1.1.1 Off By One Hatası (Off By One Error) (CWE-193)

Açıklık Önem Derecesi: [C/C++'da Yüksek] [Java'da Düşük]

Açıklığın Etkisi: Servis dışı kalma, uzaktan komut çalıştırma

Açıklığın Barındıran Dosyalar/Satırlar:

Proje Dosyası/Dosya Adı	Satır Numarası

Açıklığın Açıklaması:

Değişkenleri bellekte oluştururken boyutunu (size) veya uzunluğunu (length) belirlemede bir kademe sapma sonucu hesaplama hatası çoğu zaman meydana gelebilir. Örneğin döngülerde 2 boyutlu bir dizi alanı tahsis edilirken dizi elemanları 0 ve 1 olarak sayılır. Fakat dizinin üzerinde bir for döngü iterasyonu yanlışlıkla başlangıç koşulu i=0 ve devam koşulu i<=2 şeklinde ayarlanırsa 2 dizi elemanı yerine 3 dizi elemanına erişilecektir ve başlangıçta yeri ayarlanmış dizinin sınırları dışında potansiyel bellek bozulmasıyla sonuçlanacak yeri tahsis edilmemiş olan eleman [2]'ye yazma ve okuma yapılacaktır.

Bir diğer örnek; null byte ile sonlanan bir string (dize), sonlandırıcı null byte'ı olmadan kopyalandığında meydana gelir. Null byte olmadığında string (dizi) sonlandırılmamış olur ki bu durum belirli fonksiyonların null sonlandırıcıyı beklediklerinden dolayı belleği aşırı okumalarıyla sonuçlanır.

Problemli kullanımlara C / C++ dilinde şöyle bir örnek verilebilir:

C / C++ - Güvensiz Kod Bloğu:

```
// Off-By-One Vulnerability in For Loop

int *ptr;
ptr = (int*)malloc(5 * sizeof(int));
for (int i = 0; i <= 5; i++)
{
    ptr[i] = i * 2 + 1;     // ptr[5] alan tahsisi yapılacak,
                            // fakat sınırların dışına çıkılmış olur.
}
```

C / C++ - Güvenli Kod Bloğu:

```
// Proper Iteration in For Loop

int *ptr;
ptr = (int*)malloc(5 * sizeof(int));
for (int i = 0; i < 5; i++)
{
    ptr[i] = i * 2 + 1; // ptr[0-4] ile düzenli şekilde alan tahsisi yapılır.
}
```

C / C++ güvensiz kod bloğunda ptr dizisine 5 int boyutunda RAM'de alan tahsisi yapılıyor. Fakat for döngüsü başlangıç ve devam koşulu düzgün yapılandırılmadığı için ptr[4] son elemanken ptr[5]'e veri yazdırılmaya çalışılıyor. ptr dizisine kullanıcı girdisi yazdırıldığı durumda Off By One Error açıklığı ortaya çıkacaktır. C / C++ güvenli kod bloğunda ise ptr dizisine yine 5 int boyutunda RAM'de alan tahsisi yapılıyor. For döngüsü başlangıç ve devam koşulu düzgün yapılandırıldığı için bu sefer ptr dizisinde sınırların dışına veri yazdırma kusuru yer almayacaktır. ptr dizisine kullanıcı girdisi yazdırıldığı durumda Off By One Error açıklığı riski söz konusu olmayacaktır. For döngüsü iterasyon başlangıç ve devam koşulları düzgün yapılandırıldığından bu kullanım kodu güvenli kılmaktadır.

Java dilinde bu problemli kullanıma şöyle bir örnek verilebilir:

Java - Güvensiz Kod Bloğu:

```
// Eğer TERMINATION_MEMBER Elemanı Dizide Yoksa
// Dizide Off-by-One Hatası Durumu

for (int i = 0; i <= arr.length; i++) {       // arr[arr.length] indisi hiçbir
                                              // zaman tanımlanmadı ve bu
                                              // nedenle daima exception ile
                                              // sonuçlanacaktır.

    if (arr[i].equals(TERMINATION_MEMBER) {   // "arr" dizisinin bir elemanı
                                              // iterasyonu sonlandıracak
                                              // şekilde tanımlandığı
                                              // varsayılmakta.

        break;  // Bu komut istisnadan uzak kılacaktır,
                // fakat yalnızca arr dizisi beklenen
                // sonlandırıcı üye değeri içeriyorsa.

    }

    doSomething(arr[i]); // Eğer TERMINATION_MEMBER "arr" dizisinde bir eleman
                         // değilse, arr[arr.length] dereference edilmeye
                         // çalışıldığında bir istisna meydana gelecektir.

}
```

