## S2-020 漏洞分析

**1、漏洞介绍**

Apache Struts 2.0.0-2.3.16版本的默认上传机制是基于Commons FileUpload 1.3版本，其附加的ParametersInterceptor允许访问'class' 参数（该参数直接映射到getClass()方法），并允许控制ClassLoader。

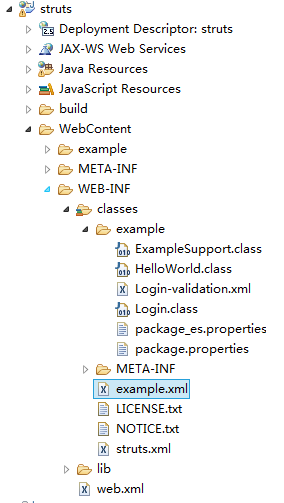
这是官网上的介绍，我分析的时候一直不明白commons fileupload 1.3在里面到底起什么作用。

**2、搭建环境**

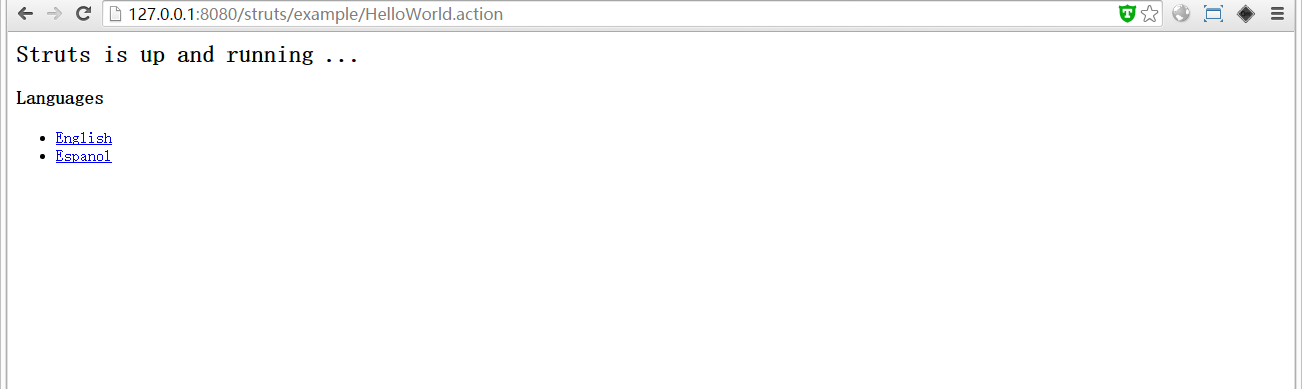
1、找到struts 2.1.14（因为刚好有现成的，我把它放到文件夹里，免得以后找不到）

2、找到里面的struts-example.war，解压后，导入到elicpse中（导入到eclipse的时候，我是先创建一个web工程，然后把example文件拷贝到相应的文件夹中，这种方法有点笨，但是我找不到好办法，直接导入整个example项目的时候出错了）

项目截图

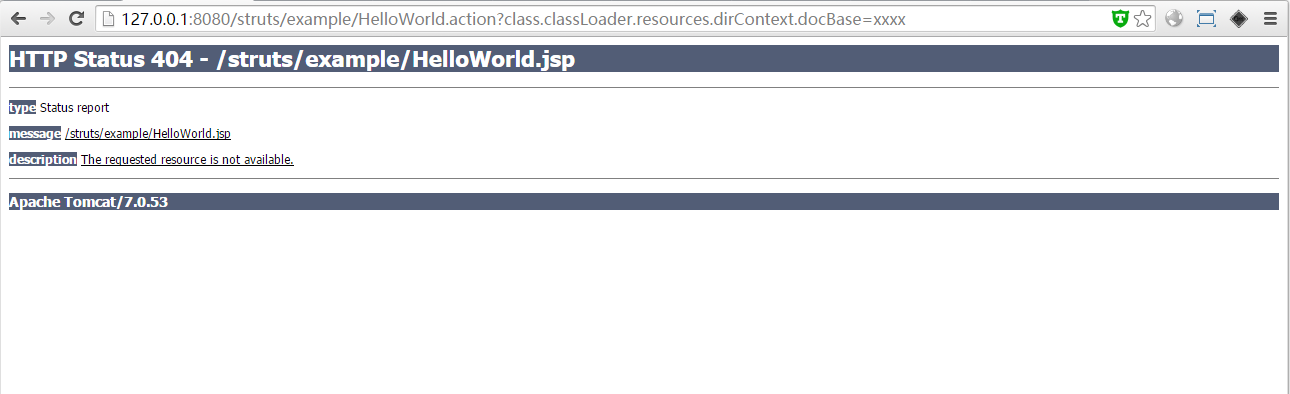


3、然后将导入的项目部署到tomcat上，并访问



4、验证POC

<http://127.0.0.1/s/example/HelloWorld.action?class.classLoader.resources.dirContext.docBase=xxxx>



