1.1.1 Integer Taşması (Integer Overflow) (CWE-190)

Açıklık Önem Derecesi: Düşük

Açıklığın Etkisi: işlevlerin uygun çalışmaması, servis dışı kalma, komut çalıştırma

Açıklığın Barındıran Dosyalar/Satırlar:

Proje Dosyası/Dosya Adı	Satır Numarası

Açıklığın Açıklaması:

Uygulamalarda kaynak kodlarda yer alabilen aritmetik işlemlerde yaşanan taşmalar veri bozulmalarına (örn; değer hesaplamalarında maksimum değerlerin minimum değer olmasına), sistemlerin çökmesine, sonsuz döngüler oluşmasına, mantıksal hataların meydana gelmesine (örn; güvenlik mekanizmalarının atlatılmasına), verinin çöp olmasına veya kayba uğramasına neden olabilir. Aritmetik işlemlerde taşmalar tampon taşmasına (buffer overflow'a) dönüşürlerse taşan sayısal değerler ile bellekte işlemler gerçekleştirilebilir ve keyfi komut çalıştırma sonucuna kadar varılabilir.

Tüm sayısal veri tipleri bir bitsel temsile sahiptir. Eğer bir aritmetik işlemde bir değer kendisinin bitsel temsilindeki bit miktarını aşarsa, yani maksimum değerin üzerine çıkarsa o zaman bitsel temsilin en başına yeni bir hane olarak ilave gelen (yani taşan) sayı kırpılacaktır ve değişken geri kalan bitler ile minimum değerden başlayacaktır. Buna "aritmetik taşma" (arithmetic overflow) veya diğer yaygın bilinen yanlış adıyla "integer taşması" (integer overflow) adı verilir. "Integer Taşması" yanlış bir isimlendirmedir, çünkü integer veri tipinin dışındaki veri tiplerinde de bu taşma durumu meydana gelebilmektedir. Eğer bir aritmetik işlemde bir değer kendisinin bitsel temsilindeki bit miktarını aşarsa, yani minimum değerin daha aşağısına inerse o zaman bitsel temsilin en başına yeni bir hane olarak ilave gelen (yani taşan) sayı kırpılacaktır ve değişken geri kalan bitler ile maksimum değerden başlayacaktır. Buna "integer düşmesi" (integer underflow) adı verilir.

"Integer Overflow" ve "Integer Underflow" işlemine şu örnekler verilebilir:

- Örneğin 8 bitlik işaretli (signed) bir integer değişkeni -128'den 127'ye bir değer aralığına sahiptir. Eğer bu integer değişkeninin tanımlı maksimum değeri 127'ye 1 eklenirse maksimum değer aşılmış olur. Değer ise bu taşma sonucu işaret biti 1 olduğundan ve geri kalan bitler 0 olacağından -128 olur. Yani değişken artık minimum değerden başlar hale gelir. Bu durum bir integer overflow (integer taşması)'dır.

- Örneğin 8 bitlik işaretli (signed) bir integer değişkeni -128'den 127'ye bir değer aralığına sahiptir. Eğer bu integer değişkeninin tanımlı minimum değeri -128'den 1 çıkarılırsa minimum değer aşılmış olur. Değer ise bu taşma sonucu işaret biti 0 olduğundan ve geri kalan bitler 1 olduğundan 127 olur. Yani maksimum değerden başlar hale gelir. Bu durum bir integer underflow (integer düşmesi)'dir.
- Örneğin işaretsiz (unsigned) 32 bitlik bir integer değişkeni 0'dan 4,294,967,295'ye bir değer aralığına sahiptir. Eğer bu integer değişkeninin tanımlı maksimum değeri 4,294,967,295'ye 1 eklenirse maksimum değer aşılmış olur. Değer ise bu taşma sonucu yeni bit hanesi değeri kırpıldığında geriye kalan bitler 0 olarak kaldığından 0 olur. Yani değişken minimum değerden başlar hale gelir. Bu durum integer overflow (integer taşması)'dır.
- Örneğin işaretli (signed) 32 bitlik bir integer değişkeni -2,147,483,648'den 2,147,483,647'ye bir değer aralığına sahiptir. Eğer bu integer değişkeninin tanımlı maksimum değeri 2,147,483,647'ye 1 eklenirse maksimum değer aşılmış olur. Değer ise bu taşma sonucu işaret biti 1 olduğundan ve geriye kalan bitler 0 olduğundan (eksi) -2,147,483,648 olur. Yani minimum değerden başlar hale gelir. Bu durum yine bir integer overflow (integer taşması)'dır.

Özetle büyük bir sayısal değer ile aritmetik işlem sonrası daha büyük bir sayısal değer oluşması gerekirken veri tipinin minimum değerine düşülürse integer overflow, küçük bir sayısal değer ile aritmetik işlem sonrası daha küçük bir sayısal değer oluşması gerekirken veri tipinin maksimuma değerine çıkılırsa integer underflow yaşanmış demektir.

