ENV04-J. Do not disable bytecode verification

When Java source code is compiled, it is converted into bytecode, saved in one or more class files, and executed by the Java Virtual Machine (JVM). Java class files may be compiled on one machine and executed on another machine. A properly generated class file is said to be *conforming*. When the JVM loads a class file, it has no way of knowing whether the class file is conforming. The class file could have been created by some other process, or an attacker may have tampered with a conforming class file.

The Java bytecode verifier is an internal component of the JVM that is responsible for detecting nonconforming Java bytecode. It ensures that the class file is in the proper Java class format, that illegal type casts are avoided, that operand stack underflows are impossible, and that each method eventually removes from the operand stack everything pushed by that method.

Users often assume that Java class files obtained from a trustworthy source will be conforming and, consequently, safe for execution. This belief can erroneously lead them to see bytecode verification as a superfluous activity for such classes. Consequently, they might disable bytecode verification, undermining Java's safety and security guarantees. The bytecode verifier must not be suppressed.

Noncompliant Code Example

The bytecode verification process runs by default. The `-Xverify:none` flag on the JVM command line suppresses the verification process. This noncompliant code example uses the flag to disable bytecode verification:

```
java -Xverify:none ApplicationName
```

Compliant Solution

Most JVM implementations perform bytecode verification by default; it is also performed during dynamic class loading.

Specifying the `-Xverify:all` flag on the command line requires the JVM to enable bytecode verification (even when it would otherwise have been suppressed), as shown in this compliant solution:

```
java -Xverify:all ApplicationName
```

Exceptions

ENV04-J-EX0: On Java 2 systems, the primordial class loader is permitted to omit bytecode verification of classes loaded from the boot class path. These system classes are protected through platform and file system protections rather than by the bytecode verification process.

Risk Assessment

Bytecode verification ensures that the bytecode contains many of the security checks mandated by the *Java Language Specification*. Omitting the verification step could permit execution of insecure Java code.

Rule	Severity	Likelihood	Remediation Cost	Priority	Level
ENV04-J	High	Likely	Low	**P27**	**L1**

Automated Detection

Static checking of this rule is not feasible in the general case.

Android Implementation Details

Under the default settings, bytecode verification is enabled on the Dalvik VM. To change the settings, use the adb shell to set the appropriate system property: for example, `adb shell setprop dalvik.vm.dexopt-flags v=a` or pass `-Xverify:all` as an argument to the Dalvik VM.

Bibliography

[Oaks 2001]	"The Bytecode Verifier"
[Pistoia 2004]	Section 7.3, "The Class File Verifier"