FIO01-J. Create files with appropriate access permissions

Files on multiuser systems are generally owned by a particular user. The owner of the file can specify which other users on the system should be allowed to access the contents of these files.

These file systems use a privileges and permissions model to protect file access. When a file is created, the file access permissions dictate who may access or operate on the file. When a program creates a file with insufficiently restrictive access permissions, an attacker may read or modify the file before the program can modify the permissions. Consequently, files must be created with access permissions that prevent unauthorized file access.

Noncompliant Code Example

The constructors for `FileOutputStream` and `FileWriter` do not allow the programmer to explicitly specify file access permissions. In this noncompliant code example, the access permissions of any file created are implementation-defined and may not prevent unauthorized access:

```
Writer out = new FileWriter("file");
```

Compliant Solution (Java 1.6 and Earlier)

Java 1.6 and earlier lack a mechanism for specifying default permissions upon file creation. Consequently, the problem must be avoided or solved using some mechanism external to Java, such as by using native code and the Java Native Interface (JNI).

Compliant Solution (POSIX)

The I/O facility `java.nio` provides classes for managing file access permissions. Additionally, many of the methods and constructors that create files accept an argument allowing the program to specify the initial file permissions.

The `Files.newByteChannel()` method allows a file to be created with specific permissions. This method is platform-independent, but the actual permissions are platform-specific. This compliant solution defines sufficiently restrictive permissions for POSIX platforms:

```
Path file = new File("file").toPath();

// Throw exception rather than overwrite existing file
Set<OpenOption> options = new HashSet<OpenOption>();
options.add(StandardOpenOption.CREATE_NEW);
options.add(StandardOpenOption.APPEND);

// File permissions should be such that only user may read/write file
Set<PosixFilePermission> perms =
    PosixFilePermissions.fromString("rw-------");
FileAttribute<Set<PosixFilePermission>> attr =
    PosixFilePermissions.asFileAttribute(perms);

try (SeekableByteChannel sbc =
         Files.newByteChannel(file, options, attr)) {
  // Write data
};
```

Exceptions

FIO01-J-EX0: When a file is created inside a directory that is both secure and unreadable by untrusted users, that file may be created with the default access permissions. This could be the case if, for example, the entire file system is trusted or is accessible only to trusted users (see FIO00-J. Do not operate on files in shared directories for the definition of a secure directory).

FIO01-J-EX1: Files that do not contain privileged information need not be created with specific access permissions.

Risk Assessment

If files are created without appropriate permissions, an attacker may read or write to the files, possibly resulting in compromised system integrity and information disclosure.

Rule	Severity	Likelihood	Remediation Cost	Priority	Level
FIO01-J	Medium	Probable	High	**P4**	**L3**

Related Guidelines

SEI CERT C++ Coding Standard	VOID FIO06-CPP. Create files with appropriate access permissions
SEI CERT C Coding Standard	FIO06-C. Create files with appropriate access permissions
ISO/IEC TR 24772:2010	Missing or Inconsistent Access Control [XZN]
MITRE CWE	CWE-279, Incorrect Execution-Assigned Permissions CWE-276, Incorrect Default Permissions CWE-732, Incorrect Permission Assignment for Critical Resource

Android Implementation Details

Creating files with weak permissions may allow malicious applications to access the files.

Bibliography

[API 2014]	
[CVE]	
[Dowd 2006]	Chapter 9, "UNIX 1: Privileges and Files"
[J2SE 2011]	
[OpenBSD]	
[Open Group 2004]	"The `open` Function" "The `umask` Function"
[Viega 2003]	Section 2.7, "Restricting Access Permissions for New Files on UNIX"