1.1.1 Bağlantı Cümleciği Enjeksiyonu (Connection String Injection) (CWE-99)

Açıklık Önem Derecesi: Yüksek

Açıklığın Etkisi: Servis Dışı Bırakma, a

Açıklığın Barındıran Dosyalar/Satırlar:

Proje Dosyası/Dosya Adı	Satır Numarası

Açıklığın Açıklaması:

Uygulamalar bir veritabanı veya harici bir sunucu (örn; Active Directory) ile iletişim kurmak için dinamik bağlantı cümlecikleri (connection strings) oluşturabilirler. Dinamik bağlantı cümlecikleri girdilerin bağlantı cümleciklerine eklenmesiyle oluşan bağlantı cümleciklerine denir. Uygulamalar dinamik bağlantı cümleciklerine istemci taraftan gelen girdileri dahil edebilirler. Böylesi bir durumda güvensiz girdiler kısıtlanmadığı için veya uygun şekilde filtrelenmediği için bağlantı cümleciğini zararlı bir şekilde manipüle edebilir. Bu açıklığa "Bağlantı Cümleciği Enjeksiyonu (CWE-99)" adı verilir.

Bir saldırgan eğer uygulamanın veri tabanı sunucusuyla iletişim halinde olan bağlantı cümleciğini manipüle edebilir durumdaysa bu açıklık yoluyla başlıca şu saldırıları düzenleyebilir:

- Uygulama performansına zarar verme (MIN POOL SIZE değerini arttırarak)
- Ağ bağlantısını kurcalama (örn; TRUSTED CONNECTION üzerinden)
- Uygulamayı saldırganın sahte veri tabanına yönlendirme
- Veri tabanı üzerindeki sistem hesabının parolasını keşfetme (Brute-Force saldırısı ile)

Açıklığın izah edilebilmesi noktasında çeşitli teknolojilerde kod örneklerine yer verilmiştir:

C# - Güvensiz Kod Bloğu:

```
//
// TR: Kullanıcı Girdisi Kullanarak Veritabanına Bağlanma
// EN: Connect to Database using User Input
//
private DbConnection OpenConnection_Unsafe(HttpRequest request)
{
    SqlConnection conn = null;
    string dbServer = request.Form["Server"];
    string dbName = request.Form["Database"];

    string userUrl = String.Format(CONN_STRING_TEMPLATE, dbServer, dbName);

    try
    {
        conn = new SqlConnection(userUrl);
        conn.Open();
    }
    catch (Exception ex)
    {
        HandleExceptions(ex);
    }
    return conn;
}
```

C# - Güvenli Kod Bloğu

```
//
// EN: Select Pre-Configured Database Connection by User Input
// TR: Kullanıcı Girisi ile Öntanımlı Veritabanı Bağlantısı Seçme
//
private DbConnection OpenConnection_SafeSelection(HttpRequest request)
{
    SqlConnection conn = null;
    int appId = Int32.Parse(request.Form["AppId"]);

    string userUrl;
    switch(appId) {
        case APP_ID1:
            userUrl=ConfigurationManager.ConnectionStrings[CONN_STRING_APP1].C
onnectionString;
            break;
        case APP_ID2:
            userUrl=ConfigurationManager.ConnectionStrings[CONN_STRING_APP2].C
onnectionString;
            break;
        case APP_ID3:
            userUrl=ConfigurationManager.ConnectionStrings[CONN_STRING_APP3].C
onnectionString;
            break;
        default:
            userUrl=ConfigurationManager.ConnectionStrings[CONN_STRING_DEFAULT
].ConnectionString;
    }

    try
    {
        conn = new SqlConnection(userUrl);
        conn.Open();
    }
    catch (Exception ex)
    {
        HandleExceptions(ex);
    }
    return conn;
}
```

