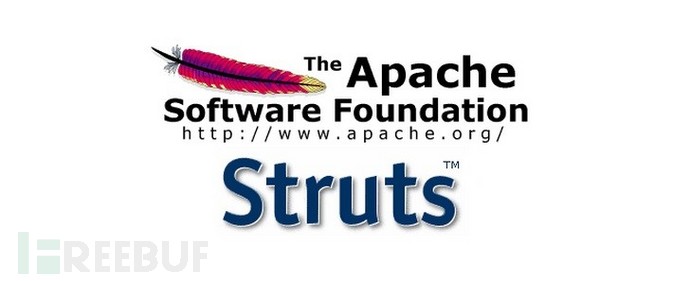
**[](http://image.3001.net/images/20160315/14580547046497.jpg)**

**0×01 Struts2标签库**

Struts2的标签库使用OGNL表达式来访问ActionContext中的对象数据。为了能够访问到ActionContext中的变量，Struts2将ActionContext设置为OGNL的上下文，并将OGNL的跟对象加入ActionContext中。

在Struts2中，如下的标签就调用了OGNL进行取值。

<**p**>parameters: <**s:property** value="#parameters.msg" /></**p**>

Struts2会解析value中的值，并当作OGNL表达式进行执行，获取到parameters对象的msg属性。

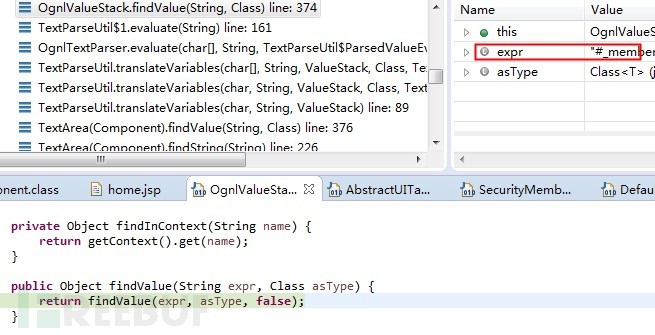
S2-029仍然是依靠OGNL进行远程代码执行。

**0×02 触发途径**

如果使用Struts2的web应用将用户提交的数据未经过滤，直接通过Struts2的标签设置成属性值，就被Struts2进行执行。

[](http://image.3001.net/images/20160317/14581976028730.jpg)

之后若页面被渲染，会传至OgnlValueStack进行执行。

[](http://image.3001.net/images/20160317/14581976284374.jpg)

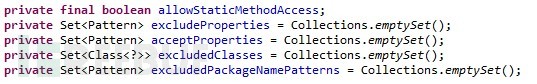
OGNL表达式会被转换成AST语法树，语法树上每个分支执行前会对其进行Accessible安全检查。

之前的Struts2远程代码执行poc，一般是通过修改StaticMethodAccess或是创建ProcessBuilder对象。

#\_memberAccess["allowStaticMethodAccess"]=true

**new** java.lang.ProcessBuilder(**new** java.lang.String[]{'cat','/etc/passwd'})).start()

不过现在的Struts封装的xwork中的OGNL已经加强了安全校验。

[](http://image.3001.net/images/20160317/14581976517520.jpg)

allowStaticMethodAccess已经变成的final属性。

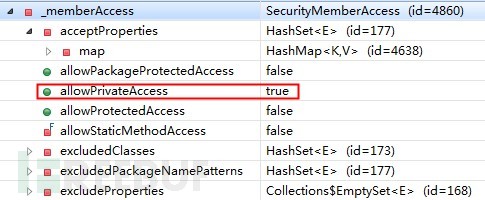
excludedClasses增加的禁用列表。

[interface ognl.ClassResolver, class java.lang.Runtime, class java.lang.ClassLoader, class com.opensymphony.xwork2.ActionContext, interface ognl.MemberAccess, class java.lang.Object, class ognl.OgnlContext, class java.lang.Class, class java.lang.Shutdown, class java.lang.System, interface ognl.TypeConverter]

excludedPackageNamePatterns同样增加了禁用列表。

[^java\.lang\..\*, ^(?!javax\.servlet\..+)(javax\..+), ^ognl.\*]

在我们的测试中allowPrivateAccess、allowProtectedAccess、excludedClasses、excludedPackageNamePatterns、acceptProperties、excludeProperties等一系列参数仍然能够修改。

[](http://image.3001.net/images/20160317/14581976641961.jpg)

修改allowPrivateAccess后，如果能够在OGNL上下文中能找到与excludedClasses类型相匹配的变量，将excludedClasses指向新的对象，就有可能绕过\_memberAccess的保护。

**0×03 结束**

Struts2作为广泛应用的Web框架，安全问题一直层出不穷。为了保证安全，在业务处理中不应信任用户提交上来的任何数据，做好安全过滤。最好能在Ognl的执行入口加入过滤黑名单，并及时将Struts更新至2.3.25。