IDS50-J. Use conservative file naming conventions

File and path names containing particular characters or character sequences can cause problems when used in the construction of a file or path name:

- Leading dashes: Leading dashes can cause problems when programs are called with the file name as a parameter because the first character or characters of the file name might be interpreted as an option switch.
- Control characters, such as newlines, carriage returns, and escape: Control characters in a file name can cause unexpected results from shell scripts and in logging.
- Spaces: Spaces can cause problems with scripts and when double quotes are not used to surround the file name.
- Invalid character encodings: Character encodings can make it difficult to perform proper validation of file and path names. (See IDS11-J. Perform any string modifications before validation).
- Namespace prefixing and conventions: Namespace prefixes can cause unexpected and potentially insecure behavior when included in a path name.
- Command interpreters, scripts, and parsers: Characters that have special meaning when processed by a command interpreter, shell, or parser.

As a result of the influence of MS-DOS, 8.3 file names of the form `xxxxxxxx.xxx`, where `x` denotes an alphanumeric character, are generally supported by modern systems. On some platforms, file names are case sensitive, and on other platforms, they are case insensitive. VU#439395 is an example of a vulnerability resulting from a failure to deal appropriately with case sensitivity issues [VU#439395]. Developers should generate file and path names using a safe subset of ASCII characters and, for security critical applications, only accept names that use these characters.

Noncompliant Code Example

In the following noncompliant code example, unsafe characters are used as part of a file name.

```
File f = new File("A\uD8AB");
OutputStream out = new FileOutputStream(f);
```

A platform is free to define its own mapping of the unsafe characters. For example, when tested on an Ubuntu Linux distribution, this noncompliant code example resulted in the following file name:

```
A?
```

Compliant Solution

Use a descriptive file name, containing only the subset of ASCII previously described.

```
File f = new File("name.ext");
OutputStream out = new FileOutputStream(f);
```

Noncompliant Code Example

This noncompliant code example creates a file with input from the user without sanitizing the input.

```
public static void main(String[] args) throws Exception {
  if (args.length < 1) {
    // Handle error
  }
  File f = new File(args[0]);
  OutputStream out = new FileOutputStream(f);
  // ...
}
```

No checks are performed on the file name to prevent troublesome characters. If an attacker knew this code was in a program used to create or rename files that would later be used in a script or automated process of some sort, the attacker could choose particular characters in the output file name to confuse the later process for malicious purposes.

Compliant Solution

This compliant solution uses a whitelist to reject file names containing unsafe characters. Further input validation may be necessary, for example, to ensure that a file or directory name does not end with a period.

```
public static void main(String[] args) throws Exception {
  if (args.length < 1) {
    // Handle error
  }
  String filename = args[0];

  Pattern pattern = Pattern.compile("[^A-Za-z0-9._]");
  Matcher matcher = pattern.matcher(filename);
  if (matcher.find()) {
    // File name contains bad chars; handle error
  }
  File f = new File(filename);
  OutputStream out = new FileOutputStream(f);
  // ...
}
```

Exceptions

FIO99-J-EX0: A program may accept a file or path name that uses "unsafe" characters provided that the developer has determined that the file is not used in a restricted sink such as a command interpreter, shell, parser,logger, or other complex subsystem that attaches a particular meaning to these characters.

Risk Assessment

Failing to use only a safe subset of ASCII can result in misinterpreted data.

Rule	Severity	Likelihood	Remediation Cost	Priority	Level
IDS05-J	medium	unlikely	medium	**P4**	**L3**

Automated Detection

Tool	Version	Checker	Description
The Checker Framework	2.1.3	**Tainting Checker**	Trust and security errors (see Chapter 8)

Related Guidelines

SEI CERT C Coding Standard	MSC09-C. Character encoding: Use subset of ASCII for safety
SEI CERT C++ Coding Standard	VOID MSC09-CPP. Character encoding: Use subset of ASCII for safety
ISO/IEC TR 24772:2010	Choice of Filenames and Other External Identifiers [AJN]
MITRE CWE	CWE-116, Improper encoding or escaping of output

Bibliography

ISO/IEC 646-1991	ISO 7-Bit Coded Character Set for Information Interchange
[Kuhn 2006]	UTF-8 and Unicode FAQ for UNIX/Linux
[Wheeler 2003]	5.4, "File Names"
[VU#439395]