By [kxlzx](http://secinn.appspot.com/pstzine/read?issue=4&articleid=3)

**摘要**

本文介绍了java开发流行框架struts2以及webwork的一些安全缺陷，并举例说明框架本身以及开发人员使用框架时，所产生的种种安全问题，以及作者挖掘框架安全漏洞的一些心得体会。

**推荐以下人群阅读**

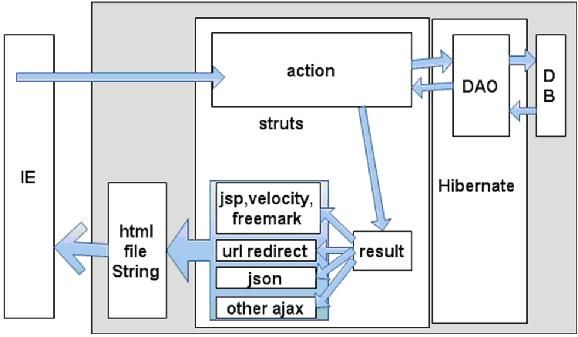
了解java开发  
了解框架开发  
了解web application安全  
“网络安全爱好者”

**正文**

当前java开发网站，通常不会是纯JSP的，大都使用了java framework。

有了这些framework，让开发人员更加快速的开发出代码，也让代码非常具有可扩展性，那些分层架构的思想，更是深入人心。这些也大大影响了安全代码审核，曾提出“分层审核代码”的思想，比如在DAO层专门检查sql注入，在view层检查xss等。这些框架都有自己的层级，本次文章主要讲的是struts这个框架的相关安全问题，也会有小部分涉及到struts后面的DAO层。

而struts这个框架更新占有市场份额极大的一个框架，它在各个层级中，位于如图所示位置：



可以看到struts在web应用中，负责处理接收用户数据，调用业务处理，以及展示数据的工作。所以本文把struts的功能分为controller层和view层，controller层来完成接收用户数据，分发用户请求，而view专门用于展示数据。

一个单独的struts，是不合逻辑的，因为架构师通常喜欢多种框架集合，让它们各自负责某一层的处理。研究一个框架的安全问题，不能仅仅站在框架的角度，还应该充分考虑到开发人员是如何使用这些框架的，他们最喜欢写什么样的代码，这样才能还原一个正常的、完整的web应用场景。

从搜索结果看，互联网中，绝大多数教程推荐struts+hibernate+spring这样的黄金组合，那么，我假设有一个应用使用了这个组合，以struts为重点，站在攻击者的角度，层层分析struts的设计缺陷。

**Struts2开发回顾与简单学习**

为了让大家回顾或者学习一下struts2，我们一起来建立一个action、jsp页面，做一个接收用户输入，之后处理一下，再展示出来给用户的过程，精通struts2的同学可以跳过此步。

————————————-struts回顾start  
首先建立action，叫做AaaaAction：

public class AaaaAction extends ActionSupport{

private String name;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String execute(){

System.out.println("exe");

return SUCCESS;

}

public String bbb(){

System.out.println("bbbbb");

return SUCCESS;

}

}

请注意execute这个方法，让用户输入action的地址后，默认会访问这个方法。

之后配置struts.xml文件

<action name="aaaaaaa">

<result name="success">user/aaa.jsp</result>

</action>

配置这个文件后，当用户输入

http://www.inbreak.net/app/aaaaaaa.action

的时候，struts会负责让AaaaAction中的execute方法处理用户请求。

处理之后，该方法返回“return SUCCESS;”，struts又负责找到result的name是seccuess所指向的jsp页面。  
把该页面解析后，返回给用户。

而用户看到的就是aaa.jsp页面的html代码。

struts2继承了webwork的所有优点，其实等于是webwork的升级，如果开发人员想让用户直接访问action中的某方法，而不是访问默认的execute方法，只要定义一个方法叫做bbb，并且是public的，用户就可以直接输入

http://www.inbreak.net/app/aaaaaaa!bbb.action

直接访问了bbb方法。

那request中的参数如果接收呢？struts2中，这个过程被包装了起来，使用非常方便，只要在action中定义一个属性，叫做public String name；。然后加入getName和setName方法，就可以像正常使用属性一样，接收到用户传递过来的变量。无论是get请求还是post请求，都可以使用这种方式接收用户输入。

整个过程就如此简单，现在大家对流程有了了解，我们就开始讨论正文，如果还是想了解更多，请自行google。

———————————-struts回顾end

**Struts2安全缺陷**

可以看到struts2在数据流向方面，有两个重点，一个是进入（in），一个是输出（out）。而我在做漏洞挖掘的思路，也是跟着这个数据的流程，开始分析的，下面我们就开始让数据进入。

Action属性默认值可以被覆盖缺陷：

在日常的java项目中，我们经常会遇到保存一个新的对象（比如注册一个用户），然后给这个对象赋予一些用户提交上来的属性值，在这里，只需要定义一个对象类：

public class User {

private Long id=0l;

private String name;

private String pass;

private Integer type=1;

。。。下面的get和set方法代码略

}

定义后，在action中，添加一个属性

User reguser;

用户注册的页面代码如下：

<form XXXXXXX>

<input name="reguser.name">

当用户提交这个form到action中后，struts2会负责自动映射reguser.name的值到reguser的相关属性（name）中，所以在execute这个方法中，就可以使用reguser.getName()拿到用户提交的reguser.name的值。所以我们下面的代码就很简单了：

public String execute(){

add(user);

add方法，更简单了，因为我们项目中集成了hibernate，这个框架自动映射user类中的各个属性，自动组成insert语句。我们只要在add中调用session.save(user);就可以保存用户到数据库中。

前文提到那么多“简单”两个字，难道这些过程都是安全的而他给我们仅仅带来了方便么？

struts2只负责映射所有对象，他提供了form验证，也只能验证form中属性值的内容，比如email格式等，并不能约束用户提交其他属性上来，于是这就变成了十分危险的功能。

