by 空虚浪子心 http://www.inbreak.net 微博：http://t.qq.com/javasecurity  
可能是由于沟通问题，导致struts2官方对我提交的S2-012漏洞名称理解错误，漏洞描述为struts2的某个示例应用出现漏洞，但是struts2是按照框架出现漏洞修补的。而这个s2-012竟然引发了一连串血案。  
其实发这篇文章，我非常恼火，任谁手里有一个0day，捂了半天，结果又被别人公开，都会非常恼火。去年我在XCON发布的S2-012漏洞，其实struts2还存在相似的漏洞。在struts中，框架接收到的用户输入，除了参数、值以外，还有其他地方，比如文件名。这个漏洞，是struts2对url中的文件名做了解析，导致的ognl代码执行。  
这中间存在一些技术细节，下面展开分析。  
**enableOGNLEvalExpression的骗局**  
从漏洞公告上看到这个词，很容易认为是struts2把ognl表达式干掉了，可以选择终结一切。事实上禁止的是OGNL的其中一种调用方式，而这种调用方式，也只是在S2-013这里调用。  
struts2有另外一段威武的代码，真正在防守这个漏洞。

|  |
| --- |
| org.apache.struts2.views.util.DefaultUrlHelper这个类： |

原本是这样写的，代码走到translateAndEncode就会调用ognl执行，它的逻辑一共包括ognl的translate，以及urlencode这两个功能。

|  |
| --- |
| **private** String buildParameterSubstring(String name, String value) {  StringBuilder builder = **new** StringBuilder();  builder.append(translateAndEncode(name));  builder.append('=');  builder.append(translateAndEncode(value));  **return** builder.toString();  } |

补丁之后，这里被改为仅仅urlencode，不再做ognl执行。这个和enableOGNLEvalExpression没有任何关系。

|  |
| --- |
| **private** String buildParameterSubstring(String name, String value) {  StringBuilder builder = **new** StringBuilder();  builder.append(encode(name));  builder.append('=');  builder.append(encode(value));  **return** builder.toString();  } |

我没有细看内容，只看方法名的变动，就感觉可以洗洗睡了，不必往下跟进了。  
**allowStaticMethodAccess骗局**  
一直以来allowStaticMethodAccess是struts2的poc标配，从第一个poc出现开始，就一直存在。  
在2013年5月27日，也就是前几天，大家可以自行查看SVNlog，struts2做了一件很猥琐的事情，把以下代码删除了：

|  |
| --- |
| **public** **void** setAllowStaticMethodAccess(**boolean** allowStaticMethodAccess) {  **this**.allowStaticMethodAccess = allowStaticMethodAccess;  } |

这个动作直接导致一个结果，以后在OGNL的POC中执行

|  |
| --- |
| #\_memberAccess["allowStaticMethodAccess"]=**true** |

一定会报错的，因为没有set方法了。  
很有终结一切的意思，就像以后有新的OGNL漏洞，就不能写这一句了。但是我可以绕过这个东西，下面结合s2-015漏洞做个示例。  
**struts2框架s2-015吐槽**  
这个漏洞，被人公布出来，实际上，发布者一共发布了几个漏洞，包含S2-015、以及S2-012。具体地址在

|  |
| --- |
| https:*//communities.coverity.com/blogs/security/2013/05/29/struts2-remote-code-execution-via-ognl-injection* |

非常详细，某同事认为他比我分析的好，所以我就不写翻译了，大家自己看。  
后来仔细想了想，猜测老外可能遇到s2-012，导致了该文章的发布，当然，这只是我的个人YY。  
发布者手握2个0day，很不幸，我也有这两个0day，去年xcon发布了一个，之后提交了官方，但是他不知道，因为官方到今年才公开修补。  
前几天官方突然公开修补了我发布的一个0day，这个老外看到s2-012后，可能也非常恼火，因为这个漏洞和他手头分析的0day刚好相同，所以一怒之下和其他0day一起发出来了，共同组成一篇文章。可以看到，发布者直接从blog发布，之后官方才收到消息开始修补。  
这个漏洞的触发代码展现形式和s2-012非常像，所以理解了s2-012后，可以联想到这个0day，很容易通过测试出来，我当时也是看到类似的使用，随手测试就发现了。相信有很多漏洞，都是类似的情况下发现的。甚至可能不止我们手里有，其实你也非常恼火。  
S2-015的poc在老外的文章中如下：

|  |
| --- |
| http:*//127.0.0.1:8080/struts2-blank/example/$%7B%23context['xwork.MethodAccessor.denyMethodExecution']=%21%28%23\_memberAccess['allowStaticMethodAccess']=true%29,%28@java.lang.Runtime@getRuntime%28%29%29.exec%28'touch%20aaa'%29.waitFor%28%29%7D.action/* |

