作者: 空虚浪子心, 　出处:IT专家网

原文链接：http://security.ctocio.com.cn/100/11466600.shtml

　　在漏洞发现者发布的POC中，并不能影响xwork 2.1.2之前的一些版本(这个版本之前的一些版本，下文会统称为老版本，之后的叫做新版)，例如struts 2.0.14(就是struts修补了N个高危漏洞后的第一个版本，最常用的版本)其实是不能打的，本文会分析这个漏洞的起因，和结果，也会给出通杀 POC的思路。

　　本文希望看懂文章的人，可以专注于分析思路，但是不希望大家拿着POC到处搞站，本文不提供任何黑客工具，所有的POC，都是已经公布过，无数人都知道的。

**xwork修补漏洞的悲剧(漏洞历史)：**

　　xwork作为struts2和webwork的核心组件，曾经在已经修补过了“xwork参数拦截器允许ognl方法执行”漏洞，并给出了漏洞公告

　　http://struts.apache.org/2.x/docs/s2-003.html

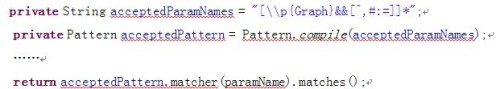
　　S2-003



**第一次修补**

　　但是这个修补，最终的结果是个悲剧。大家还记得我以前说过，在《Struts2框架安全缺陷》一文中，完全不懂web安全的开发人员，为了修补 XSS，竟然仅仅针对POC，过滤了这段字符“〈script〉”，一旦用户提交“〈script xxx〉”就绕过的傻X修补方式。

　　他们保持一贯的风格，利用正则“{\\p{Graph}&&[^,#:=]}\*”修补此漏洞。并且又仅仅针对攻击者给出的POC，做出修补，使用POC测试通过就发布了。

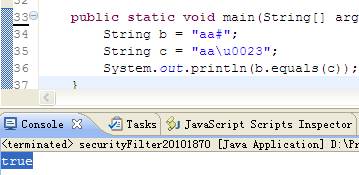


　　解析下xwork开发人员使用POC的几个悲剧的测试用例。

　　为了让ParametersInterceptor认为它是个合法ognl语句，变量中必须最终包含#。

http://inbreak.net/attachments/date_201008/struts2/TRL4SLG6B813_s.jpg

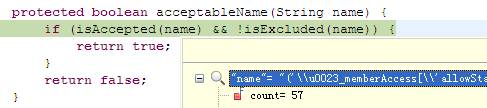
　　这行无法通过验证，java在做字符串运算时，\u0023会被转义为#之后，才会做匹配，所以返回false。下图可以看到，\u0023和#是完全相等的。



　　所以修补漏洞的开发人员，自作聪明，直接用正则把#干掉了。

　　攻击者发来的虽然是\u0023，但是这段字符在内存中做字符串运算时，会先变成\\u0023然后才做运算。java在处理用户提交的一段 string包含\时，为了保证数据完整性，会自动多加一个\用做转义，比如用户提交了数据“\n”，在内存中作字符串运算时，不会真的用换行做运算，而 是拿”\\n”这段字符做比较。那么用户提交的\u0023被转为[\\u0023](file:///\\\\)，就会绕过对#的检查。

　　证据如图：



http://inbreak.net/attachments/date_201008/struts2/V8793BLQH721_s.jpg

　　这两个case没有通过代码验证原因同上。

http://inbreak.net/attachments/date_201008/struts2/QIE27ZV347MC_s.jpg

　　悲剧就在这里，这里虽然是\\u0023，符合用户提交的场景，悲剧是这段字符里面因为有空格，就在+两边，所以无法通过，而攻击者如果去掉空格，就通过了。

　　官方就这样发布了，其实发布的是个漏洞版。

　　这是第一次修补，这次修补其实是大家都知道，因为发了公告出来，官方虽然公告说是高危，但是官方只知道攻击者可以通过ognl表达式修改 server端的session等信息。这时官方还没有意识到这其实是个远程代码执行漏洞，ognl不仅仅支持修改，还支持执行一些静态方法，比如 @Runtime@getRuntime().exec(“calc”)。

**第二次修补**

　　直到某天，可能是在版本xwork2.1.2时，官方偷偷修改安全配置，默认让SecurityMemberAccess(管理ognl权限的 类)的allowStaticMethodAccess为false，这导致静态方法不能执行，并且不知什么原因，偷偷修改正则，也同时放开了对参数名称 中空格字符的限制。

