1.1.1 Kodda Bağlantı Cümleciğinde Açık Bir Şekilde Parola Bulundurulması (Hardcoded Password in Connection String) (CWE-547)

Açıklık Önem Derecesi: Orta

Açıklığın Etkisi: Hassas Bilgilere Yetkisiz Erişim, Bilgi İfşası, Sızma girişimlerinde saldırının boyutunun artması

Açıklığın Barındıran Dosyalar/Satırlar:

Proje Dosyası/Dosya Adı	Satır Numarası

Açıklığın Açıklaması:

Uygulama geliştiricileri uygulama kaynak koduna bağlantı cümlecikleri koyabilmektedirler. Bu bağlantı cümlecikleri veritabanı, smtp, ldap gibi bir başka unsura olan bağlantı kurma gereksinimi dolayısıyla yer alabilmektedir. Güvenliğe derinlikli defans (defence-in-depth) perspektifinden yaklaşacak olursak kaynak kodda bu tarz hassas satırların açık bir şekilde yer alması güvenlik riski teşkil etmektedir.

Uygulama güvenliğinde temel esas saldırganların içeriye hiçbir şekilde girememesidir. Fakat bazen güvenlik ne kadar üst seviyede olursa olsun saldırganın içeri girmesi devasa bir yüzeyde sadece bir boşluk yakalamasına bakabilmektedir. Dolayısıyla güvenliği daha garanti hale getirmenin yolu güvenliğe katman stratejisiyle yaklaşmaktan geçer. Bu ise örneğin uygulamanıza veya ağınıza gelecekte olası bir sızma girişimi yapıldığında şayet girişim başarılı olursa saldırganın içerideyken verebileceği zararı minimize edecek, belki de tam manasıyla sıfırlayacak iç katmanınızdaki güvenliğinizle mümkündür. Uygulama sunucunuza sızıldığı senaryoyu ele alacak olursak saldırgan örneğin uygulama kaynak kodunu okuyarak veritabanı kullanıcı adı ve şifre gibi bilgileri ele geçirebilir. Bu sayede belki sadece web sunucu içerisinde uygulamanın kök dizininde sıkışıp kalacakken şimdi edindiği bilgi doğrultusunda veritabanı sunucusuna atlama yapabilir. Veritabanı yazılımı üzerinden yetki ölçüsünde veritabanı servis haklarından sistem haklarına geçiş yapabilir ve uzanabildiği ölçüde bu şekilde atlamalar yapabilir.

Uygulamanın deployment metoduyla servis edildiği durumu ele alacak olursak ve saldırganın yine web sunucuya sızdığını düşünecek olursak saldırganın uygulama kök dizininde sadece deploy edilmiş web uygulamanızı görmesi onu engellemeyecektir. Saldırgan edindiği deploy

edilmiş uygulama dosyalarına tersine mühendislik araçları (örn; Microsoft Documents resmi web sitesinde bahsedilen Ildasm.exe (IL Disassembler) aracını) kullanarak içerlerinde şifre addedilebilecek string taraması yapabilir ve kritik verileri yine elde edebilir. Ayrıca eğer saldırgan parola bilgisini ele geçirecek olursa bu parola uygulamanın yeni bir sürümü derlenmediği sürece (yani parola kolayca değiştirilemeyeceğinden) kötüye kullanımı devam edecektir. Dahası eğer bu uygulama sayısız sisteme komple veya modül olarak dağıtılırsa sistemin çalınan parolası otomatik olarak dağıtılan tüm sistemlere girilmesine olanak verecektir. Dolayısıyla bu v.b. bilgi toplamalar ile saldırganların derine inme ihtimallerini engellemek / asgariye çekmek için kaynak kodlardaki hassas olan bağlantı cümleciklerinin saklanması önerilmektedir.

> Microsoft resmi sayfası der ki:
>
> "Uygulama kodlarına gömülen bağlantı string'leri güvenlik zafiyetlerine ve bakım problemlerine yol açabilmektedir. Uygulamanın kaynak kodu içerisinde derlenmiş şifrelenmemiş bağlantı string'leri Ildasm.exe (IL Disassembler) aracı kullanılarak görüntülenebilir. Dahası her bağlantı string değişimi gerektiği zaman uygulamanız tekrar tekrar derlenmek zorunda kalabilir. Bu nedenlerden dolayı uygulama içerisinde depolu olan bağlantı string'lerinin uygulama yapılandırma dosyasına taşınmasını önermekteyiz." (https://docs.microsoft.com, 03/30/2017)

Uygulama bağlantı cümleciklerinde parolalar açık metin olarak yer aldığında "Kodda Bağlantı Cümleciğinde Açık Bir Şekilde Parola Bulundurulması" (CWE-547) açıklık bulgusu olarak işaretlenirler. Uygulamalardaki bu açıklığa şöyle bir örnek verilebilir:

> Java:

```
// Veritabanı Bağlantı Cümleciğinde Güvensiz Hardcoded Veri

private Connection getConnection() {
    Connection conn = null;
    String urlJdbc = "jdbc:sqlserver://dbServer/appDb";

    conn = DriverManager.getConnection(urlJdbc, "sa", "mypass");

    return conn;
}
```

Bu örnekte veritabanı bağlantı cümleciği kaynak kodda açık bir şekilde yer almaktadır ve veritabanı kimlik doğrulama bilgilerini ifşa etmektedir. Bu güvensiz bir kullanımdır. Güvenli

hale dönüştürülmüş kullanım için iki yol vardır. Bunlardan ilki OS Entegre Kimlik Doğrulama ile veritabanı bağlantı cümleciğini kullanmadır:

Java:

```java
// YOL 1)   ENG: Connection String with Integrated Authentication
//          TR:  Güvenli OS Entegre Kimlik Doğrulama ile Veritabanı
//               Bağlantı Cümleciği

private Connection getConnection() {

    Connection conn = null;

    String urlJdbc = "jdbc:sqlserver://dbServer/appDb;" +
"integratedSecurity=true;authenticationScheme=JavaKerberos";

    conn = DriverManager.getConnection(urlJdbc);

    return conn;
}
```

Bu birinci güvenli kullanım yolunda uygulama veritabanı bağlantısı kurabilmek için Kerberos kimlik doğrulamasından yararlanmaktadır. Böylece veritabanına dair sunucu adresi, kullanıcı adı ve parola gibi hassas bilgilerin ifşasının önüne geçilmektedir.

