1.1.1 Nesne Kaçırma (Object Hijack) (CWE-491)

Açıklık Önem Derecesi: Düşük

Açıklığın Etkisi: Bilgi İfşası

Açıklığın Barındıran Dosyalar/Satırlar:

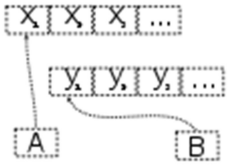
Proje Dosyası/Dosya Adı	Satır Numarası

Açıklığın Açıklaması:

Nesne yönelimli programlama dillerinde “nesne kopyalama” var olan bir nesnenin kopyasını oluşturmaya denir. Nesne kopyalama ile kastedilen sıfırdan bir nesne oluşturmak yerine var olan bir nesnenin tüm özelliklerini içeren aynı türden, fakat farklı referanslı bir nesne oluşturmaktır. Neredeyse tüm nesne yönelimli programlama dillerinde nesne kopyalama vardır. Nesnelerin çeşitli metotlarla - örn; kopyalama constructor’ı ile veya cloning ile - kopyaları oluşturulabilmektedir.

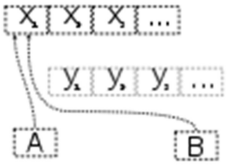
a) Sığ Kopyalama (Shallow Copy)

Sığ kopyalama bir nesnenin kopyasını oluşturma yöntemidir. Bu yönteme göre orijinal nesne kopya nesneye kopyalanırken orijinal nesnenin primitive field’ları kopya nesnenin field’larına “değer olarak” kopyalanır ve orijinal nesnenin referanslı field’ları ise kopya nesnenin field’larına “refarans olarak” kopyalanır.



A Orijinal Nesnesi,
B Boş Nesnesi

A Orijinal Nesnesi
B Nesnesine Sığ
Kopyalanır



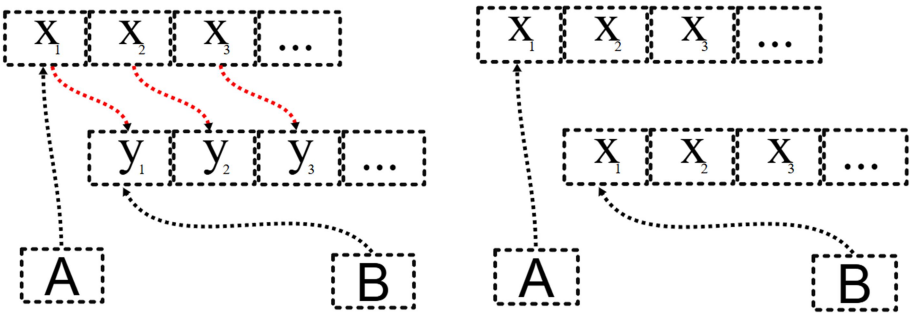
B Kopya Nesnesi
A Nesnesi
Field’larını Gösterir

Örneğin A orijinal nesnesi sığ kopyalama ile B nesnesine kopyalanırsa B kopya nesnesi referans olarak A orijinal nesnesini gösterecektir. Dolayısıyla B kopya nesnesinin field’ları A orijinal nesnesinin field’larını gösterecektir. Eğer bu nesnelerden biri güncellenirse değişiklik diğer nesnede de görünür olacaktır.

Sığ kopyalama field-by-field kopyalama veya diğer adıyla field kopyalama adıyla da anılmaktadır.

b) Derin Kopyalama (Deep Copy)

Nesne kopyalamada alternatif yöntem derin kopyalamadır. Bu yöntemde orijinal nesnenin primitive olsun veya referanslı olsun tüm field'ları "referans yoluyla" değil de "değer yoluyla" kopya nesne field'larına kopyalanır.



A Orijinal Nesnesi
B Nesnesine Kopyalanıyor.

Kopyalama Sonrası
B Kopya Nesnesi A Nesnesi
Field Değerlerini Gösterir.

Örneğin A orijinal nesnesi derin kopyalama ile B nesnesine kopyalanırsa A orijinal nesnesinin field'larının değerleri B kopya nesnesinin field'larına değer olarak kopyalanır. Böylece B kopya nesnesi field'larını gösterdiğinde A orijinal nesnesinin "kopya" değerlerini gösterir. Fakat kopyalama referans olarak değil de değer olarak olduğundan bu nesnelerden birinde var olabilecek bir güncellemeyi diğer nesne göremez. Bahsedilen bu iki yöntem de kaynak kodda bir nesnenin klonunu üretme üzerinedir.

Bir nesneden klon bir nesne / kopya bir nesne ait olduğu sınıfın constructor'ı kullanılmadan oluşturulabilmektedir. Hatta klonlama işleminin yapıldığı klonlama metodundaki doğrulamalara ve sınırlamalara uğramadan da oluşturulabilmektedir. Bu ise beklenmeyen davranışlarla - örn; kopyalama sırasında uygulanan kısıtlamaların ve doğrulamaların atlatılması ile - sonuçlanabilir. Böylece nesnelerin kopyalanarak ifşa olmaması gereken field'larının ifşa olmasına yol açabilir.

