MET04-J. Do not increase the accessibility of overridden or hidden methods

Increasing the accessibility of overridden or hidden methods permits a malicious subclass to offer wider access to the restricted method than was originally intended. Consequently, programs must override methods only when necessary and must declare methods final whenever possible to prevent malicious subclassing. When methods cannot be declared final, programs must refrain from increasing the accessibility of overridden methods.

The access modifier of an overriding or hiding method must provide at least as much access as the overridden or hidden method (*The* Java *Language Specification*, §8.4.8.3, "Requirements in Overriding and Hiding" [JLS 2015]). The following table lists the allowed accesses.

Overridden/Hidden Method Modifier	Overriding/Hiding Method Modifier
`public`	`public`
`protected`	`protected` or `public`
default	default or `protected` or `public`
`private`	Cannot be overridden

Noncompliant Code Example

This noncompliant code example demonstrates how a malicious subclass `Sub` can both override the `doLogic()` method of the superclass and increase the accessibility of the overriding method. Any user of `Sub` can invoke the `doLogic` method because the base class `Super` defines it to be `protected`, consequently allowing class `Sub` to increase the accessibility of `doLogic()` by declaring its own version of the method to be public.

```
class Super {
  protected void doLogic() {
    System.out.println("Super invoked");
  }
}

public class Sub extends Super {
  public void doLogic() {
    System.out.println("Sub invoked");
    // Do sensitive operations
  }
}
```

Compliant Solution

This compliant solution declares the `doLogic()` method final to prevent malicious overriding:

```
class Super {
  protected final void doLogic() { // Declare as final
    System.out.println("Super invoked");
    // Do sensitive operations
  }
}
```

Exceptions

MET04-J-EX0: For classes that implement the `java.lang.Cloneable` interface, the accessibility of the `Object.clone()` method should be increased from `protected` to `public` [SCG 2009].

Risk Assessment

Subclassing allows weakening of access restrictions, which can compromise the security of a Java application.

Rule	Severity	Likelihood	Remediation Cost	Priority	Level
MET04-J	Medium	Probable	Medium	P8	L2

Automated Detection

Detecting violations of this rule is straightforward.

Tool	Version	Checker	Description
Parasoft Jtest	10.3	**OOP.OPM**	Implemented

Related Guidelines

MITRE CWE	CWE-487, Reliance on Package-Level Scope
Secure Coding Guidelines for Java SE, Version 5.0	Guideline 4-1 / EXTEND-1: Limit the accessibility of classes, interfaces, methods, and fields

Bibliography

[JLS 2015]	§8.4.8.3, "Requirements in Overriding and Hiding"
[SCG 2009]	