İkinci güvenli kullanım yolu ise yapılandırma dosyası kullanmadır:

Java:

```
// YOL 2)   ENG: Read Credentials from Encrypted Properties
//          TR : Yapılandırma (Properties) Dosyasından Sunucu Adresi,
//               Kullanıcı Adı ve Şifreli Parolayı Okuyarak Güvenli
//               Veritabanı Bağlantı Cümleciği

private Connection getConnection() {
    Connection conn = null;

    try {
        Properties props = new Properties();
        FileInputStream fis = new FileInputStream(PROP_FILENAME);
        props.load(fis);
        fis.close();
    }
    catch (IOException e) {
        handleErrors(e);
    }

    String urlJdbc = "jdbc:sqlserver://" + props.getProperty(PROP_SERVERNAME)
+ "/" + props.getProperty(PROP_DBNAME);
    String username = props.getProperty(PROP_USERNAME);
    String password = decrypt(props.getProperty(PROP_PASSWORD));

    try {
        conn = DriverManager.getConnection(urlJdbc, username, password);
    }
    catch (SQLException e) {
        handleErrors(e);
    }

    return conn;
}
```

Bu güvenli veritabanı bağlantı cümleciği kullanımında öncelikle try bloğunda yapılandırma dosyası (bir .properties dosyası) açılmıştır. Ardından veritabanı bağlantı cümleciğinde veritabanı sunucu adresi, kullanıcı adı ve parola bilgileri daha önce açılan yapılandırma dosyasından deşifreleme yoluyla çekilmiştir ve bağlantı bu şekilde kurulmuştur.

Parola v.b. hassas veriler kaynak koda gömülmemelidirler. Diğer türlü kaynak kodda açık bir şekilde parola bulundurulması açıklığı şeklinde ele alınırlar. Bu açıklıktan her çeşit uygulama etkilenir. Fakat “özellikle” dağıtılan uygulamalar (masaüstü uygulama, router firmware uygulaması, firewall firmware uygulaması, yazıcı firmware uygulaması v.b.) veya açık kaynak kodlu uygulamalar daha çok etkilenir.

Eğer şu sorulardan herhangi bir tanesine evet deniyorsa kurum uygulama tarif edilen bu açıklıktan etkileniyordur:

- Kaynak koddaki hesap bilgileri hassas bir bileşene mi erişim izni veriyor (örn; Bir veritabanına, bir dosya depolamasına, bir API'ye veya bir servise) ?

- Kaynak koddaki hesap bilgileri production (yayın) ortamında mı kullanılıyor?

- Kaynak koddaki hesap bilglerini güncellemeden önce uygulamayı yeniden dağıtmak mı gerekiyor?

Uygulama kaynak kodlarında veritabanı bağlantı cümleciklerinin açık bir şekilde kodda barındığı tespit edilmiştir.

::::::::::::::::::::BULGU::::::::::::::::::::

Açıklığın Önlemi:

Eğer kullanılan veritabanı yazılımı "OS Entegre Sistem Kimlik Doğrulaması" veya Kerberos destekliyorsa uygulamada hardcoded (açık bir şekilde) kullanıcı adı ve parolayla veritabanına bağlanmak yerine bu kimlik doğrulamalar ile bağlanmak tercih edilmelidir.

Alternatif olarak veritabanı bağlantı cümleciklerini şifreli halde bir yapılandırma dosyasına taşıyıp deşifreleme yoluyla kaynak kodda kullanım da tercih edilebilir. Veritabanı bağlantı cümleciklerini yapılandırma dosyasına taşıma, entegre etme, şifreleme ve deşifreleme işlemleri için iki web teknolojisi (ASP.NET ve JAVA dili) üzerinden örnekler verilecektir.

Bu örnekler için yer alacak ana başlıklar şu şekildedir:

a) ASP.NET Web Uygulamalarında Veritabanı Bağlantı Cümleciklerindeki Kullanıcı Adı ve Parolaları Ayrı Yapılandırma Dosyasına Taşıma

i) SqlClient Sağlayıcısı (SQL Server Bağlantıları İçin)

ii) OracleClient Sağlayıcısı (Oracle Bağlantıları İçin)

iii) Oracle Data Access Sağlayıcısı (Oracle Bağlantıları İçin)

Web.Config Connection Strings Bloklarını Şifreleme (Encryption) ve Çözme (Decryption)

i) Web.Config Veri Bağlantılarını Şifreleme (Encryption)

ii) Web.Config Veri Bağlantılarını Çözme (Decryption)

Ek Başlıklar

a. Web.Config İçerisinde Birden Fazla Veri Bağlantı String'i Tanımlama

b. Harici Yapılandırma Dosyası Kullanma

b) Java (Struts, Spring, …) Web Uygulamalarında Veritabanı Bağlantı Cümleciklerindeki Kullanıcı Adı ve Parolaları Ayrı Yapılandırma Dosyasına Taşıma

Örneğin eğer kurum uygulama ASP.NET ve SQL Server kullanmaktaysa geliştirici şu başlıkları okumalıdır:

a) ASP.NET Web Uygulamaları

i) SqlClient Sağlayıcısı (SQL Server Bağlantıları İçin)

Web.Config Connection Strings Bloklarını Şifreleme (Encryption) ve

Çözme (Decryption)

Ek Başlıklar

a) ASP.NET Web Uygulamalarında Veritabanı Bağlantı Cümleciklerindeki Kullanıcı Adı ve Parolaları Ayrı Yapılandırma Dosyasına Taşıma

ASP.NET uygulamalarında web.config yapılandırma dosyasına eklenecek connection strings (bağlantı string'leri) bloğu şu şekilde bir şablona sahip olacaktır:

Web.config:

```
<?xml version='1.0' encoding='utf-8'?>
<configuration>
      ...  // size ait var olan kodlar
      <connectionStrings>
           <add name="İsim"
                providerName="Veri Sağlayıcısı İsmi"
                connectionString="Geçerli bir veritabanı bağlantı
string'i;" />
      </connectionStrings>
      ...   // size ait var olan kodlar
</configuration>
```

add Düğümü Özellikleri	Açıklaması
name	Bağlantı string'inin ismi. Örn; myConnection1
providerName	Kullanılan veritabanının .NET'teki sağlayıcısı
connectionString	Veritabanı bağlantı string'i (sql cümleciğinin işlevinde)

Tablo XXX. ADD Düğümü Aldığı Özellikler

Bu connectionStrings (bağlantı stringleri) bloğunu kurum c# kodlaması çekerek uygulama içerisinde kullanıyor olacaktır.

.NET platformu farklı farklı veritabanı sunucularıyla çalışabilir. Geliştiriciler her kullandıkları veritabanı ürünü için o ürüne özgü .NET sağlayıcısını ya da üçüncü taraf bir sağlayıcıyı projelerine dahil edebilirler. Microsoft .NET platformu için çeşitli sayıda resmi veritabanı sağlayıcısı geliştiricilere sunmuştur. Bunlar;

.NET Framework Veritabanı Sağlayıcıları	Açıklaması
.NET Framework'te SQL Server için Veri Sağlayıcısı	Microsoft SQL Server için veri erişimini sağlar. System.Data.sqlClient namespace'ini kullanır.
.NET Framework'te OLE DB için Veri Sağlayıcısı	OLE DB kullanarak veri kaynaklarını ortaya çıkarmayı sağlar. System.Data.OleDb namespace'ini kullanır.
.NET Framework'te ODBC	ODBC kullanarak veri kaynaklarını ortaya çıkarmayı sağlar. System.Data.Odbc namespace'ini kullanır.
.NET Framework'te Oracle	Oracle Veritabanı Sunucusu için veri erişimini sağlar. Oracle Veri Sağlayıcısı için .NET Framework veri sağlayıcısı Oracle Client modülü 8.1.7 ve sonrasını destekler. Bu sağlayıcı System.Data.OracleClient namespace'ini kullanır.