Java - Güvenli Kod Bloğu:

```
// "arr" Dizisinde TERMINATION_MEMBER Yoksa
// Bile Dizinin Uygun Şekilde Çalışması

for (int i = 0; i < arr.length; i++) {

    if (arr[i].equals(TERMINATION_MEMBER) {   // "arr" dizisinin bir elemanı
                                              // iterasyonu sonlandıracak
                                              // şekilde tanımlandığı
                                              // varsayılmakta.


        break;
    }

    doSomething(arr[i]);                      // Fonksiyon "arr" dizisinin her
                                              // bir i indisinde uygun bir şe-
                                              // kilde çalışacaktır.

}
```

Java güvensiz kod bloğunda arr dizisi for döngüsü içerisinde doSomething() metoduna argüman olarak girmektedir ve metot işlemini uygulamaktadır. For döngüsü başlangıç ve devam koşulu arr dizisini tanımlanmış indis aralığının dışına çıkardığından doSomething(arr[i]) metodu arr dizisinin tanımlanmamış indisindeki değerini (yani arr[arr.length]'i) almaya çalışacaktır ve bir exception (istisna) meydana gelecektir. For döngüsü içerisinde bu durumu önlemek için arr dizisinde sonlandırıcı olarak belirlenmiş bir karakter var mı if kıyaslaması yapılmaktadır ve varsa break ile for döngüsü kırılarak döngüden çıkılmaktadır. Bu if kıyaslamasının altında yatan neden doSomething() metodunun arr dizisinin tanımlanmış aralığı dışındaki değerine ulaşmaya çalışmasını önlemektir. Fakat arr dizisinde bu sonlandırıcı olarak belirlenmiş karakter yoksa veya unutulmuşsa exception (istisna) kaçınılmaz olacaktır. Java güvenli kod bloğunda ise arr dizisi yine bir for döngüsü içerisinde doSomething() metoduna argüman olarak girmektedir ve metot işlemini uygulamaktadır. For döngüsü başlangıç ve devam koşulu bu sefer arr dizisinin tanımlanmış aralığı boyunca uygun ilerlediğinden arr dizisi sonlandırıcı olarak belirlenmiş karaktere elemanlarından birinde sahip olmasa dahi doSomething(arr[i]) metodu hatasız şekilde tüm for döngüsü iterasyonları boyunca çalışabilecektir. For döngüsünde bu tanımlanan başlangıç ve devam koşulu kodu güvenli kılmaktadır.

Maksimum sayıyı bir fazlası kadar veya minimum sayıyı bir azı kadar aşma sonucu meydana gelen taşma durumuna "Off By One Error" (CWE-193) açıklığı adı verilir. Bu açıklık eğer bellekte veri okuma durumunda yer alıyorsa bellekte aşırı okumaya, eğer bellekte veri yazma durumunda yer alıyorsa aşırı yazmaya sebep olur. Bunun neticesinde uygulama çökertilerek (crash), sonlandırılarak (exit), yeniden başlamaya sevk edilerek (restart), CPU tüketimi arttırılarak veya RAM tüketimi arttırılarak servis dışı bırakılabilir. Aynı zamanda belleğe yazma yapan bir işlemde taşma meydana geldiğinde bellekteki veri modifiye edilebilir. Ayrıca bu taşmalar buffer overflow'u tetikleyebilir ve uzaktan keyfi komut çalıştırma saldırılarına imkan tanıyabilir.

Kurum uygulamada Off By One Error (CWE-193) açıklığı tespit edilmiştir.

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

Açıklığın Önlemi:
Bu açıklık hakkında alınabilecek önlem şu şekildedir:

- Verilen iterasyon sınırlarının doğruluğundan daima emin olun.

- Dizi iterasyonları kullanımlarında n boyutlu bir dizi için dizi 0 indisinden başlıyor mu ve n-1 indisinde bitiyor mu gözden geçirin.
- Karakter dizileri ve null byte sonlandırıcı karakter kullanımlarında null byte'ın kullanılır olduğunu, üzerine yazma işlemi yapılmadığını ve ele alınmamazlık (ignore) yapılmadığını gözden geçirin. Fonksiyonların off-by-one açıklıklı olmadığından emin olun. Örneğin karakter dizisinde tamponun sonuna otomatik olarak eklenen null byte'ların karakter dizisinin son elemanı yerine eklenmediğinden emin olun.
- Mümkün olduğu her yerde off-by-one hatası eğilimli olmayan ve belleği yöneten güvenli fonksiyonlar kullanın.

Referanslar:

1. http://cwe.mitre.org/data/definitions/193.html
2. https://vulncat.fortify.com/en/detail?id=desc.internal.cpp.buffer_overflow_off_by_one#C%2fC%2b%2b
3. https://vulncat.fortify.com/en/detail?id=desc.internal.cpp.out_of_bounds_read_off_by_one#C%2fC%2b%2b