**Poor Error Handling: Empty Catch Block Development Mitigation SOP**

Poor error handling with empty catch blocks can cause an application to overlook unexpected states and conditions. This type of vulnerability comes from the developer’s assumption that either that method will never fail or that it will not matter if it does.

**Defense Against Poor Error Handling: Empty Catch Block**

At a minimum, the exception that is being thrown should be logged for developers to be able to go back and solve the problematic behavior. If the exception is being ignored because the caller cannot properly handle it or it is more convenient to drop the exception, another solution would be to throw a *RuntimeException* or *Error* because they are both unchecked exceptions, meaning they will not be verified during compile time.

**Example**

try {

return em.find( AnnotationEntity.class, id );

} catch (final NoResultException ex) {

// no match found, return nothing

}

**Explanation**

1. In the example above, if *NoResultException* were ever thrown, the program would continue executing as usual because nothing is done when the exception is reached
2. At the very least, the issue should be logged for the purpose of the program having evidence recorded that the error was reached
3. The code could be rewritten in the following way:

try {

return em.find( AnnotationEntity.class, id );

} catch (final NoResultException ex) {

logger.info( “No result was found.” );

}

**References**

1. [HP Enterprise Security – Poor Error Handling: Empty Catch Block](http://www.hpenterprisesecurity.com/vulncat/en/vulncat/java/poor_error_handling_empty_catch_block.html)