Tablo XXX. .NET'te Veritabanı Sağlayıcıları

Burada SQL Server ve Oracle veritabanı sunucuları için C# connection string'lerinin (bağlantı string'lerinin) yapılandırma dosyasına taşınması, uygulamanın yapılandırma dosyasıyla entegre çalışabilmesi ve nihayetinde yapılandırma dosyasında barınan connection string'lerin (bağlantı string'lerinin) şifrelenmesi işlemleri gösterilecektir.

i) SqlClient Sağlayıcısı (SQL Server Bağlantıları İçin)

> Veritabanı bağlantısı kurulan kurum C# dosyalarının en başına bu iki kütüphane eklenir.

```
using System.Web.Configuration;  // Web.config'den veritabanı
```

```
                                    // bağlantı bilgilerini almaya
                                    // yarar
using System.Data.SqlClient;        // Uygulamada eğer SQL Server için
                                    // SqlClient kullanıyorsanız
```

> Web.Config Dosyası içerisindeki <configuration> etiketleri arasına şu bloğu yerleştirilir.

```
<connectionStrings>
      <add
           name="veritabaniBaglantisi1"
           connectionString="Server=myServerAddress;
                Database=myDataBase; User ID=myUsername;
                Password=myPassword; Integrated Security=False;"
           providerName= "System.Data.SqlClient"
      />
</connectionStrings>
```

> Daha önce veritabanı bağlantısı cümleciklerinin kullanıldığı kurum C# dosyalarının ilgili o sql satırları yerine şu satır konulur.

```
SqlConnection con = new SqlConnection(
WebConfigurationManager.ConnectionStrings["veritabaniBaglantisi1"].
ConnectionString);
```

Burada web.config içerisine yerleştirilen connectionStrings bloğunda geliştiriciler tarafından doldurulması gereken üç özellik yer almaktadır: name, connectionString ve providerName. Name özelliğine bağlantı string'inin işlevini tanımlayıcı rasgele (örn; conn1 gibi) bir değer verilebilir. Bu değer kurum c# kodlamasında web.config'deki connection string bloğunu çekmek için kullanılacaktır. providerName özelliğine halihazırda kullanmakta olunan veri sağlayıcısı ismi (yani bu örnekte System.Data.SqlClient) girilir. ConnectionString özelliğine ise temel bir veritabanı bağlantısı için Server (sunucu adresi) değeri, Database (veritabanı ismi) değeri, User ID (veritabanı kullanıcısı adı) değeri, Password (veritabanı kullanıcısı şifresi) değeri ve Integrated Security (veritabanı sunucusuna olan bağlantının windows sistem hesap bilgileriyle mi yoksa veritabanı hesap bilgileriyle mi yapılacağı ayar) değeri -

şimdilik false olarak girilmesi - gerekir. <add düğümünün connectionString özelliği hangi anahtar kelimeleri alabiliyor bilgisine

```
https://www.connectionstrings.com/all-sql-server-connection-string-keywords/
```

adresinden ulaşılabilir.

ii) OracleClient Sağlayıcısı (Oracle Bağlantıları İçin)

(-) Uyarı

.NET uygulamalarının Oracle veritabanı sunucularıyla olan bağlantı kurma sağlayıcısı OracleClient bir .NET ürünüdür. Microsoft bu ürününe olan desteğini yakın gelecekte çekeceğinden artık .NET uygulama geliştirilerinin Oracle veritabanı sunusuyla bağlantı kurmak için üçüncü taraf (third party) sağlayıcılara geçilmesini önermektedir. Şu an halihazırda kullanılmakta olan .NET uygulamaları bu durumdan belli müddet etkilenmeyecektir, fakat yeni geliştirilen uygulamalar için üçüncü taraf sağlayıcılar önerilmektedir.

> Veritabanı bağlantısı kurulan kurum C# dosyalarının en başına bu iki kütüphane eklenir.

```
using System.Web.Configuration;   // Web.config'den veritabanı
                                  // bağlantı bilgilerini almaya
                                  // yarar.
using System.Data.OracleClient;   // Uygulamanızda eğer Oracle için
                                  // OracleClient kullanıyorsanız
```

> Web.Config Dosyası içerisindeki <configuration> etiketleri arasına şu bloğu yerleştirilir.

```
<configuration>
<connectionStrings>
      <add
           name="veritabaniBaglantisi1"
           providerName="System.Data.OracleClient"
           connectionString="User Id=scott;Password=tiger;Data
                Source=inst1;Integrated Security=False;"
```

```
</connectionStrings>
</configuration>
```

> Daha önce veritabanı bağlantısı cümleciklerinin kullanıldığı kurum C# dosyalarının ilgili o sql satırları yerine şu satır konulur.

```
SqlConnection con = new SqlConnection(
WebConfigurationManager.ConnectionStrings["veritabaniBaglantisi1"].
ConnectionString);
```

Burada web.config içerisine yerleştirilen connectionStrings bloğunda tarafınızca doldurulması gereken üç özellik yer almaktadır: name, connectionString ve providerName. Name özelliğine bağlantı string'inin işlevini tanımlayıcı rasgele (örn; conn1 gibi) bir değer verebilirsiniz. Bu değer c# kodlamanızda web.config'deki connection string bloğunu çekmek için kullanılacaktır. providerName özelliğine halihazırda kullanmakta olduğunuz veri sağlayıcısı ismini (yani bu örnekte System.Data.OracleClient'ı) girmektesiniz. ConnectionString özelliğine ise temel bir veritabanı bağlantısı için Server (sunucu adresi) değeri, Database (veritabanı ismi) değeri, User ID (veritabanı kullanıcısı adı) değeri, Password (veritabanı kullanıcısı şifresi) değeri ve şimdilik Integrated Security (veritabanı sunucusuna olan bağlantının windows sistem hesap bilgileriyle mi yoksa veritabanı hesap bilgileriyle mi yapılacağı ayar) değerini - şimdilik false olarak girilmesi - yeterlidir. <add düğümünün connectionString özelliği hangi anahtar kelimeleri alabiliyor bilgisine

```
https://docs.microsoft.com/tr-
tr/dotnet/api/system.data.oracleclient.oracleconnectionstringbuilde
r?view=netframework-4.7.2#%C3%B6zellikler
```

adresinden ulaşılabilir.

iii) Oracle Data Access Sağlayıcısı (Oracle Bağlantıları İçin)

(+) Uyarı

Oracle firması tarafından resmi olarak .NET platformu için geliştirilmiş veri bağlantı

sağlayıcısı Oracle Data Access Provider (ODP.NET) artık tercih edilmektedir. Bu resmi

sağlayıcı Microsoft'a ait .NET platformu için belki third party (üçüncü taraf) harici bir

modül konumundadır, fakat Microsoft'un artık Oracle sunucular için yönlendirdiği yöntem de bu v.b. üçüncü taraf sağlayıcıların kullanılması yönündedir. ODP.NET Modülü indirme linki https://www.oracle.com/technetwork/topics/dotnet/downloads/index.html şeklindedir.

