1 – What are Authentication and Authorization ?

Authentication,bir kullanıcının herhangi bir kaynağa erişimde kimliğinin doğrulanması işlemidir. Kullanıcıya Kimsin sorusu sorulur? Bu sorunun cevabı genellikle kullanıcının kullanıcı adı ve şifre şeklinde cevap vermesiyle yanıtlanır. Güvenliğin daha yüksek tutulması gereken durumlarda kullanıcı adı ve şifre ile beraber dijital sertifikalar kullanılarak güvenliğin arttırılması sağlanabilir. Bunu kimi şirketelere girerken çalışanların giriş yapabilmeler için ringlere bastıkları kartlara benzetebiliriz. Authentication, authorization'dan önce gelmektedir.

Authorization ise kimliği doğrulanan kullanıcının erişmek istediği kaynak üzerindeki yetkilerini tanımlar. Dosya-klasör erişimleri,erişim saatleri,ayrılmış alan miktarı v.b. Buna şöyle bir örnek verebiliriz, bizim bir evimiz var ve bir iç mimar var. Ben bu iç mimara evimi döşemesi için izin verirsem bu ona özel tanımladığım bir ayrımcılık olur.

2 - What is Hashing in Spring Security ?

Öncelikle Hashing terimini açıklamak gerekir.

Hashing , açık parolalarla sisteme anında erişim sorununu çözer. Hashing, girişi bir sembol satırına dönüştüren tek yönlü bir işlevdir. Normalde bu hattın uzunluğu sabittir.

Şifreyi iki durumda hash etmemiz gerekiyor. Kullanıcı uygulamaya kaydolduğunda şifreyi hash eder ve veritabanına kaydederiz İkinci olarak Kullanıcı kimlik doğrulaması yapmak istediğinde, sağlanan şifreyi hash eder ve veritabanındaki şifre hashiyle karşılaştırırız.

Java hem PBKDF2 hem de SHA karma algoritmalarını yerel olarak desteklese de, BCrypt ve SCrypt algoritmalarını desteklemez.

Fakat Spring Security, PasswordEncoder interface'i ile önerilen tüm bu algoritmalar için destekle birlikte gelir. Pbkdf2PasswordEncoder bize PBKDF2'yi, BCryptPasswordEncoder bize BCrypt'i verir ve SCryptPasswordEncoder bize SCrypt verir.

3 - What is Salting and why do we use the process of Salting ?

Salting, parola ile birlikte karma işlemi yapılan, rastgele oluşturulmuş bir bayt dizisidir. Bu salt storage'da saklanır ve korunmasına gerek yoktur. Kullanıcı kimlik doğrulaması yapmaya çalıştığında, kullanıcının parolası kaydedilen tuzla karıştırılır ve sonuç, saklanan parolayla eşleşmelidir. Parola ve salt kombinasyonunun bir gökkuşağı tablosunda önceden hesaplanması olasılığı çok küçüktür. Salt yeterince uzun ve rastgele ise, bir gökkuşağı tablosunda karma bulmak daha da zorlaşır. Ancak donanımlar giderek daha verimli hale geliyor. Saldırganın saniyede milyarlarca karma hesaplayabileceği donanımlar gün geçtikçe olası hale gelmektedir.

4 - What is “intercept-url” pattern ?

Spring Security ile beraber kullanılan Url’leri role göre sınırlanma işlemidir. Pattern keyine verdiğimiz değer ile sınırlayacağımız url’i belirtirken, access keyine vereceğimiz değer ile de hangi role göre sınırlandırma yapacağımızı belirtiriz. Örnek kullanım şeklini vermek gerekirse,

```
<security:intercept-url pattern="/admin/*" access="ROLE_ADMIN"/>
<security:intercept-url pattern="/import/*" access="ROLE_ADMIN"/>
<security:intercept-url pattern="/user/*" access="ROLE_USER"/>
```

Bu şekilde Admin, User ya da Anonymous kullanıcılara url bazlı sınırlandırma yapabiliyoruz.

5 - What do you mean by session management in Spring Security ?