当User中有个属性type，代表User是否管理员时（1为普通用户，2为管理员），麻烦来了，攻击者在原来的注册表单中，新加入一个input，叫做

<input name="reguser.type">

然后输入值是2，把这个值一起交给action。在这个流程中，这个值，当然也会被自动带到数据库中，向下处理的逻辑中，这个用户，就已经变成管理员了。

当你看到了一个struts2或者webwork的应用，可以尝试使用属性攻击，修改当前表单，里面有所有你猜测到的属性，一并提交上来，就可能会影响整个逻辑，达到攻击目的。文中仅仅是一个例子，事实上，在数据传递的过程中，可以任意覆盖数据的默认值，本来就是一个危险的缺陷，而struts2和webwork这两个框架仅仅看到了它带来的好处，忽略了这方面基于安全性的考虑，仅仅关注了用户提交数据的正确性。对比在没有struts2这个功能的时候，我们却需要在action中一个一个的把需要的变量，从用户提交的request中解出来，一个一个处理，不可能出现这种安全问题。现在它包装了这个过程，自以为很方面，却出了严重问题。

**Action中的方法被暴力猜解缺陷**

前文提到，有一种方法可以让用户访问action时，不访问默认的execute方法，而是直接访问其他action中的方法，条件是在action中，写一个public的方法。开发人员如果需要做一个登陆后，展示所有用户列表的功能，而他的一个“解耦合”的开发习惯，将在这里导致安全缺陷。

定义一个如下的action

public class Userlogin extends ActionSupport{

private String uname="";

private String upwd;

private List list;

//getter and setter 方法略

public String login(){

if(uname!=null&&upwd!=null&&uname.equals("kxlzx")&&upwd.equals("pass"))

{//if login success

return list();

}

return false;

}

public String list(){

list.add("kxlzx");list.add("kxlzx1");list.add("kxlzx2");list.add("kxlzx3");

return "list";

}

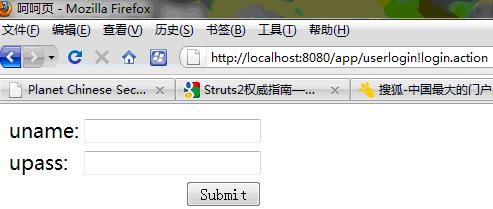
}

Userlogin中，因为list这个功能（显示所有用户列表），其实是一个通用的功能，很容易被其他地方调用，所以开发人员把它单独写成了一个方法。

当用户登陆的时候，打开

http://www.inbreak.net/app/userlogin!login.action

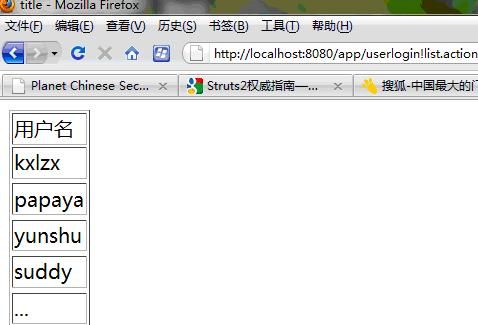
来到了用户的登陆页面，可以看到，只有用户输入正确的用户名和密码，才能最终调用list()方法，显示结果。



但是struts2把所有public的方法都暴露了出去，导致现在用户输入了

http://www.inbreak.net/app/userlogin!list.action

用户访问这个链接后，struts2调用list方法，然后返回结果给用户，所以没有登陆，就显示了所有用户信息，  
直接绕过了login中的登陆验证。



在没有struts2的时候，我们要在servlet的doget或者dopost方法中，写if判断等代码，才能让用户调用其他servlet中的方法，现在看来其实这也是一种保护措施。而现在struts2为了方便开发，把所有的public方法统一映射了出去，导致开发把一个经常使用的功能，习惯写成一个public的方法，现在居然成了严重漏洞。

**struts2的action属性设计缺陷**

再回头看看我们在action中的属性定义，你会发现，现在他们都成了漏洞，因为struts2规定属性的get和set方法，都必须是public的。

那么我们定义了

private String name;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

这段代码的时候，实际上，是写了两个public的方法。  
那这两个表面上没有任何实质含义的方法，会有什么安全隐患呢？  
这需要和前文联系起来，前文提到，我们在struts.xml文件中，定义如下：

<action name="user">

<result name="success">user/userlist.jsp</result>

<result name="addUser">user/addUser.jsp</result>

<result name="added">user/added.jsp</result>

<result name="false">user/false.jsp</result>

</action>

这段代码含义是，UserAction中，任何一个方法执行后，如果返回的是success这个字符串， 就会把user/userlist.jsp返回给用户。

如果返回是addUser，就会把user/addUser.jsp返回给用户。  
现在UserAction是管理用户的页面，在我们的系统中，有普通管理员和超级管理员，他们的区别是普通管理员可以查看用户，但是不能添加一个用户。

所以，我们在UserAction中，写了

public String addUser(){

if(true){ //事实上这里是个超级管理员的判断，我偷懒了。

return "false";

}

return "addUser";

}

这个方法的代码判断了不允许普通管理员访问，但是user/addUser.jsp这个jsp页面中并没有这个判断逻辑。  
因为开发认为只有返回addUser的时候，才会来到这个页面，而要返回addUser，则必须通过超级管理员的验证。

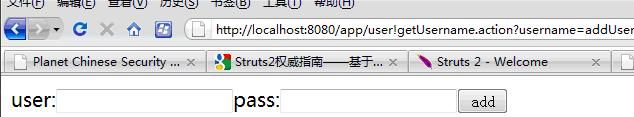
那我们能让一个方法返回addUser么？当然可以！

http://www.inbreak.net/app/user!getUsername.action?username=addUser

这个链接，struts2会怎么处理呢？

他会找struts.xml中，对应段路径user，于是找到了对应的处理Action（net.inbreak.UserAction），由于路径中有了“!getUsername”，于是就去找这个Action中的getUsername这个方法，很明显，这个方法其实是username这个属性的get方法，如果你要让Action接收用户提交的username，你就必须要定义这个方法。

那这个方法会返回什么呢？会返回action的字段username的值！哈哈！username用户已经提交给action了，链接后面写着“?username=addUser”，struts2把这个值赋予了action中的username属性。那这里返回的当然就是“addUser”！



一系列巧合后，导致现在给用户返回了user/addUser.jsp页面，这是一个添加用户的表单页面，并且用户没有去走验证是否为超级管理员这一步。

现在用户看到了一个添加用户的页面，他有两种攻击思路：  
1，直接提交，如果处理用户提交的那个action没有再次判断用户身份，那就提交成功了。  
2，如果他判断了用户身份，我们还可以csrf他，因为我们知道了这个action的地址，和它需要的参数！