C# - Güvenli Kod Bloğu - Alternatif Yöntem

```
//
// EN: Build Connection String with Builder
// TR: Builder ile Bağlantı Cümleciği Oluşturma
//
private DbConnection OpenConnection_WithBuilder(HttpRequest request)
{
    SqlConnection conn = null;
    string dbName = request.Form["Database"];

    SqlConnectionStringBuilder builder = new SqlConnectionStringBuilder();
    builder.DataSource = DB_SERVER;
    builder.InitialCatalog = SanitizeForConnString(dbName);
    builder.IntegratedSecurity = true;

    try
    {
        conn = new SqlConnection(builder.ToString());
        conn.Open();
    }
    catch (Exception ex)
    {
        HandleExceptions(ex);
    }
    return conn;
}
```

C# güvensiz kod bloğunda dbServer ve dbName nesneleri kullanıcı taraftan gelen girdileri almaktadır ve bu iki nesne bağlantı cümleciğine eklenerek bağlantı cümleciği dinamik olarak oluşturulmaktadır. Ardından oluşan bağlantı cümleciği ile veri tabanı bağlantısı başlatılmaktadır. Bu kullanım güvensizdir, çünkü kullanıcı girdileri bağlantı cümleciğine denetlenmeden eklenmektedir. Bu ise bağlantı cümleciğinin kullanıcı tarafınca enjeksiyon yoluyla kurcalanarak servis dışı bırakma, bağlantı trafiğini kurcalama, uygulamayı sahte veri tabanına yönlendirme ve kaba kuvvet saldırılarına imkan verebilir.

C# güvenli kod bloğunda switch-case ile kullanıcıların seçimi doğrultusunda ön tanımlı bağlantı cümleciklerinden birinin kullanılması yapısı vardır. Bu kullanım yarı dinamik bağlantı cümleciği oluşturma olarak ele alınabilir. Ön tanımlı bağlantı cümlecikleri kullanılması dolayısıyla güvenli kabul edilir.

C# güvenli kod bloğu alternatif yöntemde ise yüzde yüz dinamik bir bağlantı cümleciği oluşturmak gerekiyorsa, yani kullanıcı girdilerinin kesinlikle dinamik olarak bağlantı cümleciklerine eklenmesi gerekiyorsa bu durumda platformun olanaklarından yararlanarak güvenli bağlantı cümleciği oluşturma sınıflarından yararlanılabilir. Bu örnekte

SqlConnectionStringBuilder sınıfı ile dinamik olarak bağlantı cümleciği oluşturulmaktadır. SQL kod ve veri (girdi) sınırları ayrı ayrı tanımlı olduğundan bu kullanımda enjeksiyon imkanı yoktur. Dolayısıyla bu örnek de güvenli kabul edilir.

Java - Güvensiz Kod Bloğu

```
//
// Kullanıcı Girdisi Kullanarak Veritabanına Bağlanma
// EN: Connect to Database using User Input
//
private DbConnection OpenConnection_Unsafe(HttpRequest request)
{
    SqlConnection conn = null;
    string dbServer = request.Form["Server"];
    string dbName = request.Form["Database"];

    string userUrl = String.Format(CONN_STRING_TEMPLATE, dbServer, dbName);

    try
    {
        conn = new SqlConnection(userUrl);
        conn.Open();
    }
    catch (Exception ex)
    {
        HandleExceptions(ex);
    }
    return conn;
}
```

Java - Güvenli Kod Bloğu

```java
//
// EN: Select Pre-Configured Database Connection by User Input
// TR: Kullanıcı Girdisi İle Öntanımlı Veritabanı Bağlantısı Seçme
//
private Connection openConnection_SafeSelection(HttpServletRequest request)
        throws ServletException, SQLException {
    Connection conn = null;
    int appId = Integer.parseInt(request.getParameter("AppId"));

    String userUrl;
    switch(appId) {
        case APP_ID1:
            userUrl = JDBC_URL_APP1;
            break;
        case APP_ID2:
            userUrl = JDBC_URL_APP2;
            break;
        case APP_ID3:
            userUrl = JDBC_URL_APP3;
            break;
        default:
            userUrl = JDBC_URL_DEFAULT;
    }

    try {
        conn = DriverManager.getConnection(userUrl, DB_USER, DB_PASSWORD);
    } catch (Exception ex) {
        handleExceptions(ex);
    }
    return conn;
}
```