Oturumdaki zaman aşımlarının tespiti ve bu zaman aşımları sonucu nasıl bir yönlendirme yapılacağı, bir kullanıcının aynı anda kaç oturum açabileceği ve bu oturumlar sonucu ne gibi aksiyonlar alınacağı, oturumlar’ın HTTPS ile gönderilerek kullanıcı verilerinin güvenliğinin sağlanması gibi konuları içerir.

```
<session-management>
    <concurrency-control expired-url="/sessionExpired.html" ... />
</session-management>
```

Şu şekilde bizim belirlediğimiz zamandan sonra zaman aşımına uğrayan kullanıcılar tekrar istek gönderirse istediğimiz Url’ e yönlendirme yapabiliyoruz.

```
<http ...>
    <session-management>
        <concurrency-control max-sessions="2" />
    </session-management>
</http>
```

Bu şekilde çoklu oturum sayısı için xml yapılandırması gerekiyor.

Çoklu oturumlarda ise 2 yöntem öne çıkıyor. Aynı kullanıcı ile farklı bir yerden login olunduğu vakit, eğer izin verilen oturum sayısı aşılmış ise son login olunan yerde hata vermek, diğerinde ise halihazırda açılmış en eski oturumu sonlandırmaktır.

6 – Why we need Exception Handling ?

Java exception handling, beklenmeyen olaylar meydana geldiğinde bile programın normal, istenen akışının korunmasına yardımcı olduğu için önemlidir. Java exceptionları handle etmez ise, programlar çökebilir veya istekler başarısız olabilir. Bu, müşteriler için çok sinir bozucu olabilir ve tekrar ederse, bu müşterileri kaybedebilirsiniz. Bunu bir örnekle açıklamak gerekirse

```
statement 1;
statement 2;
statement 3;
statement 4;
statement 5;//exception occurs
statement 6;
statement 7;
statement 8;
statement 9;
statement 10;
```

Bir Java programında 10 ifade olduğunu ve 5. ifadede bir exception oluştuğunu varsayalım; kodun geri kalanı yürütülmeyecek, yani 6'dan 10'a kadar olan ifadeler yürütülmeyecektir. Ancak, exception handling yaptığımızda, deyimlerin geri kalanı yürütülecektir. Bu nedenle Java'da exception handling kullanıyoruz.

7 - Explain what is AuthenticationManager in Spring security ?

Basitçe söylemek gerekirse, AuthenticationManager, kimlik doğrulama için ana yöntem interface'idir.

Authentication Manager bir interfacedir ve kimlik doğrulama metodu çalıştırılmaktadır. Authentication Manager bir interface olup, Authentication Provider'a gönderir. Kimlik doğrulama işlemlerinde hangi tipte bir doğrulama işleminin yapılacağını Authentication Provider'a bildirir.

Giriş kimlik doğrulaması geçerli ve doğrulanmışsa, AuthenticationManager#authenticate, kimliği doğrulanmış bayrağı true olarak ayarlanmış bir Kimlik Doğrulama örneği döndürür. Aksi takdirde, geçerli değilse, bir AuthenticationException oluşturacaktır. Son durumda, karar veremezse null döndürür.

9 – What are the differences between OAuth2 and JWT ?

OAuth 2.0 bir protokol tanımlar, yani tokenların nasıl aktarılacağını belirtir, JWT bir token formatı tanımlar. OAuth 2.0 ve "JWT kimlik doğrulaması", Client'ın tokenı Kaynak Sunucusuna sunduğu (2.) aşama söz konusu olduğunda benzer görünüme sahiptir. Yani token, header içinde geçirilir.

Ancak "JWT kimlik doğrulaması" bir standart değildir ve Client'ın tokenı ilk etapta nasıl elde ettiğini belirtmez. OAuth'un algılanan karmaşıklığı buradan gelir: Client Yetkilendirme Sunucusu adı verilen bir şeyden erişim belirteci alabileceği çeşitli yolları da tanımlar.