由于struts2的action和jsp文件分离，导致开发人员往往会在action的方法中，执行权限判断，而jsp页面中并没有再次执行这个判断，他以为action判断就够了。而偏偏action的属性，给我们带来了一个可自定义返回result的方法，导致我们可以绕过action访问jsp页面。

**Struts2的那些result类型缺陷（redirect）**

刚才我们领教了struts2给我们带来那些属性的好处，现在我们再往后走一步，研究Action方法的返回结果。  
其实并不是只由String类型的返回结果，struts2还有其他类型的返回，比如“redirect”类型。

<action name="test">

<result name="false">user/false.jsp</result>

<result name="redir" type="redirect">${redirecturl}</result>

</action>

这段代码，大家唯一可能看不懂的，就是type=”redirect”了。

这是一个url redirect的方式，struts2为了方便大家开发，把“自定义302跳转到其他url”这种方式给包装了起来。只要如上定义，我们就可以在action中写方法：

public String redirect() {

return "redir";

}

然后定义属性

private String redirecturl;

当用户打开

http://www.inbreak.net/app/test!redirect.action?redirecturl=/a.jsp

的时候，就会302跳转到

http://www.inbreak.net/app/a.jsp

这是很常见的url跳转应用，在struts2中，如上配置一下，就可以实现。

相信明眼人都看出来了，很明显这里存在url跳转漏洞，如果用户输入了

http://www.inbreak.net/app/test!redirect.action?redirecturl=http://www.ph4nt0m.org

就会跳转到http://www.ph4nt0m.org这个钓鱼网站(-\_-!)。那么如何防御呢？

要防御url跳转到钓鱼网站，我们肯定需要一个白名单机制，或者根本就让他跳转到本站下。于是有了如下判断：

public String redirect() {

if(redirecturl.startsWith("/"))

{

return "redir";

}

return "false";

}

可能你看出来了，仅仅判断”/”开头，其实是不能杜绝url跳转漏洞的，因为

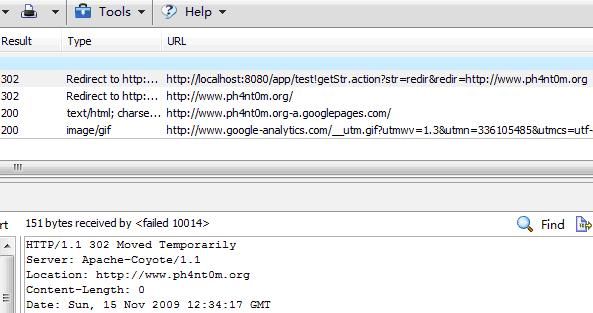
http://www.inbreak.net/app/test!redirect.action?redirecturl=//www.ph4nt0m.org

一样会跳转。而在这里却足够了，因为struts2已经接管了这个过程，只要以“/”开头，统统先给你自动加上本地域名，抓包后，你会看到  
location: http://www.inbreak.net/app//www.ph4nt0m.org

实际上是不会有问题的。

struts2也认为这样判断不会有问题了，然而用户输入

http://www.inbreak.net/app/test!getStr.action?str=redir&redirecturl=http://www.ph4nt0m.org



其实前篇已经分析过了，这样就利用action中的str属性，绕过了必须以“/”开头的判断，直接跳转了。

test里有个str属性，可自定义返回，这里自定义了“redir”，所以来到了

<result name="redir" type="redirect">${redirecturl}</result>

而redirecturl的值，也提交给了action，所以跳转了。

**Struts2的那些result类型缺陷（Ajax）**

在struts2中使用ajax，也是被struts2支持的，它提供了一种返回类型，叫做“stream”。在研究这个result的使用时，作者看到一本书，叫做《 Struts 2权威指南：基于WebWork核心的MVC开发 》。这本书非常出名，几乎所有的struts2使用者都推荐使用。

http://book.csdn.net/bookfiles/479/index.html

书上介绍ajax可以这么使用：

配置struts.xml

<action name="ajaxtest">

<result type="stream">

<param name="contentType">text/html</param>

<param name="inputName">input</param>

</result>

</action>

之后写TestajaxAction：

public InputStream input;

public String execute() throws Exception{

input = new StringBufferInputStream("aaaa<td><script>alert("kxlzx")</script>aa");

return SUCCESS;

}

其实大家都看出来我的意思了，返回了contentType为“text/html”的页面，内容为

aaaa<td><script>alert('kxlzx')</script>aa

结果浏览器解析的时候，出现了XSS漏洞。

本来默认的contentType是text/plain，不需要配置，如果用户直接打开，只会看到一个Stream，不会解析其中的html和js。现在书上介绍说要写成这样，不知道作者是否知道这个教程对大家的影响，结果已经误导了大批的开发人员。

事实上，这不是struts的问题，是struts“权威”教程的问题。权威的教程，一旦出现安全漏洞，往往会误导大批的开发人员，不知道大家在挖漏洞的时候，是否注意到了这点，特别是当官方的DEMO出现漏洞，那绝对是惊天地泣鬼神的悲剧。

**Struts2的那些result类型缺陷（自定的页面）**

有时候，开发人员为了方便，喜欢配置struts.xml如下：

<action name="test">

<result name="success">user/test.jsp</result>

<result name="testpro">user/testproperty.jsp</result>

<result name="redir" type="redirect">${redir}</result>

<result name="testloadfilepath">${testloadfilepath}</result>

<result name="false">user/redirfalse.jsp</result>

<result name="input">user/input.jsp</result>

</action>

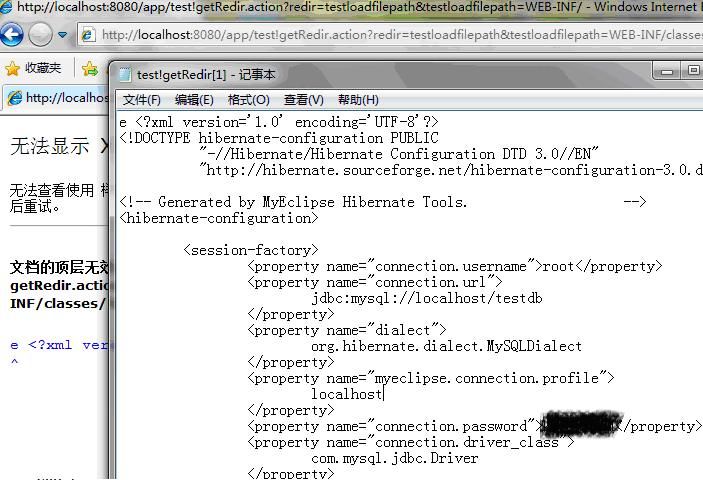
请注意，其中一条result，名称是”testloadfilepath”， ${testloadfilepath}的作用是自定义的jsp页面地址，接收session或request中传过来的这个变量的值。那么用户提交

