**Poor Error Handling: Unhandled SSL Exception Development Mitigation SOP**

Poor error handling from unhandled SSL exceptions can cause the vulnerabilities by causing the application to overlook unexpected states and conditions. These vulnerabilities occur when an SSL-specific exception is thrown or the exception is not explicitly handled. There are SSL-specific exceptions, such as *javax.net.ssl.SSLHandshakeException*, *javax.net.ssl.SSLKeyException*, and *javax.net.ssl.SSLPeerUnverifiedException*, that all convey important errors related to SSL connections. When these exceptions are not explicitly handled, the connection could be left in an unexpected and potentially insecure state.

**Defense Against Poor Error Handling: Unhandled SSL Exception**

Always check the error condition if an exception is thrown for evidence of its success or failure. If you are able to create a standard way for dealing with failures, the programmer will be less inclined to omit them. One way to create a standard for error handling is to write wrappers around commonly used functions that check and handle error conditions without additional programmer intervention. The use of non-wrapped equivalents can be prohibited and enforced by using custom rules if wrappers are implemented and adopted.

**Example**

try {

Certificate[] certs = session.getPeerCertificates();

…

} catch(Exception e) {

Util.displayThrowable(e, root);

}

**Explanation**

In the code above, an exception is thrown by getPeerCertificates() and the exception is not explicitly handled. Instead, any errors within the try block are caught by the catch block by a generic Exception*.*

**References**

1. [HP Enterprise Security – Poor Error Handling: Unhandled SSL Exception](http://www.hpenterprisesecurity.com/vulncat/en/vulncat/java/poor_error_handling_unhandled_ssl_exception.html)