**漏洞被爆**

　　这次出了远程代码执行(Struts2/XWork < 2.2.0 Remote Command Execution Vulnerability)，漏洞的发现者就是看到了\u0023的限制其实无效，研究出了绕过默认安全配置的方法，并且利用ognl允许静态方法执行，达到了远程代码执行的效果。

**原理简介：**

　　1、 用户提交了\u0023被转义为\\u0023，通过了对参数名称的验证后，最终ognl处理之前，又变成了\u0023，也就是#，符合了ognl语法。

　　2、 通过Ognl语句执行，可以在struts2和webwork运行起来时，把ognl上下文中的一些默认配置覆盖掉，漏洞发现者给出了不少可以覆盖的数值。

　　3、 虽然默认配置是禁止静态方法执行的，但是xwork的配置，其实是可以覆盖的，一旦覆盖掉“用于禁止静态方法执行”的value，当然又可以执行了。

　　4、 Runtime.getRuntime().exec()这段，其实可以当做静态方法调用，导致执行系统命令。Ognl语句调用静态方法：@Runtime@getRuntime().exec(“calc”)。

　　漏洞发现者给出了shellcode，但是经过我的测试，shellcode并不能影响所有版本，而是仅仅针对xwork2.1.2及以上的版 本有效，xwork2.1.2及以上核心被应用在struts2和webwork某些版本中，所以他们间接受到了影响，但是修补代码，是xwork去做 的。

**漏洞发现者给出POC：**



**山寨的修补方式带来的后果**

　　我看到很多人对这个漏洞简单分析了下，有一小撮不明真相的群众认为，这是因为参数名称\\u0023带来的后果，官方自己上次修补的不完善，所以，可以过滤\\u0023，搞定这件事情。

　　如图是我做见过的“其中一个”漏洞分析者，自己搞的山寨补丁，原理就是禁止\\u0023：



　　摘自互联网某篇文章，不点名了，第二点解决方案居然不符合XML规则-\_-!。

　　这只是其中一种方案，很多人说只要禁止\\u0023就可以了，不得不说，这个人应该分析了漏洞，并且了解了原理。他知道虽然官方的补丁出来了，但是官方用的是白名单形式，对参数名称限制太严厉，很多特殊符号不能使用，可能会导致部分应用出问题，这是典型的开发人员思维。

**这是官方的补丁：**

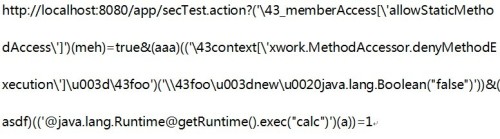
　　 http://svn.apache.org/viewvc/struts/struts2/trunk/xwork-core/src/main/java/com/opensymphony/xwork2/interceptor/ParametersInterceptor.java?view=markup&pathrev=956389

　　为了优先保证业务，只要禁止了\\u0023就可以修补漏洞，所以以上山寨方案貌似是可行的，并且经过测试POC打不了了。

　　我也看到了这个方法，这真是个悲剧，拦截\u0023能解决问题么？

　　为什么漏洞发现者，通过\u0023绕过了#限制？还有没有其他编码可以绕过？

　　答案是，还有其他编码可以绕过，仅仅控制了\u0023，是个悲剧。经过我实际测试，发现#号的8进制编码\43，也是在这里使用的，并且\043也是可以的。于是我笑了：



　　这段新的POC，没有任何一个\u0023，却一样可以执行calc，经过实际测试，绕过了所有仅仅过滤\u0023的防御。让他们慢慢修补吧，我们不着急，等大家都打上了过滤\u0023的补丁，再把这个新的POC放出来。

　　我看到neeao就很谨慎，直接上官方补丁，这是他对这个漏洞的分析：

　　http://neeao.com/archives/59/

**老版本的struts和webwork的POC不通用问题**

　　在推行修补方案时，开发人员总是从自己的角度和经验修补漏洞，他们采用过滤\u0023的方案也罢了，最起码态度端正。不像有些互联网公司的开发根本不去补，原因很简单，他用的是老版本的struts和webwork。对xwork2.1.2以下核心的struts2和webwork，POC打下去没有任何效果，所以认为这个安全级别不高，还是等等官方公告吧(现在为止，官方没有发布任何公告，修补好代码提交SVN，也没有编译后发布版本)。