Java güvensiz kod bloğunda dbServer ve dbName nesneleri kullanıcı taraftan gelen girdileri almaktadır ve bu iki nesne bağlantı cümleciğine eklenerek bağlantı cümleciği dinamik olarak oluşturulmaktadır. Ardından oluşan bağlantı cümleciği ile veri tabanı bağlantısı başlatılmaktadır. Bu kullanım güvensizdir, çünkü kullanıcı girdileri bağlantı cümleciğine yine denetlenmeden eklenmektedir. Bu ise bağlantı cümleciğinin kullanıcı tarafınca enjeksiyon yoluyla kurcalanarak servis dışı bırakma, bağlantı trafiğini kurcalama, uygulamayı sahte veri tabanına yönlendirme ve kaba kuvvet saldırılarına imkan verebilir.

Java güvenli kod bloğunda switch-case ile kullanıcıların seçimi doğrultusunda ön tanımlı bağlantı cümleciklerinden birinin kullanılması yapısı vardır. Bu kullanım yarı dinamik bağlantı cümleciği oluşturma olarak ele alınabilir. Ön tanımlı bağlantı cümlecikleri kullanılması dolayısıyla güvenli kabul edilir.

VisualBasic.NET - Güvensiz Kod Bloğu

```
'
' EN: Connect to Database using User Input
' TR: Kullanıcı Girdisi Kullanarak Veritabanına Bağlanma
'
Function OpenConnection_Unsafe(request As HttpRequest) As DbConnection
    Dim conn As SqlConnection = Nothing
    Dim dbServer As String = request.Form("Server")
    Dim dbName As String = request.Form("Database")

    Dim userConnString As String = String.Format(CONN_STRING_TEMPLATE,
dbServer, dbName)

    Try
        conn = New SqlConnection(userConnString)
        conn.Open()
    Catch ex As Exception
        HandleExceptions(ex)
    End Try

    Return conn
End Function
```

VisualBasic.NET - Güvenli Kod Bloğu

```vb
'
' EN: Select Pre-Configured Database Connection by User Input
' TR: Kullanıcı Girdisi ile Öntanımlı Veritabanı Bağlantısı Seçme
'
Function OpenConnection_SafeSelection(request As HttpRequest) As DbConnection
    Dim conn As SqlConnection = Nothing
    Dim appId as Integer = Int32.Parse(request.Form("AppId"))

    Dim userConnString As String
    Select Case appId
        Case APP_ID1
            userConnString=ConfigurationManager.ConnectionStrings(CONN_STRING_
APP1).ConnectionString
        Case APP_ID2
            userConnString=ConfigurationManager.ConnectionStrings(CONN_STRING_
APP2).ConnectionString
        Case APP_ID3
            userConnString=ConfigurationManager.ConnectionStrings(CONN_STRING_
APP3).ConnectionString
        Case Else
            userConnString=ConfigurationManager.ConnectionStrings(CONN_STRING_
DEFAULT).ConnectionString
    End Select

    Try
        conn = New SqlConnection(userConnString)
        conn.Open()
    Catch ex As Exception
        HandleExceptions(ex)
    End Try

    Return conn
End Function
```

VisualBasic.NET - Güvenli Kod Bloğu - Alternatif Yöntem

```
'
' EN: Build Connection String with Builder
' TR: Builder ile Bağlantı Cümleciği Oluşturma
'
Function OpenConnection_WithBuilder(request As HttpRequest) As DbConnection
    Dim conn As SqlConnection = Nothing
    Dim dbName As String = request.Form("Database")

    Dim builder As SqlConnectionStringBuilder = New
SqlConnectionStringBuilder()
    builder.DataSource = DB_SERVER
    builder.InitialCatalog = SanitizeForConnString(dbName)
    builder.IntegratedSecurity = True

    Try
        conn = New SqlConnection(builder.ToString())
        conn.Open()
    Catch ex As Exception
        HandleExceptions(ex)
    End Try

    Return conn
End Function
```

Vb.net güvensiz kod bloğunda dbServer ve dbName nesneleri kullanıcı taraftan gelen girdileri almaktadır ve bu iki nesne bağlantı cümleciğine eklenerek bağlantı cümleciği dinamik olarak oluşturulmaktadır. Ardından oluşan bağlantı cümleciği ile veri tabanı bağlantısı başlatılmaktadır. Bu kullanım güvensizdir, çünkü kullanıcı girdileri bağlantı cümleciğine yine denetlenmeden eklenmektedir. Bu ise bağlantı cümleciğinin kullanıcı tarafınca enjeksiyon yoluyla kurcalanarak servis dışı bırakma, bağlantı trafiğini kurcalama, uygulamayı sahte veri tabanına yönlendirme ve kaba kuvvet saldırılarına imkan verebilir.

