrulaması yapmasını sağlar, ancak kaynak sahibiyle aynı kimlikleri taşımayan istemciler için bu yöntem dışında başka grant türleri kullanılmalıdır.

Bu grant türü, genellikle **sunucu-sunucu iletişimi** ve **API entegrasyonları** için uygun olup, yüksek güvenlik önlemleri gerektiren ortamlarda uygulanmalıdır.