　　出于好奇，决定仔细研究下。我之前也不熟悉xwork源码和ognl，以前仅仅研究过struts2的部分源码，盲目的debug了好几天，解决了N个问题，才搞定：

**空格问题：**

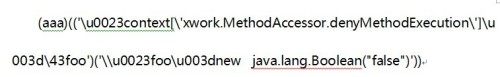
　　原POC中，会传三个参数，它们的作用，首先解析下。

　　1、第一个参数

http://inbreak.net/attachments/date_201008/struts2/11KO119RNOY7_s.jpg

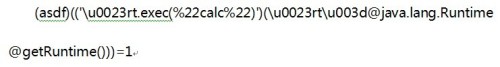
　　在高版本的struts中，allowStaticMethodAccess(允许静态方法访问访问，做权限判断)默认是false的。但是低版本的本来就是true，所以传不传都一样，可以省了。

　　2、第二个参数



　　这句必须在，否则也不会调用静态方法。

　　3、第三个参数

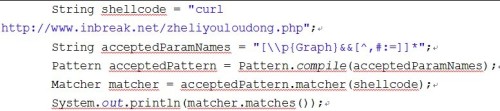


　　这句是shellcode，不能没有。

　　在老版本中，第二个参数，是不能运行的，把它弄的好看点：

http://inbreak.net/attachments/date_201008/struts2/N15LKL1G2M07_s.jpg

　　注意，new后面，有空格，在ParametersInterceptor的参数正则验证中，根本过不去。 既然老版本不允许参数中出现空格，那么如果你的shell里如果有空格，会通过么？嘿嘿。。。只要你的shell，无法通过这段验证，就不会执行：



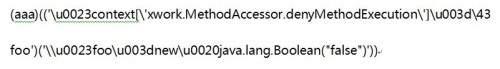
　　以上shellcode必须对空格和：符号，做16进制的转义，才能执行。

http://inbreak.net/attachments/date_201008/struts2/160P1O7KST9U_s.jpg

　　PS:大家都悄悄的，不要告诉那些“只拿poc，不看技术文章的那些不明真相群众”。

　　所以，要必须先解决的第一个问题是，改这个空格为\u0020，才能进入ognl表达式的流程。

　　在比较新一点的版本的xwork中，允许空格，当然，也是允许\u0020的，所以\u0020替换空格，就通杀了第一个问题的新老版本struts正则验证。



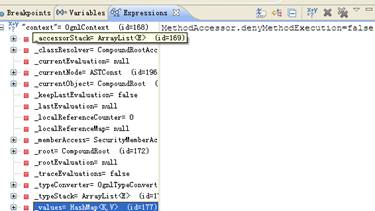
**denyMethodExecution不能修改的问题：**

　　改完之后，发现竟然还是不能赋值，经过调试，在内存中，看到的 xwork.MethodAccessor.denyMethodExecution还是true，这说明这个表达式，没有执行成功。原因是这里做了 new对象操作。先定义了#foo变量为new java.lang.Boolean类型，默认为false，之后denyMethodExecution等于#foo。这是不允许的，原来的POC导致 空指针异常(原因后面说)，后来解决了。

　　总之用这个，可以通杀新老版本，也不会爆空指针：

http://inbreak.net/attachments/date_201008/struts2/J7K64FA569BD_s.jpg

　　提交后再次查看内存中的context，发现这个值被修改为false。



　　看看shellcode

http://inbreak.net/attachments/date_201008/struts2/14TX26C392C1_s.jpg

　　Shellcode的原理是，利用ognl支持静态方法执行，调用java的执行系统命令方法(其实完全可以调用任何java代码，比如写个文件等)。

http://inbreak.net/attachments/date_201008/struts2/AA19G8J00RYT_s.jpg

　　Shellcode是可以做new操作的。

　　denyMethodExecution不能new操作，是因为这句执行时的上下文中， denyMethodExecution还是true，执行了这句，才是false，这时才可以new对象。