http://www.inbreak.net/app/test.action?testloadfilepath=user/test.jsp

当然就会返回user/test.jsp页面，非常的灵活。虽然并不是所有的开发都会这么做，但是一旦出现这种情况，会产生什么问题呢？

http://www.inbreak.net/app/test!getRedir.action?redir=testloadfilepath&testloadfilepath=WEB-INF/classes/hibernate.cfg.xml

不知道大家看懂这段url的含义没有，先调用getRedir，可以自定义返回到testloadfilepath，而testloadfilepath已经指定了WEB-INF/classes/hibernate.cfg.xml。WEB-INF目录下，都是受web容器保护的东西，默认不允许直接request相对地址来访问。该目录里面有程序编译后的class文件（可以被直接反编译为java源码），有数据库配置文件等敏感文件，现在打开如上url，直接被下载了hibernate.cfg.xml，这里放着数据库用户名和密码。



这样，攻击者就可以下载你的所有源代码，所有服务器上的文件。struts在提供给我们这种方式的时候，并没有任何官方说明这里有危险，这就是一个不定时炸弹。

**Struts2的taglib设计缺陷**

经过几个例子下来，不知道大家注意到没有，从用户输入走到这里，已经走到了输出这一步了。struts2的那些result的type，其实就是几种输出方式，有jsp、ajax、redirect，经过jsonplugin等插件配置，还可以支持其他输出方式。甚至支持一些标签库，比如freemarker等标签库。不过我们只谈struts2自带的标签库，在一个jsp页面的最上方，写上一段代码，就可以使用struts2提供的数据输出和页面数据操作的标签了。  
比以往我们在jsp输出“<%=name%>”要方便的多，下面给个例子：  
test.jsp代码

<%@ taglib prefix="s" uri="/struts-tags" %>

<s:property value="username"/>

第一行是告诉struts这里要使用struts的标签库，第二行就是一个标签的使用，含义是输出username的值，username会从session、request、page等地方取，这里不关注取数据的次序。

**struts2的taglib设计缺陷（struts2.0不支持escapeJavaScript）**

说到输出，大家都能想到XSS漏洞，那么作为一个流行框架，struts2在这里做了什么控制呢？  
struts2.0对部分标签做了默认的htmlescape：

刚才那个标签实际上效果等于

<s:property value="username" escape="true"/>

别以为做了htmlescape就够了，输出在javascript中的时候，还会出现xss漏洞。所以struts在2.1.6这个版本也支持了javascriptescape：

struts2.1.6：

<s:property value="pass" escape="true" escapeJavaScript="false"/>

默认开启如上所示，当你要输出到js中的时候，可以使用escapeJavaScript进行转义。

也就是说，一旦你确定这个struts是2.0的，只要开发人员把变量输出到js中，十有八九会出xss问题。

**struts2的taglib设计缺陷（没有富文本安全输出标签）**

而包括最高版本2.1.8在内，仍然没有支持富文本安全输出，这是一件悲剧的事情，如果用struts开发一个大众blog的应用，又支持富文本的文章，开发人员只能把htmlescape和jsescape都去掉，才能保证业务正常运行，所以导致了XSS漏洞。

**struts2的taglib设计缺陷（并不是所有输出标签都做了默认的htmlescape）**

有几个标签是不做htmlescape的，比如

<s:a>

<s:text>

<s:url>

这其实是一个严重陷阱，因为只要提到struts2，前辈们都会告诉你，放心使用，它默认做了htmlescape。那是什么原因导致一些标签没有做默认的escape呢？作者翻了下源码，也没有找出具体原因，不知道那些人是怎么想的。

并且，经过简单的fuzz，发现在特定环境下，那些做了输出转义的标签也会出现问题。

我们知道默认的htmlescape是不转义单引号的，所以，当struts标签库的源码中，出现一些标签属性的输出时，如果标签属性的周围使用的是单引号，而攻击者又能控制标签属性内容的时候，就会出现xss漏洞。如下：

<input name="username" onclick='xss'>

当这个xss的内容可以由攻击者控制，即使对xss的内容作了htmlescape，依然可以被攻击者bypass。

基于这个原理，作者搜索了struts标签库源码，那些“XXX.ftl”文件中搜索“}’”符号，找到N多，测试其中一个如下：

————-  
<s:textfield >标签，在正常使用的时候，他会放到一个<s:form>标签内，最终输出html后，会变成一个输入框。  
它有个属性叫“tooltip”，如果这个标签为用户可控制，比如从数据库中读取用户输入，而这个标签所在的  
<s:form>开启了：

<s:form tooltipConfig="#{'jsTooltipEnabled':'true'}">

的时候，用户输入的tooltip的值，会出现以下情况：

struts2.0 –>

<span dojoType="tooltip" ... caption="kxlzx<script></script>">

caption内容就是tooltip的值，从数据库查出

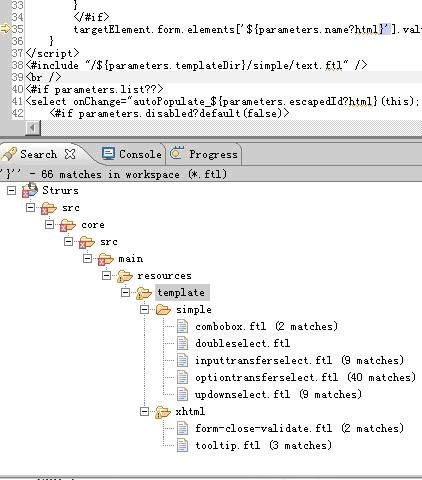
struts2.1.6&struts2.1.8 –>

<img onmouseover="domTT\_activate(this, …'StrutsTTClassic');alert('xss');a('','styleClass’…" />

