1.1.1 Uygunsuz Kaynak Kapatma veya Serbest Bırakma (Improper Resource Shutdown or Release) (CWE-404)

Açıklık Önem Derecesi: Düşük

Açıklığın Etkisi: Servis dışı kalma

Açıklığın Barındıran Dosyalar/Satırlar:

Proje Dosyası/Dosya Adı	Satır Numarası

Açıklığın Açıklaması:

Uygulamalar kaynak kodlarındaki kodlamalara göre bellekten kaynaklar ayırırlar. Örneğin kaynak kodda bir eposta bağlantısı açılıyordur ve bunun için bellekten bir bağlantı açma kaynağı ayrılır. Örneğin bir veritabanı bağlantısı açılıyordur ve bunun için bellekten bir bağlantı açma kaynağı ayrılır. Bu kaynaklar nesnelere atanırlar ve bu nesneler işleri bittiklerinde kapatılmalıdırlar. Yani verilen örneklere göre eposta bağlantısı veya veritabanı bağlantısının kapatılarak kaynakların bellekte serbest bırakılması gerekmektedir. Eğer kaynaklar bellekte kapatılmazlarsa ve uygulama akışına o şekilde devam ederse kaynaklar bellekte gereksizce yer işgal etmiş olurlar. Bu duruma kaynak sızıntısı adı verilir. Kaynak sızıntısı olduğunda ve bu bir tekrara bindiğinde uygulama bellekteki tüm kaynakları tüketebilir ve hizmet dışı kalabilir. Bellekte kaynak tüketimi ile hizmet kesintileriyle karşılaşmamak için kaynak sızıntısı açıklığı önlenmelidir. Bunun için açılan her bir kaynak işi biteceği zaman kapatılmalıdır.

Açıklığı anlamak adına ilgili açıklığa sahip bir kod bloğu örnek olarak verilmiştir:

Java:

```java
// Güvensiz Kod

// ENG: Unreleased Database Connection
// TR : Serbest Bırakılmamış Veritabanı Bağlantısı

private MyObject getDataFromDb(int id)  {
    MyObject data = null;
    Connection con = null;
    try {
        Connection con = DriverManager.getConnection(CONN_STRING);
            data = queryDb(con, id);
    }
    catch ( SQLException e ) {
        handleError(e);
    }
}
```

Bu örnekte try bloğu içerisinde getConnection() ile bir veritabanı bağlantısı başlatılmıştır ve metot içerisinde veritabanı bağlantısı sonlandırılmamıştır. Bu güvensiz kodun güvenli hale dönüştürülmüş haline ise iki ayrı örnek verilebilir. Bunlardan ilki şu şekildedir:

Java:

```java
// Güvenli Kod

// ENG: Explicit Release of Database Connection
// TR : Açık Bir Şekilde Veritabanı Bağlantısının Serbest Bırakılması

private MyObject getDataFromDb(int id)  {

    MyObject data = null;
    Connection con = null;

    try {
        Connection con = DriverManager.getConnection(CONN_STRING);
            data = queryDb(con, id);
    }
    catch ( SQLException e ) {
        handleError(e);
    }
    finally {
            if ((con != null) && (!con.isClosed())) {
                con.close();
        }
    }
}
```

Bu güvenli kodlama örneğinde try'dan sonra mutlaka uğranılacak durak olan finally durağına gelindiğinde daha önce başlatılan veritabanı bağlantısının kapatıldığını görmekteyiz. Bu güvenli kodlamada veritabanı bağlantısı açık bir şekilde (explicitly) kapatılmaktadır. Bunu üstü kapalı (implicitly) olarak da yapabiliriz.

Java:

```
// Güvenli Kod

// ENG: Automatic Implicit Release Using Try-With-Resources
// TR : Try-With-Resources Yapısı Kullanarak Kaynağı Otomatik
//      Üstü Kapalı Olarak Serbest Bırakma

private MyObject getDataFromDb(int id)  {

    MyObject data = null;
    Connection con = null;

    try (Connection con = DriverManager.getConnection(CONN_STRING)) {
        data = queryDb(con, id);
    }
    catch ( SQLException e ) {
        handleError(e);
    }
}
```

Bu örnekte try-with-resources yapısı kullanılmıştır. Try-with-resources yapısı kaynak oluşturmaya yarar ve oluşturulan kaynağı program try bloğunu tamamladığında otomatik olarak serbest bırakır. Bu güvenli kodlama örneğinde kaynak otomatik olarak bellekte serbest bırakılmaktadır. Kaynakların oluşturulup sonra serbest bırakılmadığı durumda gerçekleşen güvensiz kodlamaya "Uygunsuz Kaynak Kapatma veya Serbest Bırakma" açıklığı adı verilir.

Kurum uygulamasında kaynak sızıntısı açıklıkları tespit edilmiştir. Şekil XXX. ABCDEF'de bu durum gösterilmiştir:

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

Açıklığın Önlemi:

Nesnelere atanan açılan kaynaklar nesneler üzerinden kapatılmalıdır. İlave eklenecek kodlama satırları ile kaynaklar kapatılarak bellekte gereksiz kaynak tüketiminin önüne geçilmelidir ve böylece uygulamanın servis dışı kalması riski ortadan kaldırılmalıdır.

Referanslar:

1. https://cwe.mitre.org/data/definitions/404.html
2. https://stackoverflow.com/questions/31719018/resource-leak-resource-out-of-scope
3. https://stackoverflow.com/questions/12519335/resource-leak-in-is-never-closed
4. https://www.youtube.com/watch?v=AWVBx1XL37c&ab_channel=Coverity%2CInc.
5. https://docs.oracle.com/javase/tutorial/essential/exceptions/tryResourceClose.html