**Poor Error Handling: Overly Broad Throws Development Mitigation SOP**

Poor error handling with overly broad throws makes the application vulnerable to poor error handling and recovery. Java’s exception mechanism was specifically set up to make it easy for callers to anticipate what can go wrong and write code to handle those specific issues. To throw a generic exception would defeat the purpose of that functionality.

**Defense Against Poor Error Handling: Overly Broad Throws**

Do not declare methods to throw *Exception* or *Throwable*. If the exceptions thrown by a method are not recoverable or should not generally be caught by the caller, it would be better practice to throw unchecked exceptions rather than checked exceptions. To do this, implement exception classes that extend *RuntimeException* or *Error* instead of *Exception*, or try to add a try/catch wrapper in your method to convert checked exceptions to unchecked exceptions.

**Example**

public void doExchange() throws Exception { … }

Explanation:

1. Multiple throws can look messy, however, reducing the throws to only *Exception* can inhibit the caller’s ability to understand and handle the exceptions that occur
2. Below would be a better way to catch specific throws:

public void doExchange() throws IOException, InvocationTargetException, SQLException { … }

**References**

1. [HP Enterprise Security – Poor Error Handling: Overly Broad Throws](http://www.hpenterprisesecurity.com/vulncat/en/vulncat/java/poor_error_handling_overly_broad_throws.html)