onmouseover生成一个domTT\_activate函数调用，参数中其中一个值，是tooltip的内容。这里被bypass了。

————

这些搜出的几个个地方实际上根本没有做任何escape，就直接输出了数据。下面那个即使做了默认的htmlescape，还是会出问题，除非它默认做了javascriptEscape。struts2默认有地方做javascriptEscape么？  
答案是“没有”。所以，它们全都能被XSS！



struts2的这些escape，其实是一个很太监的安全方案，安全工程师最恨的就是这种方案，做了安全方案，还不做完全，留下一堆问题。

**struts2的HTTP Parameter Pollution处理缺陷**

webwork和struts2都有这个问题，当用户给web应用提交:

http://www.inbreak.net/app/test!redirect.action?redir=kxlzx&redir=aaad61

时，如果我们在action中定义了

private String redir;

public String getRedir() {

return redir;

}

public void setRedir(String redir) {

this.redir = redir;

}

Action就会取到redir的值为“kxlzx, aaad61”注意中间是有空格的。

这种数据是由webwork（struts2）把两个参数合并而成的，但是如果我们request.getParameter(“redir”);拿到的值，却只是第一个（值为kxlzx）。

我们知道struts2提倡使用拦截器做一些事情，他可以在action的execute方法执行之前和之后做一些操作。那就有一些开发，想当然的在这里防御一下url跳转、SQL注入、XSS等攻击。我们看看他们会怎么做：

@Override

public String intercept(ActionInvocation arg0) throws Exception {

……

String name = request.getParameter("name");

if(name!=null&&name.indexOf("'")>-1){

System.out.println("find sql injection");

request.getSession().setAttribute("msg", "find sql injection");

return "falseuser";

}

String redir = request.getParameter("redir");

if(redir!=null&&!redir.equals("http://www.b.com")){

System.out.println("find url redirect");

request.getSession().setAttribute("msg", "find url redirect");

return "falseuser";

}

return arg0.invoke();

}

在这段代码中，作者仅仅示例了在拦截器中防御sql注入和url跳转漏洞，sql注入的防御规则是检查“’”单引号，而url跳转漏洞规则是检查必须跳转到”http://www.b.com”去。作者知道没有完全防御，所以大家先不要在这里追究防御方案，仅仅是一个示例。

而开发人员在业务代码如下：

String sql = "select book\_name,book\_content from books";

if (name != null) {

sql += " where book\_name like '%" + name + "%'";

}

很明显能注入。

public String redirect() {

return "redir";

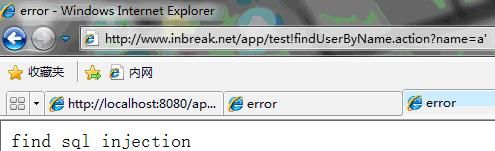
}

也明显存在url跳转漏洞。

但是由于拦截器在action之前执行，所以如果我们输入了

http://www.inbreak.net/app/test!findUserByName.action?name=a’

拦截器当然就会返回错误“find sql injection”；



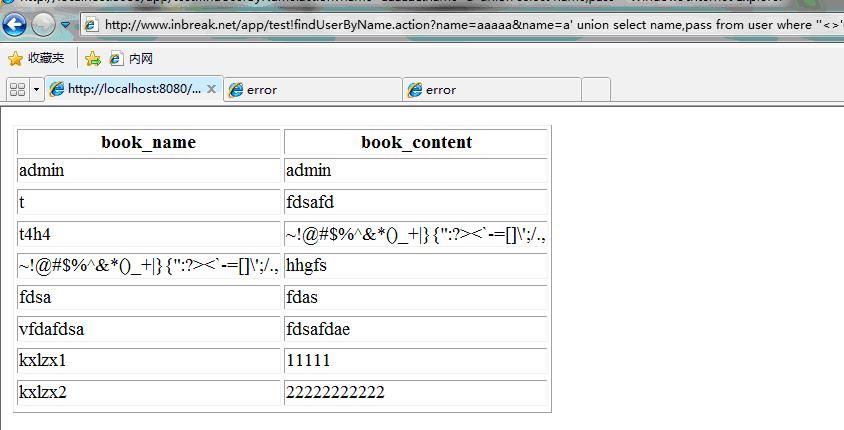
因为执行到了

String name = request.getParameter("name");

if(name!=null&&name.indexOf("'")>-1){

发现name的值确实有单引号。  
但是如果我们输入了

http://www.inbreak.net/app/test!findUserByName.action  
?name=aaaaa&name=a’ union select name,pass from user where ”<>’

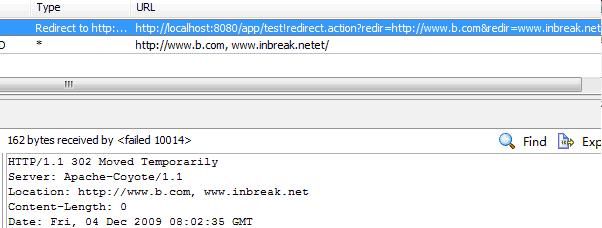


就直接绕过了拦截器的判断。因为拦截器获取的request.getParameter(“name”)，是第一个参数的值aaaaa，抛弃了第二个参数的值，但是action中的name的值，却是“aaaaa, a’ union select name,pass from user where ”<>’ ”所以被注入了

大多数拦截器都是这样做的防御，包括一些filter等。  
这件事情发生在url跳转漏洞时，却不明显，因为攻击者顶多构造一个：

http://www.inbreak.net/app/test!redirect.action?redir=http://www.b.com&redir=www.inbreak.net

抓包看看



它跳到了http://www.b.com, www.inbreak.net去了。所以IE直接报错，说打不开这个地址。但是我们还有别的浏览器，总是喜欢给大家友好信息的浏览器，看看chrome给用户什么提示：



