1.1.1 Ölü Kod: İfadenin Daima True Olması (Dead Code: Expression is Always True) (CWE-571)

Açıklık Önem Derecesi: Düşük

Açıklığın Etkisi: Uygulama güvenliğinin sürdürebilirliğini azaltma

Açıklığın Barındıran Dosyalar/Satırlar:

Proje Dosyası/Dosya Adı	Satır Numarası

Açıklığın Açıklaması:

Uygulamalarda çalışmayan, işleve sahip olmayan kodlamalar yer alabilmektedir. Bu kodlar uygulamanın kod kalitesini düşürücü etkiye sahiptir. Uygulama zamanla büyüdükçe sonradan uygulama kaynak kodlarına bakıldığında her bir kodun işlevini anlama zorlaşır ve çalışmayan kodların varlığı da bu zorluğu arttırır. Bu durum uygulama güvenliğini sürdürme noktasında bir negatiflik olarak yansır. Uygulamalardaki çalışmayan kodlar uygulama güvenliğini sürdürme noktasında dezavantaj sunduklarından dolayı dolaylı yoldan güvenliğe dokunan bir açıklık olarak ele alınırlar.

Uygulama kaynak kodlarında ifadeler daima true olabilmektedir. Bu durum çalışmayan kod blokları meydana getirmektedir. Bunu örneklemek maksatlı Java dilinde iki ayrı örnek verilmiştir:

Java Güvensiz Kod Bloğu:

```
public void setUpCalls() {
    boolean firstCall = true;
    boolean secondCall = true;

    if (fCall < 0) {
        cancelFCall();
        firstCall = false;
    }

    if (sCall < 0) {
        cancelSCall();
        firstCall = false;
    }

    if (firstCall || secondCall) {
        setUpForCall();
    }
}
```

Bu "ifadelerin daima true olması" açıklıklı örnekte if (firstCall || secondCall) bloğu daima / her koşulda true dönecektir ve içerisindeki setUpDualCall() metodu daima / her koşulda çalışacaktır. Çünkü en başta firstCall ve secondCall nesnesine true değeri atanmaktadır. Ardından blok boyunca firstCall false ataması iki kez yanlışlıkla yapılmaktadır. Aslında ikinci firstCall false ataması secondCall false ataması şeklinde olacaktı. Bu hata dolayısıyla blok boyunca secondCall true olarak kalacaktır ve değişime uğramayacaktır. Bloğun sonunda yer alan if (firstCall || secondCall) ifadesi ise firstCall true da olsa false da olsa secondCall her koşulda true olacağından daima true olacaktır ve if (firstCall || secondCall) bloğuna daima girilecektir. Bu durum çalışmayan, fuzuli bir if (firstCall || secondCall) ifadesi meydana getirecektir.

Java Güvensiz Kod Bloğu:

```
public void setUpCalls() {
    boolean firstCall = false;
    boolean secondCall = false;

    if (fCall > 0) {
        setUpFCall();
        firstCall = true;
    }

    if (sCall > 0) {
        setUpSCall();
        secondCall = true;
    }

    if (firstCall = true && secondCall == true) {
        setUpDualCall();
    }
}
```

Bu "ifadelerin daima true olması" açıklıklı diğer örnekte ise if (firstCall = true && secondCall == true) bloğunda koşulun ilk kısmı daima true dönecektir. Çünkü if ifadesindeki firstCall nesnesine yanlışlıkla == yerine = konulmuştur. Burada if (secondCall == true) koşulu ile birebir aynı çalışma şekli gerçekleşmektedir. Dolayısıyla if bloğunun ilk kısmı işlevsiz, fuzuli bir koşul meydana getirecektir.

Kurum uygulamada "ifadenin daima true olması (CWE-571)" açıklığı tespit edilmiştir:

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

Açıklığın Önlemi:

Uygulama kaynak kodlarında işlevsiz / çalışmayan kodlar giderilmeli. Örneğin bir karara bağlanıp çalışır hale getirilmeli veya kaldırılmalı.

Referanslar:

1. https://vulncat.fortify.com/en/detail?id=desc.structural.java.dead_code_expression_is_always_true#Java%2fJSP
2. https://cwe.mitre.org/data/definitions/571.html