后台输出信息一堆错误信息

然后这个网站就废了，访问什么url都没用了，说明我们的环境搭建成功。

**3、漏洞分析**

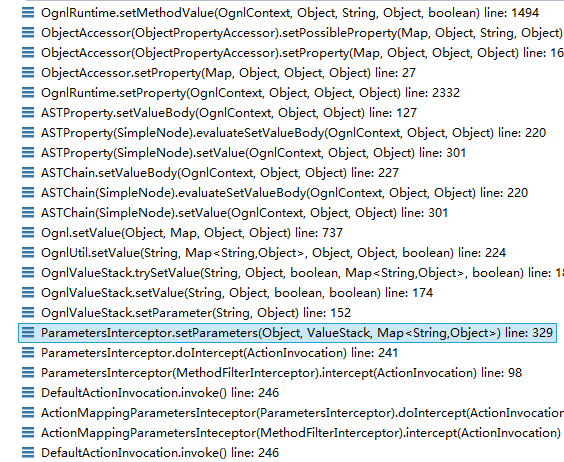
1. 分析之前，先了解一下struts执行流程，可参考<http://blog.sina.com.cn/s/blog_8f82e8280101fpbx.html>

总结来说，就是一个请求过来，struts会拦截这个请求，然后struts自身有很多个拦截器，用于分别处理这个请求。根据官方的介绍http://struts.apache.org/docs/s2-020.html，这个dos出现的位置是ParametersInterceptor，这个类或者接口。

1. 我们先确认从哪里开始分析，或者直接点说，断点打在哪里。首先我们知道Struts里面有个默认的拦截器链，定义在 struts-default.xml文件，根据官方的解释，我们知道fileuploader这个拦截器是有问题的（后来在调试的过程中，我发现出现异常的不是这个拦截器，而是params拦截器，官方的修复也是修复这个拦截器的<param name=*"excludeParams"*>属性）
2. 定位出现异常的地方，eclipse JAVA调试器和windbg一样，都可以在断点在异常抛出发生之前，具体方法可看<http://www.csdn.net/article/2012-09-03/2809495-Java-Debugging-Tips-with-Eclipse> 。我是直接跑了一遍，然后看异常，抛出的是哪一个异常，就在断点在哪个类型的异常上。



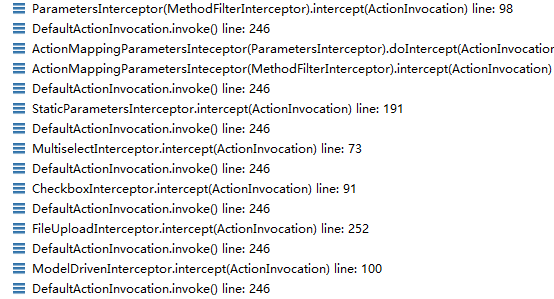
1. 异常断点后，就能查看调用栈了。

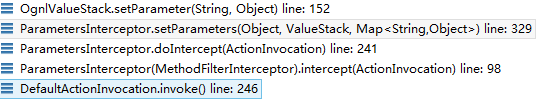


我们能够发现异常的发生在OGNL中，结合我们的POC 中的请求参数class.classLoader.resources.dirContext.docBase=xxxx，这就是OGNL在解析这个参数的时候发生了错误，OGNL的资料可以看<http://www.ibm.com/developerworks/cn/opensource/os-cn-ognl/>

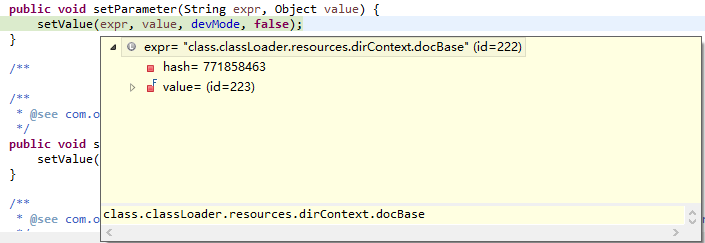
POC就表示getClass().getClassLoader().getResources.getDirContext.setDocBase(“XXX”)，至于是哪个对象，这个就要去调试的时候看了，如果对struts熟的人可能会猜action。