Aritmetik taşma açıklığına şu kod örnekleri verilebilir:

Java - Güvensiz Kod Bloğu:

```
// ENG: Addition Vulnerable to "Overflow" and "Underflow"
// TUR: "Overflow" ve "Underflow" Zafiyetli Toplama

public static int vuln_addition1(int a, int b) {
    return a + b;
}
```

Bu örnekte istemci girdisi olan a ve b signed int değişken değerleri hiçbir sınır değer kontrolü olmadan toplama işlemine tabi tutulmaktadır. Bu toplama sonucu int (integer) veri tipi bitsel temsilini aşabilir. Taşan bitin kırpılması sonrası kalan bitler ile küçük bir değer toplam sonucu olarak dönebilir. Dolayısıyla bu kod bloğu güvensiz kabul edilmektedir.

Java - Güvensiz Kod Bloğu:

```java
// ENG: Incorrect Check - Vulnerable to Underflow, and
//      Incorrect Negative Value Handling
// TUR: Doğru Olmayan Kontrol - “Underflow” Zafiyetli
//      ve Doğru Olmayan Negatif Değer Değerlendirici

public static int vuln_addition2(int a, int b) {
    if(a+b < 0)
        throw new RuntimeException ("Integer overflow!");
    return a + b;
}
```

Bu örnekte istemci girdisi olan a ve b signed int değişken değerleri integer overflow için sınır değer kontrolüne sahiptir, fakat underflow için sınır değer kontrolüne sahip değildir. Bu toplama yine güvensiz kabul edilmektedir.

Java - Güvenli Kod Bloğu (1):

```java
// ENG: Addition That will Throw Exception in Case of
//      Overflow or Underflow
// TUR: Overflow veya Underflow Durumunda İstisna Fırlatan
//      Toplama

public static int safe_addition1(int a, int b) {

    if (a > 0 && b > 0 && a + b < 0)
        throw new RuntimeException ("Integer overflow!");

    if (a < 0 && b < 0 && a + b > 0)
        throw new RuntimeException ("Integer underflow!");

    return a+b;
}
```

Bu örnekte ise “Integer Overflow” veya “Integer Underflow” için kontroller girildiğinden istemci girdisi a ve b signed integer’ları kontrol sonrası toplanmaktadır. Dolayısıyla bu kod “Integer Overflow” ve “Integer Underflow”a karşı güvenli kabul edilmektedir.

Java - Güvenli Kod Bloğu (2):

```java
// ENG: Using a bigger data type to hold all the
//      possible result values
// TUR: Tüm Olası Sonuç Değerlerini Tutmak İçin
//      Daha Büyük Veri Tipi Kullanma

public static short safe_addition2(short a, short b) {

    int total = a+b;

    if(total > Short.MAX_VALUE)
        return Short.MAX_VALUE;

    if(total < Short.MIN_VALUE)
        return Short.MIN_VALUE;

    return (short)total;
}
```

Bu örnekte de a ve b istemci girdi değişkenleri için int veri tipinden bitsel temsili daha küçük olan short veri tipi kullanılmıştır. Böylece küçük girdiler daha büyük değişkende toplanarak veri kaybı önlenmeye çalışılmıştır. Aritmetik işlem sonrası toplam değeri eğer short veri tipinin maksimum ve minimum değerleri arasındaysa bu değer döndürülmektedir, fakat eğer toplam değeri short veri tipinin maksimum ve minimum değerleri arasında değilse bu durumda short veri tipinin sınır değerlerinden maksimum veya minimum değer toplam sonucu olarak döndürülmektedir. Bu kodda da “Integer Overflow” ve “Integer Underflow” kontrolü olması dolayısıyla güvenli bir yol kabul edilmektedir.

Kurum uygulamada "Integer Taşması (CWE-190)" açıklığı tespit edilmiştir:

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

Açıklığın Önlemi:

Aritmetik işlem sonucu integer overflow olmasın diye aritmetik işlemde kullanılan istemci girdileri değerleri tanımlı bir minimum & maksimum değer aralığında mı kontrol edilmelidir. Ancak ondan sonra bu değişkenler aritmetik işlemde kullanılmalıdır.

Referanslar:

1. https://www.geeksforgeeks.org/difference-between-byte-short-int-and-long-datatype-in-java/
2. https://cwe.mitre.org/data/definitions/190.html
3. https://vulncat.fortify.com/en/detail?id=desc.dataflow.cobol.integer_overflow#C%2FC

%2B%2B

4. https://www.acunetix.com/blog/web-security-zone/what-is-integer-overflow/
5. https://stackoverflow.com/questions/5739888/what-is-the-difference-between-signed-and-unsigned-int
6. https://www.welivesecurity.com/2022/02/21/integer-overflow-how-it-occur-can-be-prevented/
7. Chat GPT: Question: What is the difference between integer overflow and integer underflow?
8. https://www.gnu.org/software/autoconf/manual/autoconf-2.63/html_node/Integer-Overflow-Basics.html
9. https://www.welivesecurity.com/2022/02/21/integer-overflow-how-it-occur-can-be-prevented/