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

Poor error handling with overly broad catches makes the application vulnerable to trapping dissimilar issues of problems that should not be dealt with at that time of the program. While multiple catch blocks can get ugly and repetitive, condensing multiple catch blocks with a high-level class, such as *Exception,* can obscure exceptions that deserve special treatment or that should not be caught at that point in the program.

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

Do not catch broad exception classes like *Exception*, *Throwable*, *Error*, or *<RuntimeException>* except at the very top of the program or thread.

**Example**

try {

DoExchange();

} catch ( Exception e ) {

logger.Error( “DoExchange failed”, e );

}

Explanation:

1. While multiple catch block may not look clean, if a new obscure exception was thrown and there was only a catch block for *Exception* (as shown above), it may prevent the compiler from pointing out the issue
2. Below would be a better way to catch specific exceptions:

try {

DoExchange();

} catch ( IOException e ) {

logger.Error( “DoExchange failed”, e );

} catch ( FormatException e ) {

logger.Error( “DoExchange failed”, e );

} catch ( TimeoutException e ) {

logger.Error( “DoExchange failed”, e );

}

**References**

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