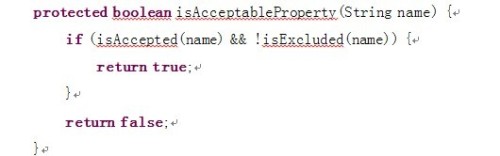
　　所以shellcode的上下文，是可以做new对象操作的。

**Xwork的bug问题：**

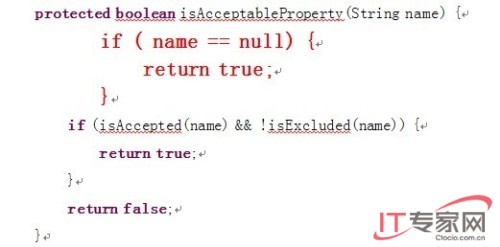
　　解决了这两个问题，其实已经给shellcode创造了完备的环境，按照xwork的逻辑，应该直接让我们调用静态方法才是，但是在shellcode运行时，居然爆出了空指针，这个问题我研究了好久才搞明白，原来是xwork自己出了bug，到了新版本时，才修补。

　　翻翻svn，看到在xwork2.1.2时，偷偷修改了一段代码。

　　SecurityMemberAccess这个类原来有个这样的方法



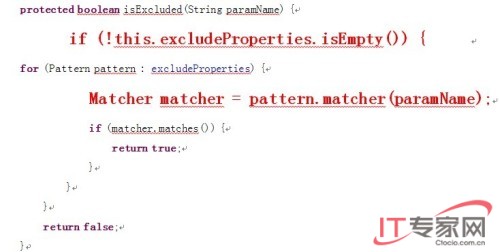
　　到了xwork2.1.2时，改为了：



　　注意标红的，如果name==null，就返回true。

　　为什么会有这行代码呢？

　　它调用isExcluded(name)，进入isExcluded方法后，做正则表达式的验证。



　　如果传进来的是个paramName是null，并且excludeProperties是有值的，必然报错。Xwork的bug就是，所有的版本，调用静态方法时，都必然会传进来一个null，并且excludeProperties也是默认有值的。

http://inbreak.net/attachments/date_201008/struts2/X545O9368V0P_s.jpg

　　任何一个静态方法的调用，这里传进来的都是null。

　　也就是说，要调用静态方法，就必须让excludeProperties这个家伙的值，是一个空的Set对象，否则对null对正则匹配，就会报错。

　　excludeProperties是个Set，在shell执行的上下文中，它的值是这样来的：

　　xwork处理用户发进来的ognl表达式时，会用xwork的SecurityMemberAccess做权限判断，以保证静态方法不会被人 随便调用。原理是传给shell执行的上下文中几个默认配置，其中一个是默认的allowStaticMethodAccess=true，也会给 excludeProperties这个Set对象会被添加一个value，结果就不是空的Set对象了。

　　所以，在shellcode上下文中，执行静态方法必然会出错。

　　让shellcode正常执行，解决方法的思路就是new一个新的HashSet。

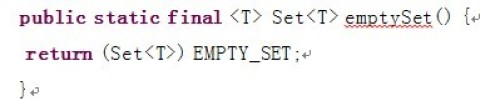
http://inbreak.net/attachments/date_201008/struts2/0ME11O681V77_s.jpg

　　但是经过我测试，这里还是不能做new的操作，因为new的时候，会调用构造方法，这实质上还是在调用方法，调用方法就会出错。为了证明我的猜 测，debug调用到这一步的时候，手工修改它为new HashSet()，立刻就通过了。不能new，又怎么能得到一个空的Set呢？这是最后一道关卡。为了搞定它，我甚至查了struts- default.xml在内存中的位置，考虑是不是使用ognl修改掉。

　　因为这个放弃了一段时间，很郁闷，当我再次拾起来，却突然看到了这个值的默认值。

http://inbreak.net/attachments/date_201008/struts2/9D45OM0J9EX7_s.jpg

　　跟进emptySet();



　　这是个有意思的知识点，我也是第一次了解到，可以使用EMPTY\_SET，取到一个空的set。

　　于是当我提交某些东西，让

http://inbreak.net/attachments/date_201008/struts2/C5G9D8ATVBU0_s.jpg

　　这一句覆盖了默认的那个有了一个值的map，重新变成了空的map，这才真正的达到了静态方法的终极调用条件。

　　其实我都快吐血了，明明是xwork自己代码写的不严谨，直到他们新的版本才发现这个bug，并修补，居然导致我研究漏洞的同时还顺带帮他们处理bug。

　　最后终于找到机会鄙视一下漏洞发现者的poc，他说老版本的struts2可以使用XXX的POC，经过我测试，不可行，原理也很简单，本文说了，POC中不能有空格。

　　这句话即使在新版本中，也是没什么影响的，虽然会改变一个值，但是不受啥影响， shellcode中，有了这句话，就兼容新老版本，一切通杀了!

