1.1.1 Hassas Veri İçeren Serileştirilebilir Sınıf Kullanılması (Serializable Class Containing Sensitive Data) (CWE-499)

Açıklık Önem Derecesi: Düşük

Açıklığın Etkisi: Bilgi sızıntısı

Açıklığın Barındıran Dosyalar/Satırlar:

Proje Dosyası/Dosya Adı	Satır Numarası

Açıklığın Açıklaması:

Nesne yönelimli programlama dillerinde nesne serileştirme adlı verilen bir mekanizma sunulur. Bu mekanizmaya göre kaynak kodda oluşturulmuş ve RAM'e yerleşmiş bir nesne tüm field'ları (değerleri dahil) ve metotları ile beraber bir dosyaya encode'lanarak kaydedilebilir. Böylece sonradan uygulamanın farklı bir noktasında bu nesne dosyadan deserileştirme yöntemiyle çekilerek tekrardan kullanılabilir ve RAM'e tekrardan yerleştirilebilir.

Örneğin Java dilinde bir nesnenin serileştirilebilmesi örneğine göz atalım.

```
public class Employee implements java.io.Serializable {
   public String name;
   public String address;
   public int SSN;             // Social Security Number
   public int number;

   public void mailCheck() {
      System.out.println("Mailing a check to " + name + " " + address);
   }
}
```

Bir nesne serileştirilebilir olacaksa nesnenin sınıf tanımında

- implements ile java.io.Serializable yer almalıdır
- field'lar transient olmamalıdır

Örneğe bakılacak olursa Employee sınıfından türeyen nesneler "implements java.io.Serializable" tanımı dolayısıyla serileştirilebilirdirler. Böylece daha önceden tanımlanmış nesne tüm barındırdığı durum ile beraber sonradan deserileştirme ile tekrardan kullanılabilir.

Serileştirme bellekteki bir nesneyi bytestream olarak, XML olarak, JSON olarak veya başka bir formatta bir dosyaya kaydetme işlemine denir. Bu işlem nesneyi bir başka platforma iletmek veya aynı platformda farklı konumlardan çağırarak kullanmak için depolamak amacıyla kullanılabilir. Ancak bu durum nesnenin içeriklerinin ifşa olması olasılığını doğurur.

Java dilinde serileştirilebilir tanımlanmış bir sınıf içerisindeki hassas veri taşıyan bir field için nesne serileştirme sırasında hariç tutulsun istisnası transient anahtar kelimesi ile eklenebilir.

```
public class Employee implements java.io.Serializable {
   public String name;
   public String address;
   public transient int SSN;         // Social Security Number
   public int number;

   public void mailCheck() {
      System.out.println("Mailing a check to " + name + " " + address);
   }
}
```

Örnekte gösterildiği gibi transient anahtar kelimesi ile SSN hassas değerini içeren field mevcut veya gelecekteki serileştirme işlemlerinden hariç tutulmuştur. Bu kullanım geçerlidir, fakat eğer sınıf içerisinde getter & setter tanımları bu field için de yapılmışsa bu durumda getter nedeniyle transient anlamsız kalacaktır ve hassas SSN değerini içeren field yine serileştirilmiş olacaktır.

Java dillerinde

- Bir sınıf hassas veri içerecek field içerdiğinde,
- Bu sınıf veya parent sınıf Serializable implement edecek şekilde tanımlandığında ve
- Hassas veri içerecek field transient tanımlanmadığında veya transient tanımlandığında ama getter tanımlandığında

“Hassas Veri İçeren Serileştirilebilir Sınıf Kullanılması (CWE-499)” açıklığı vardır denir. Açıklığın daha somut anlaşılabilmesi adına şu şekilde güvensiz kod bloğu örneği ilaveten paylaşılabilir.

Java - Güvensiz Kod Bloğu:

```
public class Purchase implements Serializable {

   private String creditCard;
   private Date expDate;
   private int CCV;

   // .. //
}
```

Purcase (satın alma) sınıfı Serializable'ı implement'e etmektedir ve 3 adet hassas veri taşıyacak field içermektedir. Bunlar kredi kart numarası, son kullanım tarihi ve CCV kodu. Bu kullanım bu sınıftan türeyecek nesnelerin iletim için veya depolama için farklı noktalardan deserileştirme ile bilgilerinin kullanılması imkanını verecektir ama olası bir ayrı güvenlik kusuru meydana geldiğinde bilgilerin ifşa olmasının yolunu da açacaktır.

Kurum uygulamada "Hassas Veri İçeren Serileştirilebilir Sınıf Kullanılması (CWE-499)" açıklığı tespit edilmiştir.

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

Açıklığın Önlemi:

Açıklığın giderilmesi noktasında tavsiyeler şu şekildedir:

- Serileştirilebilir nesneler içerisinde hassas veri depolamanmamalıdır.
- Eğer serileştirilebilir nesneler içerisinde hassas veri depolamak gerçekten gerekliyse hassas veriler tehlikeye atılmamalıdır. Bunun için data-at-rest (veri depolama) ve data-in-transit (veri iletimi) güvenlik prensipleri uygulanmalıdır. Örneğin veriler kriptografik olarak şifrelenerek depolanabilir ve taşınabilirler.

Referanslar:

1. https://www.swatips.com/articles/20210830.html
2. https://cwe.mitre.org/data/definitions/499.html
3. https://www.tutorialspoint.com/java/java_serialization.htm
4. https://www.digitalguardian.com/blog/data-protection-data-in-transit-vs-data-at-rest