1.1.1 Güvenli Olmayan Log Alma Nesnesi (Not Static Final Logger) (CWE-398)

Açıklık Önem Derecesi: Düşük

Açıklığın Etkisi: Log verisi sızıntısı

Açıklığın Barındıran Dosyalar/Satırlar:

Proje Dosyası/Dosya Adı	Satır Numarası

Açıklığın Açıklaması:

Nesne yönelimli programlamada "**private**" belirteci üyelerin (field'ların, metotların veya constructor'ların) yalnızca deklare edildikleri aynı class'da kullanılabileceği kısıtını koyar. Bu belirteç genellikle encapsulation'ı uygulamak ve diğer class'lardan yetkisiz erişimleri önlemek için kullanılır.

Nesne yönelimli programlamada - daha spesifik ifadeyle Java dilinde - "**final**" belirteci üyelerin (field'ların, metotların veya class'ların) modifiye edilemeyeceği ve extend edilemeyeceği kısıtını koyar.

Nesne yönelimli programlamada "**static**" belirteci bir class'tan tanımlanan tüm nesneler (instance'lar) için her birine bellekte ayrı üyeler (field'lar veya metot) tahsis etme yerine bellekte ortak üyeler (field'lar veya metotlar) tahsis etme kısıtı koyar. Bu nedenle static üyeleri kullanmak için nesne tanımlaması gerekmez. Ortak olduklarından doğrudan class ile çağırılabilirler.

Log alma nesneleri bu üç unsur ile tanımlanmazlarsa güvensiz log'lama mekanizması vardır denir.

- Log alma nesnelerinde "private" belirtecinin önemi:

  Log alma nesnesi ve verisinin deklare edildiği class harici diğer class'larca çalınmasını önler. Deklare edildikleri class sınırları dahilinde gizliliği ve güvenliği sağlanır.

- Log alma nesnelerinde "static" belirtecinin önemi:

  Bir class'ın tüm instance'larında yalnızca bir adet log alma nesnesine ihtiyaç vardır. Bu nedenle en az ayrıcalık prensibi (principle of least privilege) gereği static kullanılmalıdır.

- Log alma nesnelerinde "final" belirtecinin önemi:

  Bir class'ın yaşam ömrü boyunca log alma nesnesinin değişmesine gerek yoktur. Bu nedenle en az ayrıcalık prensibi (principle of least privilege) gereği final kullanılmalıdır.

Örneğin güvensiz ve güvenli tanımlanmış log alma nesnelerine java teknolojisinden örnekler verilmiştir:

Java - Güvensiz Kod Bloğu:

```
// GÜVENSİZ ÖRNEK

private final Logger logger = Logger.getLogger(MyClass.class);
```

Java - Güvenli Kod Bloğu:

```
// GÜVENLİ ÖRNEK

private final static logger = Logger.getLogger(MyClass.class);
```

Güvensiz kod bloğu örneğinde gizlilik private belirteci ile ve değişmezlik final belirteci ile sağlanmaktadır, fakat static tanımlama mevcut değildir. Bu durum en az ayrıcalık prensibine (principle of least privilege) aykırıdır. Güvenli kod bloğunda ise 3 anahtar unsurun da tanıma dahil edildiği görülmektedir.

Tek bir (static) log alıcı nesne kullanmak, gizli (private) yapmak ve sabit (final) tanımlamak iyi bir programlama pratiğidir. Bu pratiğe uyulmadığında olası olumsuz durumlarda log verilerinin sızıntısı yaşanabilir.

Kurum uygulamada güvensiz log mekanizması tespit edilmiştir:

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

Açıklığın Önlemi:

Log alıcı nesneler static, final ve private tanımlanmalıdır.

Uyarı:

İhtiyaç doğrultusunda private yerine protected tanımlama da uygulanabilir, fakat public asla tanımlanmamalıdır.

Referanslar:

1. https://cwe.mitre.org/data/definitions/398
2. https://www.scholarhat.com/tutorial/java/java-access-modifier-default-private-protected-public
3. https://www.w3schools.com/java/ref_keyword_private.asp
4. https://www.datacamp.com/doc/java/private
5. https://www.programiz.com/java-programming/examples/access-private-members
6. https://www.geeksforgeeks.org/final-keyword-in-java/
7. https://www.w3schools.com/java/ref_keyword_static.asp
8. https://stackoverflow.com/questions/6653520/why-do-we-declare-loggers-static-final
9. https://vulncat.fortify.com/en/detail?category=Poor%20Logging%20Practice&subcategory=Logger%20Not%20Declared%20Static%20Final#Java%2fJSP
10. https://owasp.org/www-community/vulnerabilities/Poor_Logging_Practice
11. https://topic.alibabacloud.com/a/why-it-is-good-practice-to-declare-loggers-private-static-and-final_8_8_31689121.html
12. https://derscanner.com/vulnerability-database/Java-:-Logger-not-static-final