Vb.net güvenli kod bloğunda switch-case ile kullanıcıların seçimi doğrultusunda ön tanımlı bağlantı cümleciklerinden birinin kullanılması yapısı vardır. Bu kullanım yarı dinamik bağlantı cümleciği oluşturma olarak ele alınabilir. Ön tanımlı bağlantı cümlecikleri kullanılması dolayısıyla güvenli kabul edilir.

Vb.net güvenli kod bloğu alternatif yöntemde ise yüzde yüz dinamik bir bağlantı cümleciği oluşturmak gerekiyorsa, yani kullanıcı girdilerinin kesinlikle dinamik olarak bağlantı cümleciklerine eklenmesi gerekiyorsa bu durumda platformun olanaklarından yararlanarak güvenli bağlantı cümleciği oluşturma sınıflarından yararlanılabilir. Bu örnekte SqlConnectionStringBuilder sınıfı ile dinamik olarak bağlantı cümleciği oluşturulmaktadır.

SQL kod ve veri (girdi) sınırları ayrı ayrı tanımlı olduğundan bu kullanımda enjeksiyon imkanı yoktur. Dolayısıyla bu örnek de güvenli kabul edilir.

Kurum web uygulamasında “Bağlantı Cümleciği Enjeksiyonu (CWE-99)” açıklığı tespit edilmiştir:

:::::BULGU:::::

Açıklığın Önlemi:

Bu açıklığı gidermek için tavsiye edilen önlemler şu şekildedir:

- Tüm girdiler kaynağı neresi olursa olsun doğrulamadan geçirilmelidir. Bu doğrulama belirli yapıdaki girdileri reddetme işlemi yerine sadece belirli yapıya uyanların kabul edildiği şeklinde olmalıdır. Yani whitelist (beyaz liste) kullanılmalıdır. Siyah liste (blacklist) önlemi kullanılmamalıdır. Bir girdiyi doğrulamada şunlar kontrol edilebilir:
    - Veri türü (int, string, float, byte, … v.b. tipte mi geliyor kontrolü)
    - Boyutu (x KB, y MB,…v.b. boyutta mı geliyor kontrolü)
    - Aralığı (Veri ölçülebilir bir sayısal değerse aralık sınırlarını aşıyor mu kontrolü)
    - Formatı (çözümlenebilir şifreli, tek yönlü şifreli, açık metin, … v.b. biçimde mi geliyor kontrolü)
    - Beklenen Değerlerden Biri Olup Olmadığı (Whitelist’te tanımlı desenlerden biri mi geliyor kontrolü)
- Kullanıcılara veri tabanı bağlantı cümleciklerinin kontrolü için izin verilmemelidir. Güvensiz bir veri (özellikle kullanıcı girdileri) temel alınarak dinamik bağlantı cümlecikleri oluşturmaktan sakınılmalıdır.
- Tüm bağlantı cümlecikleri uygun bir konfigürasyon mekanizmasında depolanmalıdır. Eğer uygulamanın çalışması anında (runtime’da) dinamik olarak bir bağlantı cümleciği oluşturulması gerekiyorsa güvensiz veriler doğrudan bağlantı cümleciklerine dahil edilmemelidir. Bunun yerine kullanıcılara öntanımlı bağlantı cümleciklerinden birini seçme izni tanımlanabilir.
- Eğer uygulamanın çalışması anında kesinlikle dinamik olarak bir bağlantı cümleciği oluşturulması gerekiyorsa kullanılan platformun sağladığı sınıflar kullanılabilir. Örn; DbConnectionStringBuilder ve somut uygulamasını yapan sınıflar gibi.

Referanslar:

1. http://cwe.mitre.org/data/definitions/99.html
2.