1.1.1 Blok Sınırlandırma Eksikliği (Missing Block Delimitation) (CWE-483)

Açıklık Önem Derecesi: Düşük

Açıklığın Etkisi: Uygulama güvenliğinin sürdürebilirliğini azaltma

Açıklığın Barındıran Dosyalar/Satırlar:

Proje Dosyası/Dosya Adı	Satır Numarası

Açıklığın Açıklaması:

Programlama dillerinin bazısında blokları sınırlandırmak adına parantezler (veya diğer sınırlandırıcılar) isteğe bağlı şekilde kullanılır. Eğer bir blokta sınırlandırıcıya yer verilmiyorsa blokta olduğu düşünülen bir ifade blok içerisinde ele alınmaması durumu yaşanabilir ve bu durum mantıksal bir hataya sebebiyet verebilir. Bazı durumlarda bu mantıksal hata güvenlik etkilerine sahip olabilir.

if ifadelerinde, for döngülerinde v.b. blok yapılarda süslü parantezlere yer verilmediği durumlar yaşanabilmektedir. Bu durum programcılar tarafından blokta birden fazla kod satırı olmadığında tercih edilmektedir. Fakat bu tercih tavsiye edilen bir kullanım şekli değildir. Bu alışkanlık daha sonradan koşullu çalışması gereken satırların koşulsuz çalışmasına sebebiyet verebilir ve güvenlik üzerine kurgulanmış kod satırları kullanıldığında bu satırların beklenmedik şekilde çalışmasına yol açabilir. Süslü parantezler kullanmadan blok yapılarını kullanmak düşük bir güvenlik riski sunmasına karşın uygulama dünyasındaki diğer güvenlik açıklıkları ile beraber değerlendirilen bir açıklıktır.

Örneğin Java dilinde bu açıklığa bir örnek verilmiştir:

Java Güvensiz Kod Bloğu:

```java
public class Example1{
    public static void main(String[] args){
        boolean condition = true;
        int x = 10, y = 5;

        do{
            if(x == y){
                for(int i = 0; i<x; i++)
                    x--;
                    y++;
            }
        } while(x>0);
    }
}
```

Bu örnekte süslü parantez kullanım prensibi benimsenmediği için döngüde çalışması gereken bir satırın (y++'nın) döngü dışında çalışması durumu gerçeklemiştir. For döngüsünde çalışmayan satır gelecekte bir güvenlik fonksiyonunun başına gelirse bir güvenlik riski ortaya çıkacaktır.

Java Güvenli Kod Bloğu:

```java
public class Example1{
    public static void main(String[] args){
        boolean condition = true;
        int x = 10, y = 5;

        do{
            if(x == y){
                for(int i = 0; i<x; i++) {
                    x--;
                    y++;
                }
            }
        } while(x>0);
    }
}
```

Bu örnekte ise süslü parantez prensibini takip eden bir geliştiricinin prensibi daima uygulaması dolayısıyla daha önce yaşanan riske yakalanmadığı ve y++ satırının blok içerisinde çalıştırıldığı gösterilmiştir.

Örneğin C++ dilinde bu açıklığa bazı örnekler verilmiştir:

C++ Güvensiz Kod Blokları:

```
# ÖRNEK A
if (condition)
    firstActionInBlock();
    secondAction();                 // Uygunsuz; koşul dışında çalışmaktadır.
    thirdAction();

# ÖRNEK B
if (condition) firstActionInBlock(); secondAction();

// Uygunsuz; secondAction koşul dışında çalışmaktadır.

# ÖRNEK C
if (condition) firstActionInBlock(); // Uygunsuz.
    secondAction();                 // Koşul dışında çalışmaktadır.

# ÖRNEK D
if (condition); secondAction();     // Uygunsuz; secondAction
                                    // koşul dışında çalışmaktadır.

# ÖRNEK E
String str = null;
for (int i = 0; i < array.length; i++)
    str = array[i];
    doTheThing(str);                // Uygunsuz; doTheThing() yalnızca son
                                    // dizi elemanında çalışmaktadır.
```

Örnek A'dan E'ye kadar blok sınırlandırıcı kullanılmaması dolayısıyla koşulsuz çalışan kod satırları örnekleri gösterilmiştir.

C++ Güvenli Kod Blokları:

```
# Güvensiz Örnek A'nın Prensipli Hali
if (condition) {
    firstActionInBlock();
    secondAction();
}
thirdAction();


# Güvensiz Örnek E'nin Prensipli Hali
String str = null;
for (int i = 0; i < array.length; i++) {
    str = array[i];
    doTheThing
}
```

Bu örneklerde ise süslü parantez prensibini takip eden bir geliştiricinin prensibi daima uygulaması dolayısıyla daha önce yaşanan koşulsuz kod satırlarının çalışması riskine yakalanmadığı gösterilmiştir.

Kurum uygulamada "Eksik Blok Sınırlandırma (CWE-483)" açıklığı tespit edilmiştir:

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

Açıklığın Önlemi:

Bloklarda **"daima"** süslü parantezler kullanılmalıdır. Gerekli olmasa dahi parantezlerin daima konulması prensip olarak benimsenmelidir. Böylece gelecekteki olası hatalı işlemlerin önüne prensibe riayet edileceğinden geçilmiş olacaktır.

Referanslar:

1. https://rules.sonarsource.com/cpp/tag/cwe/RSPEC-2681
2. https://cwe.mitre.org/data/definitions/483.html
3.