**JAAS – Java ile kimlik kontrolü**

Java ile yazdığınız uygulamanın kaynak kodlarında bir satır bile değiştirmeden kullanıcı adı ve şifre kontrolünü, akıllı kart kullanımını ya da parmak izi kullanımını nasıl eklersiniz?

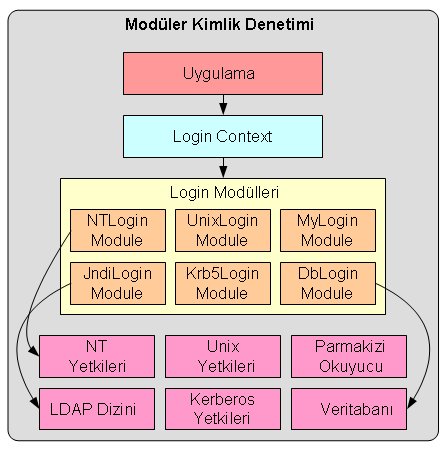
**JAAS (Java Authentication and Authorization Service)** paketleri Java’nın içindeki güvenlik sistemini biraz daha geliştiriyor ve farklı kaynaklar kullanılarak güvenlik sistemleri oluşturmayı mümkün hale getiriyor.

Java SDK içindeki **java.security.\*** paketleri bir uygulamanın güvenliğinin sağlanması için temel metotları içeriyor. Bunlar arasında **byte-code doğrulayıcısı** (byte-code verifier), **sınıf yükleyicisi**(class loader), **güvenlik yöneticisi**(security manager), **erişim denetleyicisi**(access controller), **politikalar**(policy) ve **koruyucu alanlar**(protecting domain) sayılabilir.

Bunlara ek olarak genişletme kütüphaneleri (extension API) sayesinde uygulamaların güvenliği istenilen seviyelerde arttırılabiliyor. JAAS bunlardan birisi…

JAAS temel olarak uygulamaya kimlik bilgisi döndürecek bir **LoginContext**ve bu kimlik bilgisinin çeşitli yollarla elde edilmesini sağlayan bir **LoginModule**‘den oluşuyor.

LoginModule sınıfından genişletilerek oluşturulan modüller kimlik bilgisi için bir veritabanına bağlanan, Kerberos ya da LDAP dizininden kullanıcı bilgisi alan, akıllı kart okuyucudan veya parmak izi okuyucudan bilgi aktaran bir yapıda tasarlanabilir. Uygulamadan bağımsız olarak modüller tasarlamak bize eklenip çıkarılabilir bir kimlik denetimi yapısı sağlar.



Uygulamamızda JAAS kullanmak için şöyle bir yol izliyoruz:

Öncelikle bir konfigürasyon dosyası oluşturuyoruz. Daha sonra uygulamayı çalıştırırken parametre olarak bu dosyayı vereceğiz.

**jaas.config**

Ornek   
{  
   DBLoginModule required   
        debug="true"   
        url="jdbc:mysql://localhost/jaasdb?user=root&password=pass"   
        driver="org.gjt.mm.mysql.Driver";  
};

Uygulamayı çalıştırırken kullanımı:

java -Djava.security.auth.login.config=jaas.config JaasTest

Burada kullanacağımız modül, bir MySql veritabanına bağlanarak kullanıcı adı kontrolü yapıyor.

JaasTest uygulamamıza bu modüllere erişip kullanacak LoginContext’i tanımlıyoruz.

**JaasTest.java**

...  
/\* JAAS sınıfları \*/  
import java.security.\*;  
import javax.security.auth.\*;  
import javax.security.auth.login.\*;  
  
...  
  
        try {  
            // LoginModule ile haberleşecek olan sınıf  
            ConsoleCallbackHandler cbh = new ConsoleCallbackHandler();  
           
            LoginContext lc = new LoginContext("Ornek", cbh);  
  
            try {  
                // LoginModule içinde login metodunu kendimize uygun şekilde  
                // tanımlayacağız.  
                lc.login();  
  
                // Subject : kimlik denetimi bilgilerini içeren nese   
                subject = lc.getSubject();  
  
                // Principal : Yetkili olan kişiler vb.  
                Iterator it = subject.getPrincipals().iterator();  
  
                // Credential : Yetkiler, haklar vb.  
                it = subject.getPublicCredentials(Properties.class).iterator();  
  
                ...  
                // Programın çalışma rutinleri  
                  
                // Program çıkışında   
                lc.logout();  
            } catch (LoginException lex)   
            {  
              ....  
            }  
  
        } catch (Exception ex)   
        {  
            ...  
        }  
  
        System.exit(0);  
    }

Temel anlamda kullanılan sınıf ve metotlar böyle…  
Şimdi MySQL veritabanına bağlanıp kontrol yapan login modülümüze bakalım.

**DBLoginModule.java**

...  
  
import java.security.\*;  
import javax.security.auth.spi.LoginModule;  
import javax.security.auth.login.LoginException;  
import javax.security.auth.Subject;  
import javax.security.auth.callback.\*;  
  
...  
  
public class DBLoginModule implements LoginModule   
{  
  
    // initial state  
    CallbackHandler callbackHandler;  
    Subject  subject;  
  
...  
  
    public void initialize(Subject subject, CallbackHandler callbackHandler,  
            Map sharedState, Map options)   
    {  
      // ön işlemler  
    }  
      
    public boolean login() throws LoginException   
    {  
        // veritabanına bağlanıp yapılacak kontroller  
        // Buraya akıllı kart, parmak izi okuyucu vb.   
        // kontrolleri eklenebilir  
    }  
  
    public boolean commit() throws LoginException   
    {  
       // Otomatikmen çalışır  
       // login işlemi doğru yapıldı ise true, değilse false döndürür  
    }  
  
    public boolean abort() throws javax.security.auth.login.LoginException   
    {  
      // işlem başarısız  olduğunda burası çalışır  
    }  
  
    public boolean logout() throws javax.security.auth.login.LoginException   
    {  
      // uygulamadan çıkışta yetkileri temizler   
    }  
}

Bu örnekte veritabanına bağlandık. Sadece LoginModule ve jaas.config dosyalarında değişiklik yapılarak istenen kontrol yöntemi uygulanabilir. Böylelikle uygulamamızın kaynak koduna müdahale etmeden güvenlik kontrolü ekleyebiliriz.

JAAS konusunda bilgilenmek isteyenler için maalesef Türkçe doküman yok. Ancak İngilizce olarak [şu bilgilendirici sunum](http://downloads.urbancode.com/presentations/jaas/DevelopingWithJAAS.ppt), [şu yazı](http://www.javaworld.com/javaworld/jw-09-2002/jw-0913-jaas.html), Sun sitesindeki [orijinal kılavuz](http://java.sun.com/j2se/1.4.2/docs/guide/security/jaas/JAASLMDevGuide.html) faydalı olabilir. Ayrıca [veritabanına bağlanan](http://www.javaworld.com/javaworld/jw-09-2002/jaas/jw-0913-jaas.zip), [akıllı kart okuyucudan bilgi alan](http://www.gemplus.com/techno/smartjaas/index.html), [parmak izi okuyucuya bağlanan](http://www.coresecuritypatterns.com/downloads/J2EEBiometrics.pdf) örnekler incelenebilir.