> Veritabanı bağlantısı kurulan kurum C# dosyalarının en başına bu üç kütüphane eklenir.

```
using System.Web.Configuration;  // Web.config'den veritabanı
                                 // bağlantı bilgilerini almaya
                                 // yarar
using Oracle.DataAccess.Client;  // Uygulamanızda eğer Oracle için
                                 // Data Access kullanıyorsanız
using Oracle.DataAccess.Types;   // Uygulamanızda eğer Oracle için
                                 // Data Access kullanıyorsanız
```

> Web.Config dosyası içerisindeki <configuration> etiketleri arasına şu ayar bloğu yerleştirilir.

```
<configuration>
      <connectionStrings>
           <add name="veritabaniBaglantisi1"
                providerName="Oracle.DataAccess.Client"
                connectionString="UserId=scott; Password=tiger;
                      DataSource=inst1; Pooling=False"/>
      </connectionStrings>
</configuration>
```

> Daha önce veritabanı bağlantısı cümleciklerinin kullanıldığı kurum C# dosyalarının ilgili o sql satırları yerine şu satır konur.

```
SqlConnection con = new SqlConnection(
WebConfigurationManager.ConnectionStrings["veritabaniBaglantisi1"].
ConnectionString);
```

Burada web.config içerisine yerleştirilen connectionStrings bloğunda tarafınızca doldurulması gereken üç özellik yer almaktadır: name, connectionString ve providerName. Name özelliğine bağlantı string'inin işlevini tanımlayıcı veya rasgele (örn; conn1 gibi) bir değer verebilirsiniz. Yalnız bu değer c# kodlamanızda bloğu çekmek için kullanacağınız bir değer olacaktır. providerName özelliğine halihazırda kullanmakta olduğunuz veri sağlayıcısı ismini (yani bu örnekte Oracle.DataAccess.Client'ı) girmektesiniz. ConnectionString özelliğine ise temel bir veritabanı bağlantısı için Data Source (yani örn; sunucu adresi:port/sid) değeri, User ID (veritabanı kullanıcısı adı) değeri, Password (veritabanı kullanıcısı şifresi) değeri ve Pooling (Sql bağlantısı nesnesinin kullanım sonrası bağlantı havuzundan otomatikmen kaldırılması ya da kaldırılmaması ayarı) değeri - şimdilik false olarak belirlenmesi - yeterlidir. <add düğümünün connectionString özelliği Oracle Data Access Provider için hangi anahtar kelimeleri alabiliyor bilgisine

```
https://docs.oracle.com/cd/B28359_01/win.111/b28375/featConnecting.htm
```

adresinden ulaşılabilir.

Oracle Access Data sağlayıcıda yerel ve uzak veritabanı bağlantısı için iki farklı connectionStrings Data Source kullanımı vardır:

x) Oracle Sunucusu Yerel Sistemdeyken (Yani Oracle Sunucusu Web Uygulaması ile Aynı Sunucudayken)

```
<add name="veribaglanti1"
     providerName="Oracle.DataAccess.Client"
     connectionString="UserId=scott; Password=tiger;
           DataSource="localhost:PORT/DBNAME";Pooling=false" />
```

y) Oracle Sunucusu Uzak Bir Sistemdeyken (Yani Oracle Sunucusu Web Uygulaması ile Aynı Sunucuda Değilken)

y.1) Uzak Oracle Sunucusu Bir Takma Ada (Alias'a) Sahipken

```
<add name="veribaglanti1"
     providerName="Oracle.DataAccess.Client"
     connectionString="User Id=scott; Password=tiger;
       DataSource=TNSName; Pooling=false" />
```

y.2) Uzak Oracle Sunucusu Bir TNS Ada (Takma Ada) Sahip Değilken

```
<add name="veribaglanti1"
     providerName="Oracle.DataAccess.Client"
     connectionString="UserId=scott; Password=tiger; Data
       Source = (DESCRIPTION=(ADDRESS_LIST = (ADDRESS =
       (PROTOCOL = TCP)(HOST=X.Y.Z.T) (PORT=portNumber)))
       (CONNECT_DATA = (SERVER=DEDICATED(SERVICE_NAME =
       myDBName))); Pooling=false" />
```

Not: TNS Adı, diğer adıyla Servis Adı (Oracle Sunucusu Takma Adı) oluşturarak TNS adı üzerinden connection string'leri kullanmak için oracle sunucusundaki tnsnames.ora dosyasına şu şekilde bir şablona sahip tanımlamada bulunmak gerekir.

tnsnames.ora:

```
kullanıcıHesaplari = (DESCRIPTION = (ADDRESS = (PROTOCOL = tcp)
(HOST = X.Y.Z.T)(PORT = 1521))(CONNECT_DATA = (SERVICE_NAME =
kullanicilar@X.Y.Z.T)))
```

Host yerine oracle sunucusu IP adresi gelir. Port kısmına oracle sunucusundaki oracle servisinin port numarası gelir. Service_name kısmına ise oracle veritabanı ismi gelir.

Web.Config Connection Strings Bloklarını Şifreleme (Encryption) ve Çözme (Decryption)

i) Web.Config Veri Bağlantılarını Şifreleme (Encryption)

Web.Config yapılandırma dosyasındaki blokları (bu senaryo için connection strings bloklarını) şifrelemek için asp.net framework'ünün bir aracı olan aspnet_regiis.exe 'den faydalanılabilir. Bu aracı sorunsuz kullanabilmek için CMD penceresinin yönetici olarak çalıştırılması gerekir.

Başlat:

```
CMD (Sağ Tık -> Yönetici Olarak Çalıştır)
```

Ardından aspnet_regiis.exe dosyasının yer aldığı dizin bulunmalıdır.