Chrome也认为这是一个错误的链接，所以给出了“正确”的链接地址。这不是刚好被钓鱼网站利用么？  
struts2的官方漏洞公告和修补后引发的安全缺陷

从有struts2，到现在为止，官方一共发布了4个漏洞，在

http://struts.apache.org/2.x/docs/security-bulletins.html  
\* S2-001 — Remote code exploit on form validation error  
\* S2-002 — Cross site scripting (XSS) vulnerability on <s:url> and <s:a> tags  
\* S2-003 — XWork ParameterInterceptors bypass allows OGNL statement execution  
\* S2-004 — Directory traversal vulnerability while serving static content

从名字上，可以看出漏洞的内容，作者仅仅对其中两个做了源码级别的漏洞修补评估，发现了很多悲剧的事情。  
同学们有兴趣可以去研究剩下两个漏洞。

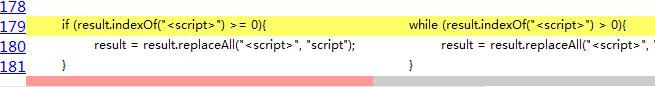
**struts2的官方漏洞公告和修补后引发的安全缺陷（S2-002）**

先看看“S2-002 — Cross site scripting (XSS) vulnerability on <s:url> and tags”这个漏洞。

顾名思义是对<s:url>和<s:url>的xss漏洞修补，但是前文提到，这里有XSS漏洞，难道是在忽悠大家？我们看看这帮工程师是怎么修补的，来到这个svn地址：

http://svn.apache.org/viewvc/struts/struts2/trunk/core/src/main/java/org/apache/struts2/views/util/UrlHelper.java?r1=614814&r2=615103&diff\_format=h

注意这两行：



看到这两行代码的时候，作者笑了，因为作者仿佛看到了至少两件悲剧的事情，现在把它们写成故事：  
第1件悲剧的事情，某年某月某日，一个脚本小子给官方报告漏洞，说在使用<s:url>标签的时候，代码为：

<s:url action="%{#parameters.url}"></s:url>

之后他输入了

http://www.inbreak.net/app/test!testpro.action?url=<script>alert(‘hacked by kxlzx’)</script>

并告诉官方这里是一个XSS漏洞，希望官方修补掉。  
官方很重视，一个开发就去修补，添加如下判断：

if (result.indexOf("<script>") >= 0){

result = result.replaceAll("<script>", "script");

并进行了冒烟测试、功能测试、黑盒测试、白盒测试。认为没有问题了，因为提交攻击者给的恶意url后，输出了

scriptalert('hacked by kxlzx')</script>

结果并没有在页面执行xss脚本。后来那脚本小子也测试了一下，发现没问题，这事情就过去了，瞒着人民大众，悄悄的修补了。

第2件悲剧的事情，又过了某人某月某日，某另一个脚本小子又发了漏洞，还是那段代码，但是url改成了：

http://www.inbreak.net/app/test!testpro.action?url=<<script>>alert(‘hacked by kxlzx’)</script>

注意，这里是<<script>>，经过了replaceAll函数后，刚好变成了<script>，重新组成了XSS漏洞。  
官方这次不得不重新重视起来，决定把if判断，变成了while，不管你有多少<<<<<<<script>>>>>>>，都给你变成

scriptalert('hacked by kxlzx')</script>

并进行了冒烟测试、功能测试、黑盒测试、白盒测试。这次还发了公告出来，说这里没问题了，我们很重视安全漏洞，已经修补了。

作者看到这里，测试新的bypass官方修补代码的url为：

http://www.inbreak.net/app/test!testpro.action?url=<script kxlzx=kxlzx>alert(‘hacked by kxlzx’)</script>

于是XSS脚本又被执行了，因为这里是<script kxlzx=kxlzx>，不是<script>，所以不符合判断条件，没有被replaceAll，再次bypas了漏洞修补。。。

**struts2的官方漏洞公告和修补后引发的安全缺陷（S2-004）**

这个漏洞的修补，比上一个更加令人无奈，这是一个/../获取资源文件的漏洞

S2-004 — Directory traversal vulnerability while serving static content

要了解这个漏洞的成因，大家需要先了解一个知识点。

当struts的FilterDispatcher收到url请求如下两个路径下的文件时：

http://www.inbreak.net/app/struts/

或

http://www.inbreak.net/app/static/

会去取struts核心包的core.src.main.resources.org.apache.struts2下面的静态资源文件，这些资源文件其实是一些js脚本和一些css文件。前文提到

<img onmouseover="domTT\_activate(this, …'StrutsTTClassic');alert('xss');a('','styleClass’…" />

代码中的domTT\_activate，其实就是

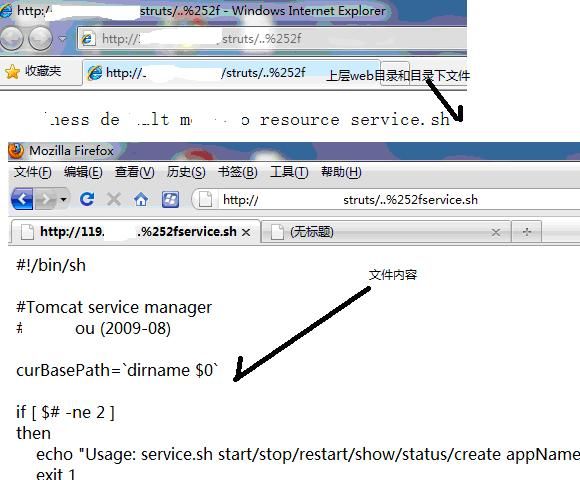
http://www.inbreak.net/app/struts/domTT.js

文件中的一个函数。

在struts2.0的时候，只要你敢上某几个版本的struts2，攻击者就可以通过

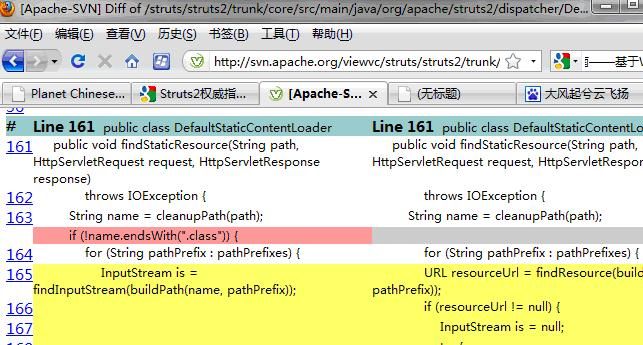
http://www.inbreak.net/app/struts/..%252f  
http://www.inbreak.net/app/struts/..%252f..%252f..%252fWEB-INF/classess/example/Login.class/

