**XML Entity Expansion Injection Development Mitigation SOP**

An XML Entity Expansion Injection vulnerability, also known as an XML Bomb, can occur when using XML parsers configured to not prevent nor limit document type definition (DTD) entity resolution. These attacks are denial of service attacks that benefit from valid and well-formed XML blocks that expand exponentially until they exhaust the server allocated resources. XML allows the definition of custom entities, which act as string substitution macros. By nesting recurrent entity resolutions, an attacker can crash the server resources.

**Defense Against XML Entity Expansion Injection**

XML parsers must be configured securely so that it does not allow DTD custom entities as part of an incoming XML document.

**Example**

private static XMLStreamReader createXmlReader(InputStream dbqXmlStream) {

XMLInputFactory factory = XMLInputFactory.newInstance();

**factory.setFeature(XMLInputFactory.IS\_SUPPORTING\_EXTERNAL\_E**

**NTITIES, false);**

try {

return factory.createXMLStreamReader(dbqXmlStream);

} catch(XMLStreamException e) {

throw new SystemFaultException(“An exception occurred”,

e);

}

}

**Explanation**

This is an example of how to correctly prevent a XML Entity Expansion Injection vulnerability because the factory sets the feature of XMLInputFactory.IS\_SUPPORTING\_EXTERNAL\_ENTITIES to false. This restricts access to external entities. If that line (bolded above) were not included, there would be no restriction to external entities. That property requires the parser to resolve external parsed entities.

**References**

1. [HP Enterprise Security – XML Entity Expansion Injection](http://www.hpenterprisesecurity.com/vulncat/en/vulncat/dotnet/xee_injection.html)
2. [Oracle Docs – Field Details](http://docs.oracle.com/javase/6/docs/api/javax/xml/stream/XMLInputFactory.html#IS%5FREPLACING%5FENTITY%5FREFERENCES)