CMD (as Administrator):

```
C:\Windows\system32\> dir /s c:\aspnet_regiis.exe
```

Sıralanan aspnet_regiis.exe araç dizin yollarından .NET framework versiyonu en yüksek olan dizin yoluna gidilmelidir. Örn;

CMD (as Administrator):

```
C:\Windows\system32\> cd "C:\Windows\Microsoft.NET\Framework64\
4.0.30319"
```

Ardından aspnet_regiis.exe aracı verilecek -pef parametresi ve arguman değerleriyle çalıştırılmalıdır:

CMD (as Administrator):

```
// Şifreleme (Encryption)
C:\Windows\Microsoft.NET\Framework64\4.0.30319>
aspnet_regiis.exe -pef "connectionStrings" "C:\inetpub\wwwroot"
```

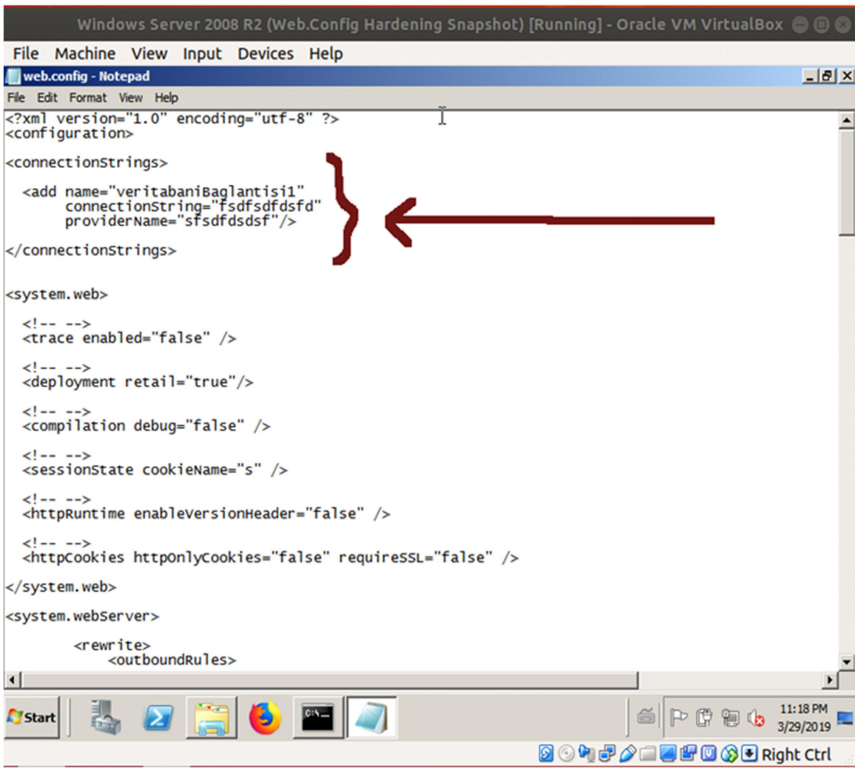
Çıktı:

Encrypting configuration section...

Succeeded!

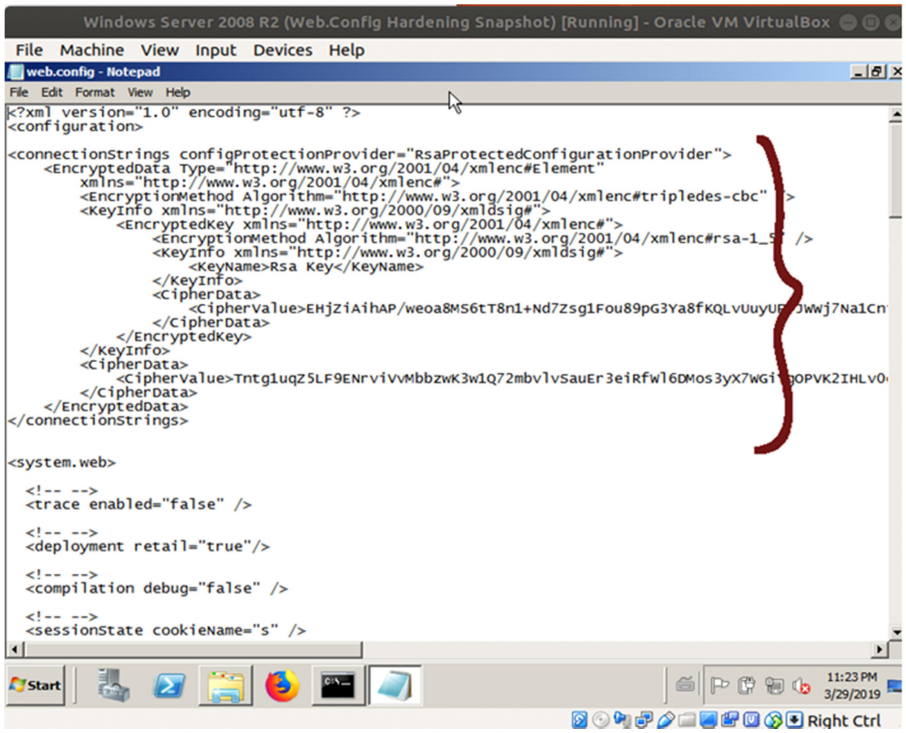
Yukarıdaki kodlama ile C:\inetpub\wwwroot dizininde yer alan web.config dosyası içerisindeki <connectionStrings> bloğunun AES algoritması ile şifrelenmesi sağlanır. Örneğin;

~ Şifreleme Öncesi ~



Şekil XXX. Yapılandırma Dosyasında Açık Tutulan Bir Bağlantı String Bloğu Gösterimi

~ Şifreleme Sonrası ~



Şekil XXX. Yapılandırma Dosyasında Şifreli Halde Tutulan Bir Bağlantı String Bloğu Gösterimi

Web.Config dosyasına şifrelenmiş connection strings'i c# kodlaması içerisinde kullanabilmek için

C# :

```
ConnectionStringsSection encyptedConnectionStrings =
WebConfigurationManager.OpenWebConfiguration().GetSection("connecti
onStrings") as ConnectionStringsSection;


      ConnectionStringSettings decryptedConnectionStrings =
encryptedConnectionStrings.SectionInformation.UnprotectSection() as
ConnectionStringSettings;


      SqlConnection con = new
SqlConnection(decryptedConnectionStrings);
```

şeklinde sırasıyla web.config yapılandırma dosyasının açılması, connectionStrings bloğunun cımbızlanması, sonra bu bloğun çözülmesi (decrypt edilmesi) ve son olarak çözülen connection string'in sql bağlantı kurma metoduna koyulması gerekmektedir.

ii) Web.Config Veri Bağlantılarını Çözme (Decryption)

Web.Config yapılandırma dosyasında şifrelenmiş (encypt edilmiş) connection strings bloklarını düzenlemek / modifiye etmek maksadıyla çözmek için (decypt etmek için) yine aspnet_regiis.exe aracı kullanılabilir:

Not:

Şifrelenmiş (encrypt edilmiş) bir web.config dosyası sadece şifrelendiği makinada çözülebilir (decrypt edilebilir).

Bu işlem için öncelikle CMD penceresinin yönetici olarak çalıştırılması gerekir.

