IDS17-J. Prevent XML External Entity Attacks

Entity declarations define shortcuts to commonly used text or special characters. An entity declaration may define either an *internal* or *external entity*. For internal entities, the content of the entity is given in the declaration. For external entities, the content is specified by a Uniform Resource Identifier (URI).

Entities may be either *parsed* or *unparsed*. The contents of a parsed entity are called its *replacement text*. An unparsed entity is a resource whose contents may or may not be text, and if text, may be other than XML. Parsed entities are invoked by name using an *entity reference*; unparsed entities are invoked by name.

According to XML W3C Recommendation, section 4.4.3, "Included If Validating" [W3C 2008]:

> *When an XML processor recognizes a reference to a parsed entity, to validate the document, the processor MUST include its replacement text. If the entity is external, and the processor is not attempting to validate the XML document, the processor MAY, but need not, include the entity's replacement text.*

Because inclusion of replacement text from an external entity is optional, not all XML processors are vulnerable to external entity attacks during validation.

An *XML external entity (XXE)* attack occurs when XML input containing a reference to an external entity is processed by an improperly configured XML parser. An attacker might use an XXE attack to gain access to sensitive information by manipulating the URI of the entity to refer to files on the local file system containing sensitive data such as passwords and private user data. An attacker might launch a denial-of-service attack, for example, by specifying `/dev/random` or `/dev/tty` as input URIs, which can crash or indefinitely block a program.

Noncompliant Code Example

This noncompliant code example attempts to parse the file `evil.xml`, report any errors, and exit. However, a SAX (Simple API for XML) or a DOM (Document Object Model) parser will attempt to access the URI specified by the `SYSTEM` attribute, which means it will attempt to read the contents of the local `/dev/tty` file. On POSIX systems, reading this file causes the program to block until input data is supplied to the machine's console. Consequently, an attacker can use this malicious XML file to cause the program to hang.

```
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;

import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

class XXE {
  private static void receiveXMLStream(InputStream inStream,
                                       DefaultHandler defaultHandler)
      throws ParserConfigurationException, SAXException, IOException {
    SAXParserFactory factory = SAXParserFactory.newInstance();
    SAXParser saxParser = factory.newSAXParser();
    saxParser.parse(inStream, defaultHandler);
  }

  public static void main(String[] args) throws ParserConfigurationException,
      SAXException, IOException {
    try {
      receiveXMLStream(new FileInputStream("evil.xml"), new DefaultHandler());
    } catch (java.net.MalformedURLException mue) {
      System.err.println("Malformed URL Exception: " + mue);
    }
  }
}
```

This program is subject to a remote XXE attack if the `evil.xml` file contains the following:

```
<?xml version="1.0"?>
<!DOCTYPE foo SYSTEM "file:/dev/tty">
<foo>bar</foo>
```

Compliant Solution (`EntityResolver`)

This compliant solution defines a `CustomResolver` class that implements the interface `org.xml.sax.EntityResolver`. This interface enables a SAX application to customize handling of external entities. The customized handler uses a simple whitelist for external entities. The `resolveEntity()` method returns an empty `InputSource` when an input fails to resolve to any of the specified, safe entity source paths.

```
import java.io.IOException;
import org.xml.sax.EntityResolver;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;

class CustomResolver implements EntityResolver {
  public InputSource resolveEntity(String publicId, String systemId)
      throws SAXException, IOException {

    // Check for known good entities
    String entityPath = "file:/Users/onlinestore/good.xml";
    if (systemId.equals(entityPath)) {
      System.out.println("Resolving entity: " + publicId + " " + systemId);
      return new InputSource(entityPath);
    } else {
      // Disallow unknown entities by returning a blank path
      return new InputSource();
    }
  }
}
```

The `setEntityResolver()` method registers the instance with the corresponding SAX driver. When parsing malicious input, the empty `InputSource` returned by the custom resolver causes a `java.net.MalformedURLException` to be thrown. Note that you must create an `XMLReader` object on which to set the custom entity resolver.

```
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;

import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.DefaultHandler;

class XXE {
  private static void receiveXMLStream(InputStream inStream,
      DefaultHandler defaultHandler) throws ParserConfigurationException,
      SAXException, IOException {
    SAXParserFactory factory = SAXParserFactory.newInstance();
    SAXParser saxParser = factory.newSAXParser();

    // Create an XML reader to set the entity resolver.
    XMLReader reader = saxParser.getXMLReader();
    reader.setEntityResolver(new CustomResolver());
    reader.setContentHandler(defaultHandler);

    InputSource is = new InputSource(inStream);
    reader.parse(is);
  }

  public static void main(String[] args) throws ParserConfigurationException,
      SAXException, IOException {
    try {
      receiveXMLStream(new FileInputStream("evil.xml"), new DefaultHandler());
    } catch (java.net.MalformedURLException mue) {
      System.err.println("Malformed URL Exception: " + mue);
    }
  }
}
```

Risk Assessment

Failure to sanitize user input before processing or storing it can result in injection attacks.

Rule	Severity	Likelihood	Remediation Cost	Priority	Level
IDS17-J	Medium	Probable	Medium	P8	L2

Automated Detection

Tool	Version	Checker	Description
The Checker Framework	2.1.3	**Tainting Checker**	Trust and security errors (see Chapter 8)
Fortify	1.0	**Missing_XML_Validation**	Implemented
SonarQube	6.7	**S2755** **S4435**	Untrusted XML should be parsed with a local, static DTD XML transformers should be secured

Related Vulnerabilities

CVE-2008-2370 describes a vulnerability in Apache Tomcat 4.1.0 through 4.1.37, 5.5.0 through 5.5.26, and 6.0.0 through 6.0.16. When a `RequestDispatcher` is used, Tomcat performs path normalization before removing the query string from the URI, which allows remote attackers to conduct directory traversal attacks and read arbitrary files via a `..` (dot dot) in a request parameter.

Related Guidelines

SEI CERT C Coding Standard	STR02-C. Sanitize data passed to complex subsystems
SEI CERT C++ Coding Standard	VOID STR02-CPP. Sanitize data passed to complex subsystems
SEI CERT Perl Coding Standard	IDS33-PL. Sanitize untrusted data passed across a trust boundary
ISO/IEC TR 24772:2013	Injection [RST]
MITRE CWE	CWE-116, Improper Encoding or Escaping of Output

Android Implementation Details

This rule uses Microsoft SQL Server as an example to show a database connection. However, on Android, `DatabaseHelper` from SQLite is used for a database connection. Because Android apps may receive untrusted data via network connections, the rule is applicable.

Bibliography

[OWASP 2005]	A Guide to Building Secure Web Applications and Web Services
[OWASP 2007]	OWASP Top 10 for Java EE
[OWASP 2008]	Testing for XML Injection (OWASP-DV-008)
[Seacord 2015]	IDS17-J. Prevent XML External Entity Attacks LiveLesson
[W3C 2008]	Section 4.4.3, "Included If Validating"