# CWE Detail – CWE-492

## Description

Inner classes are translated into classes that are accessible at package scope and may expose code that the programmer intended to keep private to attackers.

## Extended Description

Inner classes quietly introduce several security concerns because of the way they are translated into Java bytecode. In Java source code, it appears that an inner class can be declared to be accessible only by the enclosing class, but Java bytecode has no concept of an inner class, so the compiler must transform an inner class declaration into a peer class with package level access to the original outer class. More insidiously, since an inner class can access private fields in its enclosing class, once an inner class becomes a peer class in bytecode, the compiler converts private fields accessed by the inner class into protected fields.

## Threat-Mapped Scoring

Score: 1.8

Priority: P4 - Informational (Low)

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: Read Application Data — Notes: "Inner Classes" data confidentiality aspects can often be overcome.

## Potential Mitigations

**•** Implementation: Using sealed classes protects object-oriented encapsulation paradigms and therefore protects code from being extended in unforeseen ways. (Effectiveness: N/A)

**•** Implementation: Inner Classes do not provide security. Warning: Never reduce the security of the object from an outer class, going to an inner class. If an outer class is final or private, ensure that its inner class is private as well. (Effectiveness: N/A)

## Applicable Platforms

**•** Java (Class: None, Prevalence: Undetermined)

## Demonstrative Examples

**•** N/A

**•** Although this is an acceptable use of inner classes it demonstrates one of the weaknesses of inner classes that inner classes have complete access to all member variables and methods of the enclosing class even those that are declared private and protected. When inner classes are compiled and translated into Java bytecode the JVM treats the inner class as a peer class with package level access to the enclosing class.

**•** However as demonstrated in the previous example, because InterestAdder is a non-static member inner class of the BankAccount class, InterestAdder also has access to the private member variables of the BankAccount class - including the sensitive data contained in the private member variables for the bank account owner's name, Social Security number, and the bank account number.

**•** As with the previous examples a solution to this problem would be to use a static inner class, a local inner class or an anonymous inner class. An alternative solution would be to have the applet implement the action listener rather than using it as an inner class as shown in the following example.

## Notes

**•** Other: Mobile code, in this case a Java Applet, is code that is transmitted across a network and executed on a remote machine. Because mobile code developers have little if any control of the environment in which their code will execute, special security concerns become relevant. One of the biggest environmental threats results from the risk that the mobile code will run side-by-side with other, potentially malicious, mobile code. Because all of the popular web browsers execute code from multiple sources together in the same JVM, many of the security guidelines for mobile code are focused on preventing manipulation of your objects' state and behavior by adversaries who have access to the same virtual machine where your program is running.