1.1.1 Aşırı Geniş Catch Bloğu (Overly Broad Catch) (CWE-396)

Açıklık Önem Derecesi: Düşük

Açıklığın Etkisi: Güvenli yazılım altyapı eksikliği, Hata yönetimlerinden doğru geri besleme ve verim alamama

Açıklığın Barındıran Dosyalar/Satırlar:

Proje Dosyası/Dosya Adı	Satır Numarası

Açıklığın Açıklaması:

Programlama dillerinde çeşitli istisnalar meydana geldiğinde bu istisnaları yakalayan ve tanımlanacak işlemlerin uygulanmasını sağlayan catch adlı bloklar vardır. Bu catch bloklarını aşırı geniş istisnalar yakalayacak şekilde kullanmak gelecekte karmaşık hata yönetimlerine (error handling'e) neden olur ve bu durum güvenlik zafiyetleri doğurması muhtemel bir yola sevk edebilir.

Uygulamalarda catch blokları programlama dilindeki en geniş istisna sınıflarını yakalayacak şekilde kullanıldığında "Aşırı Geniş Catch Bloğu" (CWE-396) şeklinde bulgu olarak işaretlenirler. Bu bulguyu göstermek adına aşağıda güvensiz bir catch kullanımı (diğer bir ifadeyle güvensiz bir hata yönetimi) örneği verilmiştir:

Java:

```
// KÖTÜ KOD
...
    try {
        doExchange();
    }
    catch (Exception e) {
        logger.error("doExchange failed", e);
    }
...
```

Bu uygulamada en başta uygulama geliştirilirken doExchange()'in fırlatacağı istisna türleri bellidir ve bu nedenle catch bloğunda geniş istisna yakalaması yapılsa da bu belli istisnalara göre işlemler catch içerisinde kodlanır. Uygulamayı geliştirmeye devam ederken daha ileri bir zamanda doExchange() metodunda yapılacak bir güncelleme sonucu doExchange() metodu yeni bir istisna türü fırlatabilir hale gelebilir. Bu durumda buradaki catch bloğu bu yeni tür

istisnayı yine yakalayacaktır, ama bu yanıltıcı olacaktır. Çünkü mevcut catch bloğu yeni istisna türüne özgü davranışı ve muameleyi sunmayacaktır. catch bloğu içerisindeki kod satırları daha önce belirlenen satırlardan oluşmaktadır. Bu da gelecekte yaşanacak siber saldırılara karşı doğru hata ayıklamanın yapılmasını, sorunun kaynağının çözümlenmesini güçleştirecektir. Catch bloğunun güvensiz kullanımına karşılık güvenli kullanımına şu örnek verilebilir:

Java:

```
// İYİ KOD
...
    try {
        doExchange();
    }
    catch (IOException e) {
        logger.error("doExchange failed", e);
    }
    catch (InvocationTargetException e) {
        logger.error("doExchange failed", e);
    }
    catch (SQLException e) {
        logger.error("doExchange failed", e);
    }
...
```

Birden fazla catch bloğu kullanmak çirkin görünebilir ve tekrar tekrar aynı iş yapılıyormuş gibi görünebilir. Fakat Exception gibi yüksek seviyeli sınıf yakalayan yoğun catch blokları kullanmak özel muameleyi hak eden istisnaların atlanılmasına sebebiyet verebilir. Uygulama büyüdükçe aşırı geniş catch bloğu kullanılması ile yeni tür istisnalar yakalandığında bu yeni istisnalar mevcut catch bloğu tanımına göre ayrı bir muamele göremeyeceğinden gelecekte güvenlik noktasında doğru geri besleme alınamamasına sebebiyet verebilir.

Sonuç olarak iyi kodda gösterildiği gibi ayrı ayrı catch tanımı girilmelidir. Gelecekte try bloğunda farklı bir istisna türü fırladığında bu durumda yeni catch tanımı girerek uygulamada güvenlik noktasında daha doğru bir geri besleme alınabilir. Özel muamele gösterilmesi gereken catch blokları ise pratik olarak bu şekilde daha doğru uygulanabilir hale gelebilir.

Kurum uygulamasında güvensiz catch bloğu kullanıldığı tespit edilmiştir. Bu durum Şekil XXX. ABCDEF'de gösterilmiştir.

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

Açıklığın Önlemi:

Aşırı geniş istisna ile catch kullanmak yerine spesifik istisnalar ile birden fazla catch kullanmak tercih edilmelidir.

Referanslar:

1. https://cwe.mitre.org/data/definitions/396.html
2. https://vulncat.fortify.com/en/detail?id=desc.structural.dotnet.poor_error_handling_overly_broad_catch_block
3. https://www.w3schools.com/cpp/cpp_exceptions.asp
4. https://intellij-support.jetbrains.com/hc/en-us/community/posts/206928985-Overly-broad-catch-block-a-real-story
5.