Başlat:

```
CMD (Sağ Tık -> Yönetici Olarak Çalıştır)
```

Ardından aspnet_regiis.exe dosyasının yer aldığı dizin tespit edilir:

CMD (as Administrator):

```
C:\Windows\system32\> dir /s c:\aspnet_regiis.exe
```

Sıralanan aspnet_regiis.exe araç dizin yollarından .NET framework versiyonu en yüksek olan dizin yoluna gidilir. Örn;

CMD (as Administrator):

```
C:\Windows\system32\>
cd "C:\Windows\Microsoft.NET\Framework64\4.0.30319"
```

Ardından aspnet_regiis.exe aracı verilecek -pdf parametresi ve arguman değerleriyle çalıştırılır:

CMD (as Administrator):

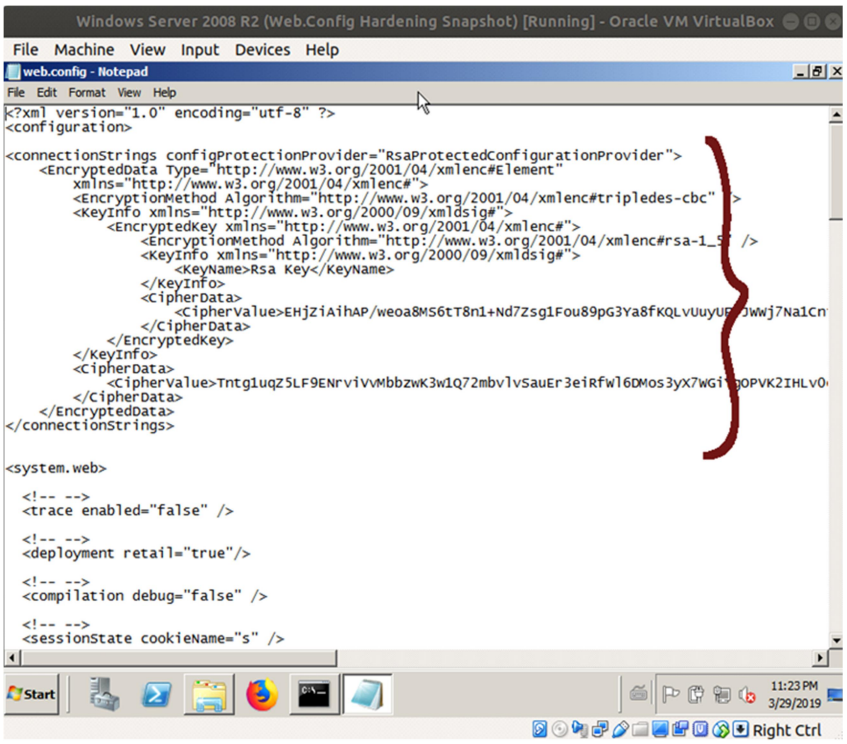
```
C:\Windows\Microsoft.NET\Framework64\4.0.30319>
aspnet_regiis.exe -pdf "connectionStrings" "C:\inetpub\wwwroot"
```

Çıktı:

Decrypting configuration section...

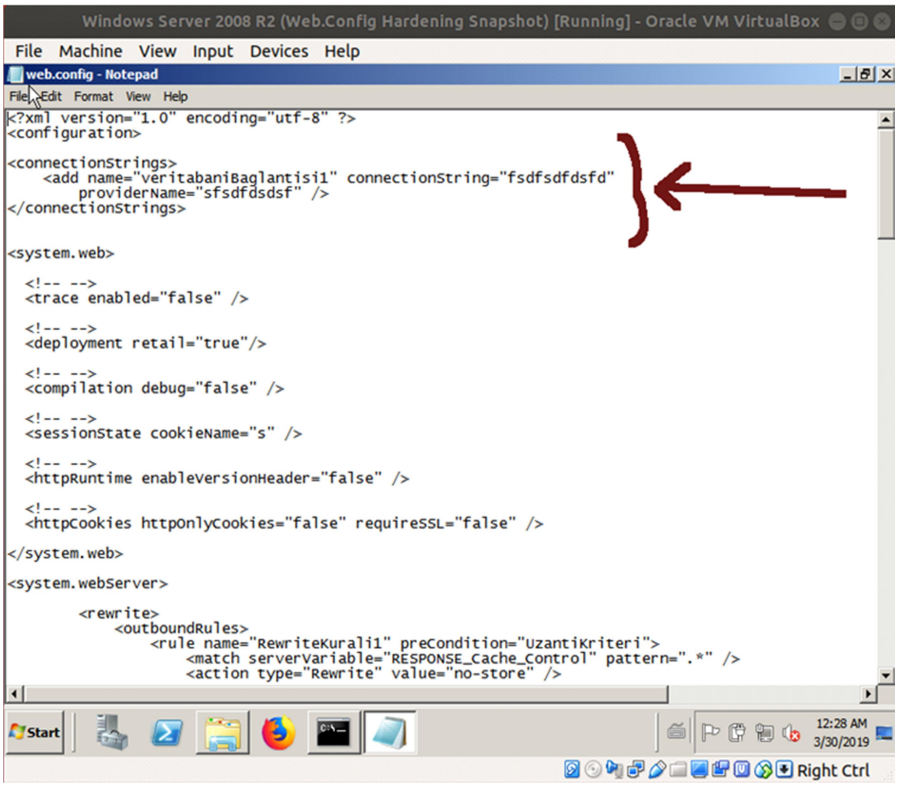
Succeeded!

~ Şifrelenmiş Bloğu Çözmeden Önce ~



Şekil XXX. Yapılandırma Dosyasında Şifreli Halde Tutulan Bağlantı String Bloğu

~ Şifrelenmiş Blok Çözüldüğünde ~



Şekil XXX. Yapılandırma Dosyasında Şifresi Çözülmüş Bağlantı String Bloğu Gösterimi

Ek:

a. Web.Config İçerisinde Birden Fazla Veri Bağlantı String'i Tanımlama

Web uygulaması içerisinde birden fazla çeşitte (rolde) veritabanı bağlantısı kuruluyor olabilir. Bu durumda web.config yapılandırma dosyasında her biri için ayrı ayrı connection strings (veri bağlantı string'leri) tanımlaması yapılabilmektedir ve C# ile bu bağlantılar çekilebilmektedir.

Örneğin web.config yapılandırma dosyasına 3 adet veri bağlantısı string tanımlamasının eklendiği örnek gösterilmiştir:

```
<configuration>
      <connectionStrings>
         <add name="conn1"
              providerName="Veri Sağlayıcısı İsmi"
              connectionString= "Geçerli Bir Veri Bağlantısı
               String'i 1" />
```

```
        <add name="conn2"
             providerName="Veri Sağlayıcısı İsmi"
             connectionString="Geçerli Bir Veri Bağlantısı
              String'i 2" />
        <add name="conn3"
             providerName="Veri Sağlayıcısı İsmi"
             connectionString="Geçerli Bir Veri Bağlantısı
              String'i 3" />
    </connectionStrings>
</configuration>
```

Bu durumda kurum C# kodlamasında şu şekilde bir kullanımda bulunulması gerekir.

```
var conn1 = ConfigurationManager.ConnectionStrings["conn1"].
ConnectionString;

var conn2 = ConfigurationManager.ConnectionStrings["conn2"].
ConnectionString;

var conn3 = ConfigurationManager.ConnectionStrings["conn3"].
ConnectionString;
```

b. Harici Yapılandırma Dosyası Kullanma

Web.Config içerisine yerleştirilen bloklar dilenirse harici bir dosyaya konulup web.config içerisinde dahil edilmek suretiyle kullanılabilir. Bunun avantajı deploy edilen web uygulamalarının web.config dosyası içerisinde değişiklik yapılması ihtiyacı duyulduğunda değişikliğin uygulanabilmesi için uygulamanın tekrar deploy edilip sunucuya konulmasını mahal bırakmadan dinamik olarak yapılan her değişikliğin deploy edilmiş web uygulamasında uygulanabilmesini sağlamasıdır. Örneğin veri bağlantı string'leri zaman zaman değiştirilme ihtiyacı duyulan bir yapılandırma ayarıdır. Bu nedenle bu yapılandırma ayarının (<connectionStrings> ... </connectionStrings> bloklarının) harici bir yapılandırma dosyasına taşınıp web.config içerisine dahil edilmek suretiyle kullanımı tercih edilebilir.