**最后凑出通杀0DAY在这里：**

**被和谐了，没办法，毕竟在互联网公司里，要考虑到其他互联网公司同行的处境，虽然他们不一定都是漏洞公布的POC不能打的版本。**

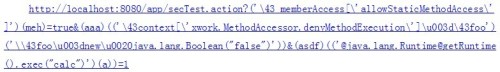
　　POC，不要找我要了，本文仅仅说下技术研究。

**回顾下：**

　　适合新版本绕过双引号和\u0023防御的poc：

http://inbreak.net/attachments/date_201008/struts2/A6R1D8574YT2_s.jpg

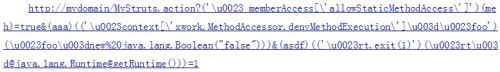
　　适合新版本绕过\u0023的poc



　　仅仅适用老版本(struts2.0.12)的POC

http://inbreak.net/attachments/date_201008/struts2/5C82PF577289_s.jpg

　　仅仅适用新版本的也就是漏洞发现者放出来的poc



[Java Security](http://www.inbreak.net/archives/category/%E5%8E%9F%E5%88%9B%E6%96%87%E7%AB%A0/java-security), [原创文章](http://www.inbreak.net/archives/category/%E5%8E%9F%E5%88%9B%E6%96%87%E7%AB%A0)[struts2](http://www.inbreak.net/archives/tag/struts2), [webwork](http://www.inbreak.net/archives/tag/webwork), [xwork](http://www.inbreak.net/archives/tag/xwork), [框架安全](http://www.inbreak.net/archives/tag/%E6%A1%86%E6%9E%B6%E5%AE%89%E5%85%A8)

[← 安全系统达人秀](http://www.inbreak.net/archives/166)

[Tomcat远程拒绝服务漏洞分析(CVE-2010-2227) →](http://www.inbreak.net/archives/168)

[发表评论？](http://www.inbreak.net/archives/167#respond)

7 条评论。

1. **axis** [2010 年 08 月 06 日 在 上午 11:30](http://www.inbreak.net/archives/167#comment-340)

有技术研究的风范了，哈哈

[登录以回复](http://www.inbreak.net/wp-login.php?redirect_to=http%3A%2F%2Fwww.inbreak.net%2Farchives%2F167)

1. **nuke** [2010 年 08 月 06 日 在 下午 3:32](http://www.inbreak.net/archives/167#comment-341)

确实好文 ，读得畅快

[登录以回复](http://www.inbreak.net/wp-login.php?redirect_to=http%3A%2F%2Fwww.inbreak.net%2Farchives%2F167)

1. **wofeiwo** [2010 年 08 月 06 日 在 下午 4:51](http://www.inbreak.net/archives/167#comment-342)

写得好！每一点都说的透彻。  
struts开发者们要是也像你这么一一分析，他们的代码就不会这么悲剧了

[登录以回复](http://www.inbreak.net/wp-login.php?redirect_to=http%3A%2F%2Fwww.inbreak.net%2Farchives%2F167)

1. **天津西站医院** [2010 年 08 月 10 日 在 上午 9:02](http://www.inbreak.net/archives/167#comment-343)

好文章 支持

[http://www.00857.cn](http://www.00857.cn/)

[登录以回复](http://www.inbreak.net/wp-login.php?redirect_to=http%3A%2F%2Fwww.inbreak.net%2Farchives%2F167)

1. **ssssssss** [2010 年 11 月 24 日 在 下午 3:16](http://www.inbreak.net/archives/167#comment-356)

1.0的行不???????????

[登录以回复](http://www.inbreak.net/wp-login.php?redirect_to=http%3A%2F%2Fwww.inbreak.net%2Farchives%2F167)

1. **kxlzx** [2010 年 11 月 24 日 在 下午 11:11](http://www.inbreak.net/archives/167#comment-357)

1.0的时候，struts还没有任何与xwork接触的地方。所以当然不行了。