Yani asıl fark, JWT'nin yalnızca bir belirteç biçimi olması, OAuth 2.0'ın bir protokol olmasıdır. Bir JWT'yi token formatı olarak kullanabilir.

10- What is method security and why do we need it ?

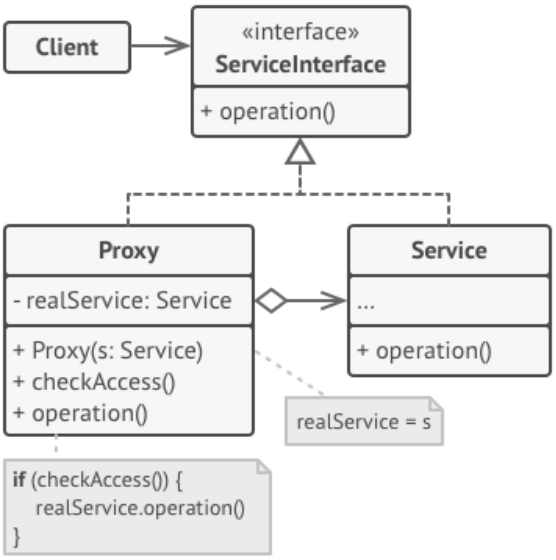
Method düzeyinde güvenlik, yetkisiz kullanıcıların rolleri ve ayrıcalıkları dışında etkinlikler gerçekleştirmesini önlemek içindir. Herhangi bir uygulamanın soyut tasarımı, frontend'i backend'den bağımsız (veya gevşek bir şekilde bağlanmış) tutar.

Birbirleri arasındaki bu kopukluk nedeniyle, backend güvenlik framework'ü, web düzeyinde güvenliğin kusursuz bir şekilde uygulandığını varsayamaz ve bu nedenle, uygulamalarda yöntem düzeyinde güvenlik uygulamak önemli hale gelir.

11 – What Proxy means and how and where can be used ?

Proxy, bir nesnenin kendisine erişimi kontrol etmesi için bir vekil nesne veya yer tutucudur. Proxy, bir nesnenin arayanı ile gerçek nesnenin kendisi arasında bulunduğundan, gerçek (veya hedef) nesnenin çağrılmasını engellemeye veya hedef nesne çağrılmadan önce bir şeyler yapmaya karar verebilir.

Başka bir deyişle, proxy'ler, güvenlikle ilgili davranış, önbelleğe alma veya performans ölçümleri gibi bu nesnelere ekstra davranışlar uygulamak için gerçek nesneler için yedek olarak kullanılabilir. Sınıfın ana logic'inden önce veya sonra bir şey execute etmek istiyorsak, proxy bunu sınıfı değiştirmeden yapmanıza izin verir.



Resimde de görüldüğü gibi bir Interface'i implemente eden ana bir A service'imiz vardır. İçerisinde çeşitli logicler yapar. Proxy'de bu aynı interface'i implemente eder. Proxy class'ı içerisinde A service'ini bir field olarak tutar. Proxy işlemlerini bitirdikten sonra isteği service nesnesine pass eder. Genellikle proxy'ler, service nesnelerinin tüm yaşam döngüsünü yönetir.

12 – Waht is Wrapper Class and where can be used ?

Adından da anlaşılacağı gibi, bir wrapper sınıfı bir veri tipinin etrafını sarar (kapsar) ve ona bir nesne görünümü verir. Bir nesne olarak veri tipinin gerekli olduğu her yerde, bu nesne kullanılabilir. Wrapper sınıfları, nesnenin paketini açmak ve veri türünü geri vermek için yöntemler içerir.
Bir çikolata ile karşılaştırılabilir. Üretici, kirlenmeyi önlemek için çikolatayı bir miktar folyo veya kağıtla sarar. Kullanıcı çikolatayı alır, ambalajı çıkarır ve fırlatır ve yer.

- Primitive veri tiplerini bir obje olarak kullanmak istersek kullanabiliriz.
- Java.util package içinde sadece sınıflarla uygulama yapabiliriz ve biz de wrapper classları bu şekilde kullanabiliriz
- ArrayList ve Vector gibi veri yapıları için wrapper classlar aracılığıyla primitive typeları kullanabiliriz.
- Multithreading senkronizasyonu için gerekli bir obje oluşturma amaçlı kullanabiliriz.