Harici yapılandırma dosyasına connection string'leri taşıma ve o şekilde kullanma istenirse hazırlanacak harici yapılandırma dosyası web.config'deki gibi tanımlamaları içermeden yalnızca dahil edilmek istenen bloğu içerecek şekilde ayarlanmalıdır. Örn;

C:\inetpub\wwwroot\connections.config İçeriği:

```
<connectionStrings>
      <add name="Name"
           providerName="Veri Sağlayıcı İsmi"
           connectionString="Geçerli Bir Veri Bağlantısı String'i;"
      />
</connectionStrings>
```

Bu harici yapılandırma dosyası web.config ana yapılandırma dosyası içerisine ise şu şekilde dahil edilir:

C:\inetpub\wwwroot\web.config İçeriği:

```
<?xml version='1.0' encoding='utf-8'?>
<configuration>
      ...  // size ait var olan kodlar
      <connectionStrings configSource="connections.config"/>
      ...  // size ait var olan kodlar
</configuration>
```

b) Java (Struts, Spring, …) Web Uygulamalarında Veritabanı Bağlantı Cümleciklerindeki Kullanıcı Adı ve Parolaları Ayrı Yapılandırma Dosyasına Taşıma

Java web uygulamalarında kaynak koddaki hassas veriler kısıtlanmış izne sahip .properties uzantılı oluşturulacak dosyalara taşınmalıdırlar ve hassas bu veriler .properties dosyası içerisinde şifrelenmiş halde tutulmalıdırlar. Hassas verilerin şifrelenmesi ve daha sonra uygulama çalışırken dinamik olarak şifrelerinin çözülmesi için geliştiriciler teknik detaya inmeden (zaman kaybetmeden) kolay bir şekilde açık bir kullanım arayüzüne (yani metotlarına) sahip Jasypt adlı kütüphaneden yararlanabilirler.

Jasypt Veri Şifreleme / Çözme Kütüphanesi

```
http://www.jasypt.org/download.html
```

Apache Commons Configuration Kütüphanesi (Jasypt'in ihtiyaç duyduğu kütüphane)

```
https://commons.apache.org/proper/commons-
configuration/download_configuration.cgi
```

Jasypt kütüphanesinin şifreleme ve çözme işlemlerinin bir örneğine yer verilmiştir ve kodların ilgili alanlarında yorum satırları ile kodun anlamı ifade edilmiştir:

```java
package aboutEncryption;

// Jasypt Veri Sifreleme ve Cozme Islemleri
// Icin Kullanilan (Gerek Duyulan) Kutuphaneler
import org.apache.commons.configuration2.builder.FileBasedConfigurationBuilder;
import org.apache.commons.configuration2.builder.fluent.Parameters;
import org.apache.commons.configuration2.convert.DefaultListDelimiterHandler;
import org.apache.commons.configuration2.ex.ConfigurationException;
import org.apache.commons.configuration2.PropertiesConfiguration;

// Java uygulamalarinda Veri Sifreleme
// ve Cozme Icin Kullanilan Bir Kutuphane
import org.jasypt.encryption.pbe.StandardPBEStringEncryptor;

// Ekstra
import org.apache.commons.logging.Log;

public class EncryptDecrypt {
    private final String propertyFileName;
    private final String propertyKey;
    private final String isPropertyKeyEncrypted;

    final String decryptedUserPassword;

    public EncryptDecrypt(String pPropertyFileName,String pUserPasswordKey, String pIsPasswordEncryptedKey) throws Exception {
        this.propertyFileName = pPropertyFileName;
        this.propertyKey = pUserPasswordKey;
        this.isPropertyKeyEncrypted = pIsPasswordEncryptedKey;
        try {
            encryptPropertyValue();
        } catch (ConfigurationException e) {
            throw new Exception("Problem encountered during encryption process",e);
        }
        decryptedUserPassword = decryptPropertyValue();

    }

    private void encryptPropertyValue() throws ConfigurationException {
        System.out.println("Starting encryption operation");
        System.out.println("Start reading properties file");

        // Properties Dosyasini Açma (( Guncel Kisim ))
        FileBasedConfigurationBuilder<PropertiesConfiguration> builder =
            new FileBasedConfigurationBuilder<PropertiesConfiguration>(PropertiesConfiguration.class)
                    .configure(new Parameters().properties()
                        .setFileName(propertyFileName)
                        .setThrowExceptionOnMissing(true)
                        .setListDelimiterHandler(new DefaultListDelimiterHandler(';'))
                        .setIncludesAllowed(false));

        PropertiesConfiguration config = builder.getConfiguration();

        // Parola zaten sifreli mi degil mi bilgisini görmek
        // için properties boolean bayragindaki veriyi çekme
        String isEncrypted = config.getString(isPropertyKeyEncrypted);

        //Parola sifreli mi degil mi kontrolü
        if(isEncrypted.equals("false")){
            String tmpPwd = config.getString(propertyKey);

            //Sifreleme Nesnesi
            StandardPBEStringEncryptor encryptor = new StandardPBEStringEncryptor();

            // Sifreleme için Tuz (Salt)
            encryptor.setPassword("jasypt");

            // Sifrelemenin Uygulanmasi
            String encryptedPassword = encryptor.encrypt(tmpPwd);

            // Properties dosyasindaki parolanin üzerine sifrelenmis halini yazma
            config.setProperty(propertyKey, encryptedPassword);

            // Boolean bayragi sifrelemenin halihazirda zaten yapilmis oldugu bilgisini belirtmek için true yapilir.
            config.setProperty(isPropertyKeyEncrypted,"true");

            // Properties dosyasi kaydedilir
            builder.save();
        }else{
             System.out.println("User password is already encrypted.\n ");
        }
    }

    private String decryptPropertyValue() throws ConfigurationException {
         System.out.println("Starting decryption");

         // Properties Dosyasini Açma (( Guncel Kisim ))
         FileBasedConfigurationBuilder<PropertiesConfiguration> builder =
             new FileBasedConfigurationBuilder<PropertiesConfiguration>(PropertiesConfiguration.class)
                     .configure(new Parameters().properties()
                         .setFileName(propertyFileName)
                         .setThrowExceptionOnMissing(true)
                         .setListDelimiterHandler(new DefaultListDelimiterHandler('\n'))
                         .setIncludesAllowed(false));

         PropertiesConfiguration config = builder.getConfiguration();

         // Sifreli parolayi çekme
         String encryptedPropertyValue = config.getString(propertyKey);

         // Sifreleme nesnesini olusturma
         StandardPBEStringEncryptor encryptor = new StandardPBEStringEncryptor();

         // Parola sifrelemede kullanilan tuzu (salt'i) girme
         encryptor.setPassword("jasypt");

         // Parolanin sifresini çözme
         String decryptedPropertyValue = encryptor.decrypt(encryptedPropertyValue);

         return decryptedPropertyValue;
    }
```