浏览你的web目录，下载web目录下的文件。  
先不说漏洞修补，请读者赶紧想想，你公司的开发人员，是否使用了struts2，并且把“Struts 2.0.0 – Struts 2.0.11.2 ”之间的几个版本包装了或者根本没有包装，直接上了web应用。如果有这种情况，就可以直接用以上方式攻击，这几天作者找了几个大型门户网站的漏洞，发现他们都存在这个漏洞，顺便下载了他们的数据库配置文件，同时报告了漏洞。



Struts官方可能也受到了攻击，于是修补了代码。  
作者同样查看了svn修补记录：

http://svn.apache.org/viewvc/struts/struts2/trunk/core/src/main/java/org/apache/struts2/dispatcher/DefaultStaticContentLoader.java?r1=674498&r2=687425&pathrev=687425&diff\_format=h



可以看到“ if (!name.endsWith(“.class”)) {”这行代码在修补漏洞时，被删除了。

修补前的代码中，为什么以前要过滤.class文件呢？是因为struts提供了一个功能：  
如果开发人员想自己使用这个静态文件映射的功能，可以配置web.xml

<filter>

<filter-name>struts</filter-name>

<filter-class>

org.apache.struts2.dispatcher.FilterDispatcher

</filter-class>

<init-param>

<param-name>packages</param-name>

<param-value>net.inbreak.action</param-value>

</init-param>

</filter>

如上配置后，当用户输入

http://www.inbreak.net/app/struts/domTT.js

的时候，实际上struts会去找net.inbreak.action这个文件夹下的domTT.js文件发给用户，而不再寻找核心包的那个文件夹了。这个功能开放后，官方为了防止对应包下的class文件被下载后反编译成源码，所以写了行代码，过滤.class文件。

就因为这行代码的存在，一时间，刚巧又正是struts2流行的时代。互联网大批的文章介绍struts2核心源码，在介绍到FilterDispatcher的时候，必然会提到，这里会过滤.class文件，如果开发人员使用这个功能，可以放心，自己的class文件不会被人下载。

后来出了漏洞，攻击者可以用

http://www.inbreak.net/app/struts/..%252f..%252f..%252fWEB-INF/classess/example/Login.class/

绕过官方限制，下载class文件。最终的确修补了这个/../的漏洞。然而悲剧的是，因为class文件实际上还是可以被下载，所以官方修补的同时，去掉了“if (!name.endsWith(“.class”)) { ”这一行代码，可能是觉得这一行代码太丢人。

曾经的教程，还在互联网上，告诉大家class文件不会被下载，官方也发表声明修补了/../漏洞。但是看到教程的开发们，却早已把目录映射了静态文件：

<param-name>packages</param-name>

<param-value>net.inbreak.action</param-value>

如果这个开发的net.inbreak.action包下有个UserLogin.class文件，在struts2有漏洞的版本，会面临服务器上所有文件被下载的命运。即使开发升级了struts，因为核心代码中的class文件的判断去掉了，导致这个文件还是可以被

http://www.inbreak.net/app/struts/UserLogin.class

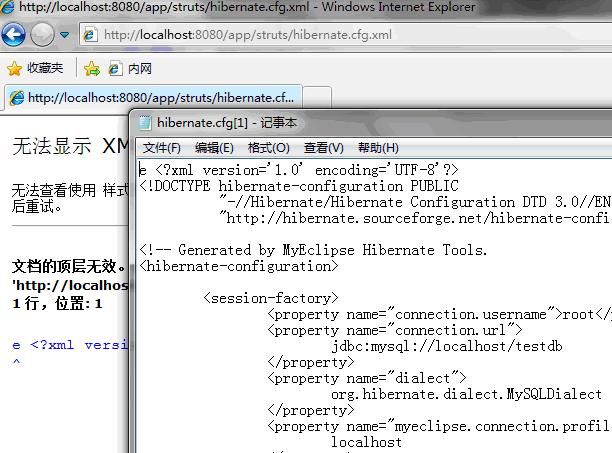
官方在没有任何通知的前提下，在教程满天飞告诉大家class文件不会被下载的前提下，为了面子悄悄的取掉了这个判断。这回好了，无论升级否，都不让人消停，万一开发人员头脑发热，配置如下：

<param-name>packages</param-name>

<param-value>/</param-value>

就出大事了，可以直接下载资源目录下所有文件。

http://www.inbreak.net/app/struts/hibernate.cfg.xml



**总结**

作者挖了struts2的一些漏洞，也挖了一些web其他框架的漏洞（不在本文范围），总结下挖取这些框架漏洞的方式。

首先，是上不上框架的问题。一旦开发用了这些框架，web应用会直接面临一些漏洞：

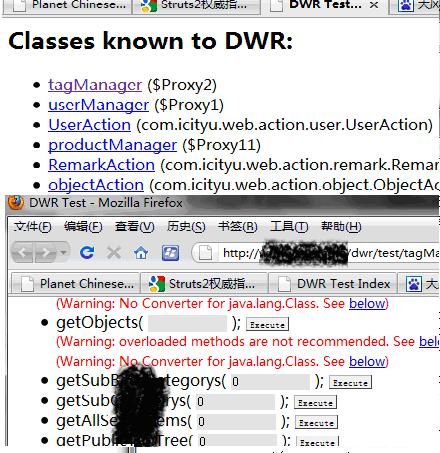
**1，开放了某功能，导致采用框架的应用默认漏洞**

因为这个框架在未经允许的情况下，悄悄的打开了某些功能，可能是为了方便开发等作用，结果导致了漏洞的产生。

举个例：  
比如DWR这个AJAX框架一旦使用，在某些版本里，输入

http://www.inbreak.net/app/dwr/

就能看到一个页面，里面是所有ajax方法的映射，甚至一些service方法没有配置，自动映射了出来。你可以在这个页面给那些映射出来的方法提交参数，调用方法。比如有个getUserpasswdByid的方法，看名称就知道，传递用户id，返回密码。



再举例就是本文中的struts2的../../漏洞。  
如果要挖这种漏洞，绝对是0day级别的漏洞，所以大家有必要怀疑每一个实现步骤，这种漏洞其实大部分就出现在开发环境和正式环境的差异，以及一些奇奇怪怪的功能点上。