Örneğin Java dilinde bu durum örneklenmiştir.

Java - Güvensiz Kod Bloğu:

```
// Class B Klonlama Kontrollerini ve
// Kısıtlamalarını Silerek A Nesnelerini
// Kaçırır.

public class A implements Cloneable {

    public Object clone() {

        A cloneObj;

        /*

            Bu metot içerisinde cloneObj nesnesine this
            nesnesi klonlanır.

            Klonlama sırasında kısıtlamalar ve doğrulamalar
            ayrıca uygulanır.

        */
        return A;
    }
}

public class B extends A {

    public Object clone() {

        B cloneObj;

        /*

            Bu metot içerisinde cloneObj nesnesine this
            nesnesi klonlanır.

            A süper sınıfındaki kısıtlamalar ve doğrulamalar
            clone() yeniden tanımlandığından tamamen bypass'lanır.

        */

        return B;
    }
}
```

Java - Güvenli Kod Bloğu:

```
// Class B Klonlama Kontrollerini ve
// Kısıtlamalarını Atlatamaz.

public class A implements Cloneable {

    public final Object clone() {

        A cloneObj;

        /*

            Bu metot içerisinde cloneObj nesnesine this
            nesnesi klonlanır.

            Klonlama sırasında kısıtlamalar ve doğrulamalar
            ayrıca uygulanır.

        */
        return A;
    }
}

public class B extends A {

    /*

      A süper sınıfındaki klonlama kısıtlamaları ve
      doğrulamaları clone() metodu “override” edilerek
      bypass'lanamaz. Çünkü süper class’ın clone()
      metodu final tanımlandığı için override edilemezdir
      / değiştirilemezdir.

    */

    public Object clone() {

        B cloneObj;

        return B;
    }
}
```

Java güvensiz kod bloğunda A sınıfı Cloneable sınıfını implement’e etmektedir. Dolayısıyla A sınıfına clone() metodu gelmektedir. clone() metodu içerisinde nesne kopyalama öncesi

kontroller uygulandığı ve sonra kopyalama işleminin field field uygulandığı varsayılmaktadır. A sınıfından kalıtım yoluyla gelen B sınıfı ise clone() metodunu override etmektedir. Bu nokta kritiktir. Çünkü clone() metodu override edilerek kopyalama işlemi için super class'da uygulanan kontroller atlatılmaktadır ve doğrudan kopyalama yapılabilmektedir. Bu ise nesne kaçırma adı verilen açıklığın doğmasına sebeptir. Bu şekilde nesneler kopyalanarak açığa çıkmaması gereken field değerleri açığa çıkabilir.

Java güvenli kod bloğunda ise A sınıfı yine Cloneable sınıfını implement'e etmektedir. Dolayısıyla A sınıfına clone() metodu gelmektedir. clone() metodu içerisinde nesne kopyalama öncesi kontroller uygulandığı ve sonra kopyalama işleminin field field uygulandığı tekrardan varsayılmaktadır. A sınıfından kalıtım yoluyla gelen B sınıfı ise clone() metodunu bu sefer tekrardan tanımlayamayacaktır (override edemeyecektir). Çünkü süper sınıf (super class) A'da clone() metodu final tanımlanmıştır. Bu ise alt sınıflarda (subclasses) clone() metodunun override edilmesini yasaklar. Böylece harici bir sınıf üzerinden harici bir nesne kopyalama denendiğinde ve hassas veriler ortaya çıkarılmaya çalışıldığında ana kopyalama metodu kısıtlamaları her daim uygulanabilir olacağından ifşaların önüne geçilebilecektir.

> Bilgi:
>
> clone() metodu bir nesnenin kopyasını oluşturmaya yarar. Varsayılanda sığ kopyalama yapar. Derin kopyalama gerekirse clone() metodu içerisinde uygulanabilir.

Bir class (sınıf) Cloneable interface'ini implement'e ettiğinde ve public clone() metodunu final olarak kullanmadan tanımladığında Nesne Kaçırma (CWE-491) açıklığı ortaya çıkar. Çünkü bu durum subclass'lara clone() metodunu override edebilmelerine yol açar ve nesne field değerlerinin ifşasına neden olabilecek güvenlik riski doğurur.

Kurum uygulamada Nesne Kaçırma (CWE-491) açıklığı tespit edilmiştir.

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

Açıklığın Önlemi:

Bir Cloneable nesnesi tanımlanırken daima clone() metodu final ayarlanmalıdır. Böylece herhangi bir clone() metodu overridding'inin (yani kısıtlamaların ve doğrulamaların) bypass'lanmasının önüne geçilebilir.

Referanslar:

1. https://en.wikipedia.org/wiki/Object_copying
2. https://stackoverflow.com/questions/184710/what-is-the-difference-between-a-deep-copy-and-a-shallow-copy

3. http://cwe.mitre.org/data/definitions/491.html
4. https://appsec.backslash.security/cwe/491
5. https://www.scaler.com/topics/final-method-in-java/