Şekil XXX. Jasypt Kullanımı I

```java
package aboutEncryption;

import org.junit.Test;
import org.junit.Assert;

public class EncryptDecryptTest {
    public EncryptDecryptTest() {
    }

    @Test
    public void encryptDecryptTest() throws Exception {
        //Sifrelenecek Properties Dosyasi Ismi
        String propertyFileName = "veritabaniAdminBilgileri.properties";

        //Properties dosyasindaki anahtar sözcük (sol yanda yer alan)
        String userPwdKey = "database.user.password" ;

        //Properties dosyasindaki sifrenin halihazirda sifreli durumda olup
        //olmadigini bize söyleyecek anahtar sözcük
        String isPwdEcnryptedKey = "is.database.user.password.encrypted";

        //Invoke the constrsuctor
        EncryptDecrypt app = new EncryptDecrypt(propertyFileName,userPwdKey,isPwdEcnryptedKey);

        //Sifresi Çözülmüs Parola
        String result = app.decryptedUserPassword;
        Assert.assertTrue("Decrypted password should be \"myPassWord\"", result.equalsIgnoreCase("myPassWord"));
    }

}
```

Şekil XXX. Jasypt Kullanımı II

Kullanıcı Adı ve Şifre gibi hassas verilerin taşınacağı properties dosyasının formatı bu örnek için şu şekilde olmalıdır:

~ Şifreleme Öncesi ~

C:\veritabaniAdminBilgileri.properties :

```
database.user.name=jack

database.user.password=richer234

is.database.user.password.encrypted=false
```

~ Şifreleme Sonrası ~

C:\veritabaniAdminBilgileri.properties :

```
database.user.name=jack

database.user.password=Gsi3ZNTtven+WjExGA1Q3UD1brweHOqf

is.database.user.password.encrypted=true
```

Yukarıdaki kodun proje geliştiricilerince uyarlanarak kullanılması gerekmektedir. Bu şekilde kaynak kodda açık hassas veri bulundurulmasının önüne geçilerek karşılaşılabilecek felaket senaryolarına karşı zarar görülmesine sebep olabilecek unsurlardan bir tanesi giderilmiş olacaktır.

Referanslar:

1. https://www.owasp.org/index.php/Preventing_LDAP_Injection_in_Java
2. https://www.webguvenligi.org/wp-content/uploads/2007/11/authenticationtrk.pdf
3. https://www.owasp.org/index.php/Use_of_hard-coded_password
4. https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/connection-strings-and-configuration-files
5. https://stackoverflow.com/questions/4227921/how-can-we-have-two-connection-strings-in-web-config-and-switch-betweeen-them-in
6. https://stackoverflow.com/questions/5642474/setting-up-connection-string-in-asp-net-to-sql-server
7. https://www.c-sharpcorner.com/UploadFile/7d3362/various-ways-to-specify-connection-string-in-Asp-Net-web-app/
8. https://www.youtube.com/watch?v=v1EWUAYNA1g
9. https://www.youtube.com/watch?v=Gf3kFBAovXw
10. https://stackoverflow.com/questions/15365210/failed-to-encrypt-the-section-connectionstrings-using-provider-rsaprotectedco
11. https://superuser.com/questions/401495/equivalent-of-unix-find-command-on-windows
12. https://docs.microsoft.com/en-us/previous-versions/aspnet/hh8x3tas(v%3Dvs.100)
13. https://www.youtube.com/watch?v=lzm_uTW9Ug8
14. http://www.sortedset.com/how-to-encrypt-decrypt-a-password-stored-in-a-properties-file-with-java-jasypt-apache-commons-configuration/
15. https://github.com/sortedset/jasyptblog
16. https://commons.apache.org/proper/commons-configuration/userguide/upgradeto2_0.html
17. https://commons.apache.org/proper/commons-configuration/apidocs/org/apache/commons/configuration2/ex/ConfigurationException.html
18. https://commons.apache.org/proper/commons-configuration/apidocs/org/apache/commons/configuration2/PropertiesConfiguration.html
19. https://www.programcreek.com/java-api-examples/?class=org.apache.commons.configuration2.builder.FileBasedConfigurationBuilder&method=save
20. https://stackoverflow.com/questions/22541455/warning-the-method-assertequals-from-the-type-assert-is-deprecated
21. https://www.tutorialspoint.com/junit/junit_suite_test.htm
22. http://www.jasypt.org/dependencies.html
23. https://www.youtube.com/watch?v=2xA82u3Q84M
24. http://docs.vizrt.com/viz-content-pilot-guide/5.6/configuration_database_service_names_and_sid.html
25. https://docs.oracle.com/database/121/ODPNT/featConnecting.htm#ODPNT199
26. https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/oracle-and-adonet
27. https://www.oreilly.com/library/view/adonet-cookbook/0596004397/ch01s09.html
28. https://docs.microsoft.com/tr-tr/dotnet/framework/data/adonet/connection-string-syntax#sql-server-authentication-with-sqlclient
29. https://www.c-sharpcorner.com/UploadFile/nipuntomar/connection-strings-for-oracle/
30. https://www.oracle.com/webfolder/technetwork/tutorials/obe/db/12c/r1/appdev/dotnet/Web_version_Fully_Managed_ODPnet_OBE/odpnetmngdrv.html
31. https://stackoverflow.com/questions/46408085/oracle-connection-in-c-sharp-connection-string
32. https://stackoverflow.com/questions/8202999/encrypt-a-connection-string-how
33. https://stackoverflow.com/questions/8195099/java-how-to-store-password-used-in-application/8195195#8195195
34. https://www.appmarq.com/public/security,1025048,Avoid-hard-coded-password-in-connection-string

35. https://rules.sonarsource.com/typescript/RSPEC-2068
36. https://www.appmarq.com/public/tqi,1020820,Avoid-hardcoded-passwords-TypeScript
37. https://docs.snyk.io/scan-application-code/snyk-code/security-rules-used-by-snyk-code/javascript-and-typescript
38. https://rules.sonarsource.com/php/RSPEC-2068
39. https://offensive360.com/how-to-prevent-hardcoded-passwords/