IDS06-J. Exclude unsanitized user input from format strings

The `java.io` package includes a `PrintStream` class that has two equivalent formatting methods: `format()` and `printf()`. `System.out` and `System.err` are `PrintStream` objects, allowing `PrintStream` methods to be invoked on the standard output and error streams. The risks from using these methods are not as high as from using similar functions in C or C++ [Seacord 2013]. The standard library implementations throw an exception when any conversion argument fails to match the corresponding format specifier. Although throwing an exception helps mitigate against exploits, if untrusted data is incorporated into a format string, it can result in an information leak or allow a denial-of-service attack. Consequently, unsanitized input from an untrusted source must never be incorporated into format strings.

Noncompliant Code Example

This noncompliant code example leaks information about a user's credit card. It incorporates untrusted data in a format string.

```
class Format {
  static Calendar c = new GregorianCalendar(1995, GregorianCalendar.MAY, 23);
  public static void main(String[] args) {
    // args[0] should contain the credit card expiration date
    // but might contain %1$tm, %1$te or %1$tY format specifiers
    System.out.format(
      args[0] + " did not match! HINT: It was issued on %1$terd of some month", c
    );
  }
}
```

In the absence of proper input validation, an attacker can determine the date against which the input is verified by supplying an input string that includes the `%1$tm`, `%1$te`, or `%1$tY` format specifiers. In this example, these format specifiers print 05 (May), 23 (day), and 1995 (year), respectively.

Compliant Solution

This compliant solution excludes untrusted user input from the format string. Although `arg[0]` still may contain one or more format specifiers, they are now rendered inert.

```
class Format {
  static Calendar c =
    new GregorianCalendar(1995, GregorianCalendar.MAY, 23);
  public static void main(String[] args) {
    // args[0] is the credit card expiration date
    // Perform comparison with c,
    // if it doesn't match, print the following line
    System.out.format(
      "%s did not match! HINT: It was issued on %terd of some month",
      args[0], c
    );
  }
}
```

Risk Assessment

Incorporating untrusted data in a format string may result in information leaks or allow a denial-of-service attack.

Rule	Severity	Likelihood	Remediation Cost	Priority	Level
IDS06-J	Medium	Unlikely	Medium	**P4**	**L3**

Automated Detection

Static analysis tools that perform taint analysis can diagnose some violations of this rule.

Tool	Version	Checker	Description
The Checker Framework	2.1.3	**Tainting Checker**	Trust and security errors (see Chapter 8)
Parasoft Jtest	2020.2	**PB.API.VAFS**	Ensure the correct number of arguments for varargs methods with format strings

Related Guidelines

SEI CERT C Coding Standard	FIO30-C. Exclude user input from format strings
SEI CERT Perl Coding Standard	IDS30-PL. Exclude user input from format strings
ISO/IEC TR 24772:2013	Injection [RST]
MITRE CWE	CWE-134, Uncontrolled Format String

Bibliography

[API 2006]	Class Formatter
[Seacord 2013]	Chapter 6, "Formatted Output"
[Seacord 2015]	IDS06-J. Exclude unsanitized user input from format strings LiveLesson