13 – What is SSL ? What is TLS ? What is the difference ? How can we use them ?

Secure Sockets Layer (**SSL**), internet üzerinden hassas bilgi göndermek ve almak için modern bir şifreleme yöntemi kullanan güvenlik protokolüdür. Bir kullanıcının tarayıcısı ile kullanıcının istediği web sitesinin sunucusu arasında güvenli bir kanal oluşturarak çalışır. Bu kanaldan geçen herhangi bilgi, bir uçta şifrelenir ve diğer uçtan alındığında şifre çözülür. Böylece, birisi bu bilgileri eline geçirse bile, bilginin şifreli olmasından ötürü hiçbir fayda sağlayamayacaktır.

TSL (Transport Layer Security / Taşıma Katmanı Güvenliği), IETF standartlar yolu protokolüdür ve önceki SSL spesifikasyonları esas alınarak SSL'i de kullanıma sunan Netscape tarafından geliştirilmiştir. Bu açıdan değerlendirildiğinde, SSL için TLS'nin öncülü diyebiliriz. Bu nedenle kimi zaman SSL/TLS olarak da adlandırılır. Kısaca bu süreci açıklamak gerekirse, SSL protokolünün yayınlanan son versiyonu 3.0 sonrasında TLS 1.0 ile devamlılık sağlanmıştır.

TLS iki katmandan oluşur. TLS Record Protocol (TLS Kayıt Protokolü) ve TLS Handshake Protocol (TLS El Sıkışma Protokolü)

Handshake Protokolü, sunucu ve kullanıcıların kimlik doğrulama işlemlerini gerçekleştirir. Handshake Protocol veri iletişimi öncesine şifreleme algoritmaları ile şifreleme anahtarlarına izin verilirken, Record Protocol ile bağlantının güvenliği sağlanır.

- Güvenlik noktasında hem SSL hem de TLS hemen hemen eşit seviyede güvenlik sunar. Ancak, SSL süreci güvenlik ile başlatıp doğrudan güvenli veri iletişimine geçerken, TLS sunucuya güvenli olmayan bir açılış mesajı gönderir. Eğer bu mesaj üzerinden istemci ile sunucu arasında bir handshake gerçekleşmezse (message authentication) bağlantı kurulmaz. Handshake sağlanmış ise güvenli bağlantı kurulur ve veri iletişimi sağlanır.
- Eğer istemci bir SSL sertifikasına sahip değilse TLS protokolü "Sertifika Yok" mesajını geçirebilir. SSL'de ise ayrıca bir bilgilendirmeye gerek yoktur.
- H-MAC herhangi bir hash fonksiyonuyla yönetilebilir olduğu için TLS pek çok durumda MAC (H-MAC) kullanır. Ancak, SSL MD5 ve SHA kullanmaktadır.
- TLS sertifika doğrulama mesajını handshake içerisinde iletir, SSL ise doğrulama mesajını daha kompleks bir süreç içerisinde iletmektedir.
- TLS, "bitti mesajı" için PRF çıktısıyla birlikte istemci ve sunucudan gelen "bitti" mesajlarını kullanır. SSL ise anahtar üretimine benzer biçimde bir "bitti mesajı" oluşturur.

TLS kullanımı için öncelikle sunucu ayarlarından SSL 2.0 ve SSL 3.0'ın devre dışı bırakılması gerekmektedir.

14 - Why do you need the intercept-url ?

Varolan kaynaklarımıza url üzerinden kimler tarafından nasıl ulaşılacağını belirlemek için ihtiyaç duyuyoruz.

```
<security:intercept-url pattern="/admin/*" access="ROLE_ADMIN"/>
<security:intercept-url pattern="/import/*" access="ROLE_ADMIN"/>
<security:intercept-url pattern="/user/*" access="ROLE_USER"/>
```

Şu şekilde de örnek verebiliriz.