由于POC中存在#\_memberAccess[“allowStaticMethodAccess”]=true，所以发布者提到升级到s2-014可以缓解。  
其实发布者误解了，但是struts2开发者没有误解，所以赶紧推出了S2-015。  
但是如果不讲出来，你还是会发现那个POC在S2-014之后其实不能打，就如老外文章中所说，被缓解了。那要怎么打呢？  
**OGNL的POC有个小技巧**  
这个东西的含义，是允许静态方法执行，那么官方禁止修改这个设置，意思就是永远禁止静态方法执行。  
因为POC中的“@java.lang.Runtime@getRuntime”其实就是在执行静态方法，所以才一定要开启静态，但是这只是java代码的一种写法罢了。  
我们可以用另一个写法，绕过这个限制。

|  |
| --- |
| *//by 空虚浪子心 http://www.inbreak.net 微博：http://t.qq.com/javasecurity*  (**new** java.lang.ProcessBuilder('calc')).start() |

这段代码中，没有调用任何静态方法，仅仅是new一个对象，之后执行其中一个动态方法，所以不必allowStaticMethodAccess一样能达到执行系统命令的效果。  
这个小技巧，可以干很多事情。  
1、可以绕过某些WAF，我不告诉你是哪些，免得你拿去骗奖品。  
2、可以为以后新的OGNL代码执行铺路，避免0day来了，我们居然因为这个不会写POC。  
**S2-015的修补**  
简单说一句，这里没有什么研究价值，这次修补，官方采用了限制action的名称，只能

|  |
| --- |
| [a-z]\*[A-Z]\*[0-9]\*[.\-\_!/]\* |

**总结struts2出现过的ognl表达式输入点**  
1、request参数名  
2、request参数值  
3、request文件名  
4、request的cookie名  
5、respose的body  
惨不忍睹，好像HTTP头基本都出了问题，也没剩下多少了。一个流行框架，能够在这么多地方出现远程代码执行，真是难为struts开发者了。  
同时也问一下使用struts的同学们，你们这些年，是怎么过来的？  
在阿里巴巴，我时常分析struts2漏洞然后发报告，有时候会是0day，那就要出个补丁给各个项目用，最后等到官方发布补丁时，我们再评估是否需要重新更新回去。导致我们时常劝说开发人员尽量不要使用这个框架，尤其是项目初期评审时，发现struts2，深恶痛绝，说很多很多话用于吓唬开发人员。  
在这种趋势下，我对这个东西已经再无任何侥幸心理，决定推出一个虚拟补丁。至于阿里的真实方案，我肯定不能告诉大家，但是可以讲讲思路。  
**统一防御方案**  
首先升级到最新版本。  
在ognl这个语言的入口，加入拦截代码，一旦发现危险调用，直接干掉。  
代码原理是，在OGNL执行之前，对语句做判断，看到有黑名单的代码，就干掉。理论上，开发人员理论上不会自己写OGNL用于操作文件，执行命令等，他们最多从session中取一个值，或者在页面上取一个值。  
覆盖掉Ognl.Ognl类，添加如下代码：

|  |
| --- |
| **public** **static** Object parseExpression(String expression)  **throws** OgnlException {  *// hackedbykxlzx by 空虚浪子心 http://www.inbreak.net 微博：http://t.qq.com/javasecurity*  *//。。。下面是白名单列表，请各位同学自行搜索java危险代码，之后加入列表，实在不会的，找几个webshell看看，我肯定不会把阿里正在使用的列表告诉你们的。*  String evalMethod[] = { "Runtime", "new file" };  String methodString = **null**;  methodString = expression.toLowerCase();  **for** (**int** i = 0; i < evalMethod.length; i++) {  **if** (methodString.indexOf(evalMethod[i].toLowerCase()) > -1) {  Log.securityLog(Log.getInfo()+"|OGNL正在执行恶意语句|" + methodString  + "|看到这个消息，请联系安全工程师！！！","4700012@qq.com");  }  }    **try** {  OgnlParser parser = **new** OgnlParser(**new** StringReader(expression));  **return** parser.topLevelExpression();  } **catch** (ParseException e) {  **throw** **new** ExpressionSyntaxException(expression, e);  } **catch** (TokenMgrError e) {  **throw** **new** ExpressionSyntaxException(expression, e);  }  } |

为什么要加入QQ邮箱呢？具体原因不说，只说结果，结果是，我的邮箱可以收到0DAY，你如果真的看懂了，自己猜猜原因？  
by 空虚浪子心 http://www.inbreak.net 微博：http://t.qq.com/javasecurity