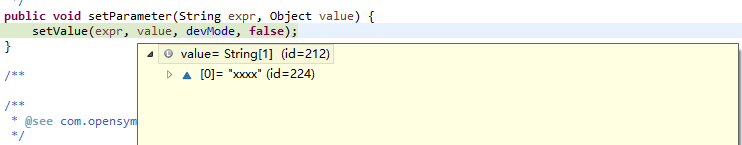
1. 然后回溯调用栈，查看参数和函数，了解这个异常发生之前struts做了些什么（这个先大致捋一捋，因为这个漏洞也比较老了，就不详细分析了，以后如果有时间，再具体一点分析）。再看看调用栈



可以看到这个拦截链是层层连接的，最终出现问题的拦截器是ParameterInterceptor

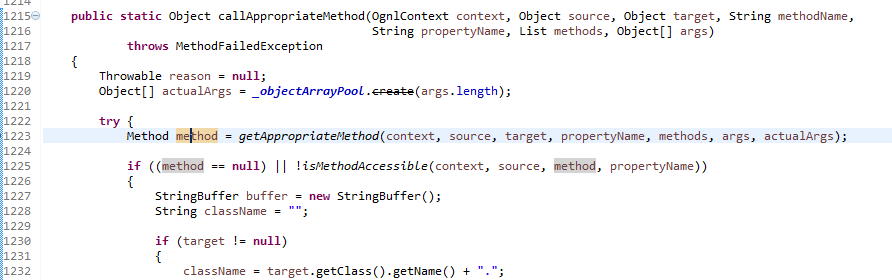
异常之前的调用setParameter,传递的参数正好是



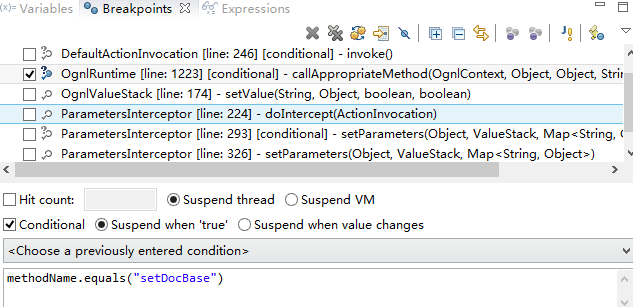


所以就是这里开始解析OGNL表达式。我们直接看抛出异常之前处理了什么。

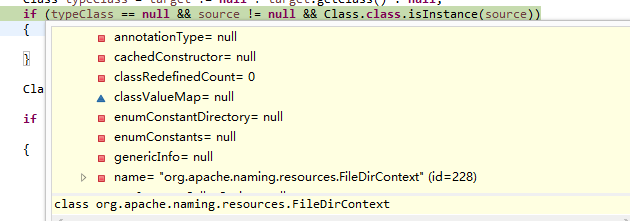
重新再发生异常的那个函数下断点。



我比较好奇的是method表示什么，所以我断在method那句话上。重新触发漏洞。不过很多别的拦截器都要走这条语句上，所以必须设置条件断点，只断我们关注的点。

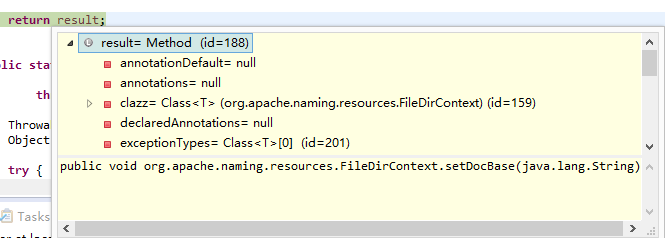


跟进



可以猜到，应该是要调用这个类的setDocBase方法

返回结果



然后调用



最后来到了异常点，说明是这个函数有问题，这个函数就是调用setDocBase方法，然后传递的参数为XXXX。这一步出错了。

先简短分析到这里。后续再补充。

**4、漏洞利用**

老漏洞，就不写利用脚本了。可以看看别人的方法：

<http://sec.baidu.com/index.php?research/detail/id/18>

http://drops.wooyun.org/papers/1377