1.1.1 Aşırı Geniş Throw Tanımlaması (Overly Broad Throws) (CWE-397)

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 metot tanımlarında (fonksiyon tanımlarında) çeşitli istisnalar meydana geldiğinde bu istisnaları bir üst bloğa (metodun çağırıldığı yere) fırlatılmasını sağlayacak tanımlar mevcuttur. Bu tanımlara throw tanımı adı verilir. Throw tanımları metot ismi ve parametreleri sonunda yer alır. Bu tanımların aşırı geniş istisnalar ile belirlenmesi gelecekte karmaşık hata yönetimlerine (error handling'e) neden olur ve bu durum güvenlik zafiyetleri doğurması muhtemel bir yola sevk edebilir.

Uygulamalarda metotların parametre sonlarında yer alan fırlayan istisnalar listesi throw tanımları en geniş istisna yakalayacak şekilde belirlendiğinde "Aşırı Geniş Throw Tanımı" (CWE-397) şeklinde bulgu olarak işaretlenirler. Bu bulguyu göstermek adına aşağıda güvensiz bir throw tanımı örneği verilmiştir:

Java:

```
// KÖTÜ KOD
...
      public void doExchange() throws Exception {
        ...
      }
...
```

Bu örnekte uygulama geliştirilirken doExchange()'in fırlatabileceği istisna türü bellidir. Bir üst blokta (metodun çağırıldığı yerde) fırlayan istisna catch ile yakalanacaktır ve ona göre bir hata yönetimi işlemi uygulanacaktır. 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 metottaki geniş throw tanımı bu yeni istisnayı da kapsayacağından program genel akışında bir sorun teşkil etmeyecektir, ama bu

yanıltıcı olacaktır. Çünkü yeni istisna türüne özgü davranışı ve muameleyi üst bloktaki catch sunamayacaktır. 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. Throw tanımının güvensiz kullanımına karşılık güvenli kullanımına şu örnek verilebilir:

Java:

```
// İYİ KOD
...
      public void doExchange() throws IOException,
                                InvocationTargetException,
                                SQLException {
            ...
      }
...
```

Birden fazla throw tanımı girmek çirkin görünebilir ve tekrar tekrar aynı iş yapılıyormuş gibi görünebilir. Fakat Exception gibi yüksek seviyeli throw tanımı girmek özel muameleyi hak eden istisnaların üst bloktaki (metotların çağırıldığı yerdeki) catch bloklarınca atlanılmasına sebebiyet verebilir. Uygulama büyüdükçe aşırı geniş throw tanımı kullanılması ile yeni tür istisnalar fırladığında bu yeni fırlayan istisnalar mevcut üst bloktaki 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 ve dar kapsamlı throw tanımı girilmelidir. Gelecekte metotta farklı bir istisna türü fırladığında bu durumda yeni throw tanımı girmek buna tekabül eden bir catch bloğunu da girmek gerektiğinin farkındalığını verecektir ve uygulamada güvenlik noktasında daha doğru bir geri besleme alınabilecektir.

Kurum uygulamasında güvensiz throw tanımı kullanıldığı tespit edilmiştir. Bu durum Şekil XXX. ABCDEF'de gösterilmiştir.

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

Açıklığın Önlemi:

Aşırı geniş throw tanımı kullanmak yerine spesifik ve birden fazla throw tanımı kullanmak tercih edilmelidir.

Referanslar:

1. https://cwe.mitre.org/data/definitions/397.html