**2，扩展了某功能，导致安全问题**

我们的web应用，本来是可以自己写代码实现一些功能的，但是这个框架为了方便开发和管理，把开发人员要写的代码包装了下，只要简单的几行就能实现原本大批代码才能做到的功能。而这些便利，却带来了很多安全问题。挖漏洞的同时，应该特别注意哪些地方比原来方便了很多，扩展了很多，这些扩展和方便，是否考虑到了安全因素。

比如本文介绍的struts的result（urlredirect）相关漏洞。

**3，DEMO或教程或用户指南误导**

自从框架出现后，为了让人们最快的了解和使用框架，官方出了很对用户手册，demo等功能。而很多大牛们，也相应的出了教程或者书籍，但是这些教程，DEMO，书籍上的例子，恰恰产生了很多漏洞。甚至开发本来不按照教程走，不会有漏洞，却被教程误导了。如果黑客看到这些书籍，请立刻把他列到你的扫描器中，绝对有不少开发会按照教程上去做。

例如本文提到的那个书籍介绍我们使用ajax的事情。

**4，原本有安全方案，但是被某功能代码覆盖**

这其实是一件悲剧的事情，告诉我们要注意在日常开发中和漏洞修补中，是否有不明真相的开发人员，为了实现某个功能，悄悄的把原本的安全方案断章取义的覆盖掉了。挖漏洞的时候，要特别注意安全方案附件的代码变动，很可能发现非常弱智的逻辑问题。

例如本文提交的class文件的判断。

**5，不完善的安全**

一个安全方案的实施，应该是彻头彻尾的，要注意方案的完整性，不能有些地方方案OK，有些地方不能实施。在挖漏洞的时候，也不要被安全方案吓住，并不是有了方案，听起来牛X，就绝对不会有漏洞的，最起码应该经过全面fuzz。

例如本文提到的XSS遗漏点，以及富文本的遗漏。

**6，版本升级后，没有醒目安全公告**

我们知道所有的架构师都不愿意有事没事升级框架，特别是不稳定的框架版本，因为这个升级可能带来很多不可预料的问题，所以可能即使看到了安全公告，也没有去升级。如果不懂安全，更不愿意升级框架了。所以官方必须做到一个漏洞的修补，一个公告的发布，必须带有相关的代码log。告诉大家具体哪里做了改动。而挖漏洞的同学更应该盯紧这些地方，百般推敲和测试修改的部分，不要被一次公用的测试结果吓到，模糊的认为漏洞已经修补了。

**7，悲剧的方案**

很多时候，我们会看到官方修补漏洞，或者一些安全方案的实施结果。那是不是真的都能达到修补漏洞的效果呢？

例如本文的<s:url>标签的xss漏洞，官方都这个漏洞的修补，真是绞尽脑汁，最终还是悲剧了。

**8，优秀的方案，悲剧的执行**

RT，不再说明。

**9，挑战web服务器配置**

这个问题有必要说一说，struts有事没事做个静态映射做什么？其实是目的就是为了框架和应用分离，很明显那些js文件应该放在项目中的web目录下，但是为什么要做呢？还不是因为struts包发布的时候，还没有项目，只有框架。

它为了达到即使上了任何项目，也能有办法访问到它提供的那些js的目的，只好被逼无奈做了这个功能，静态目录映射。无论任何项目，部署后，只要url后面根目录加上/struts，或者/static就可以访问js。后来做了这个功能感觉良好还不算，居然把功能提供出来，给推荐给开发人员使用。归根结底是因为struts挑战了web服务器的配置，非要自己做静态映射。要知道人家web服务器做的映射，是经过多少年黑客入侵打磨出来的，struts有吗？

这种情况突出在单独映射某目录，单独对某目录做权限，做DIR列表等功能，如果你看到一个框架也做了这种功能，恭喜你！赶快去挖，十有八九存在漏洞！

**10，没有安全方案，也没有提醒**

本文其实没有提到一些web漏洞，比如csrf，比如session fix，比如传输加密等，很明显struts是存在漏洞的，只是作者觉得这些东西没必要这里说，大家都是明眼人，看到form里没有token，百分百csrf嘛！

想想官方，官方也明明知道上了自己的框架后，会存在这些漏洞，为什么连个提醒都没有呢？本来开发就不知道，你又藏着掖着。如果框架负责任，发个公告，说如果你用了我的框架，实际上要小心什么什么攻击。。。

呃。。。我明白官方为什么不敢说了。-\_-!  
这些框架除了那些“只要你用，必然有漏洞”的安全缺陷外，还有不少问题，会出现在开发人员使用框架的过程中：  
1，两个框架都方便，结合起来有漏洞  
有个框架叫做Spring security，是基于url的访问权限控制，做的很好。如果你不是管理员，绝对不能访问admin目录。但是有很多web框架，访问一个action或者访问一个controller，不止一个url可以访问，在这里做了兼容性处理，多个url指向同一个应用，导致Spring security这种基于url的访问控制，名存实亡。

2，开发人员“正常”使用框架后，可能产生漏洞  
这是最最惨的事情，框架绝对不会认领这种类型的漏洞的，他会认为这是开发的问题。然而本文的“action方法暴力破解、url redirect扩大化”这两个安全缺陷，实际上也是框架存在的意义（方便开发）带来的后果。官方会修补么？我想它顶多会说，大家不要这样那样而已，绝对不会做什么安全方案的。要知道这些漏洞是具有struts或者webwork特色的，只有使用了这些框架才会有的。

3，危险功能点  
框架带来了一些功能点，比如Tag lib的一些XSS，只要使用，必定有漏洞。

4，框架给开发挖坑  
这根本就是陷阱，还是要说/static、/struts，如果开发不配置，顶多下载个js，影响不大，万一开发使用了这个功能，映射了某个目录，那就掉进坑里去了。

5，一个框架带来的漏洞被另一个框架最大化  
本文提到了变量默认值被覆盖后，又因为hibernate不需要写sql语句，而最终被存储进了数据库中。  
回想一个问题，如果让我们自己写sql语句实现，难道在添加普通用户的时候，会真的专门写个变量，接收用户传进来注册管理员的字段么？

但是这是hibernate得问题么？当然不是，只是因为有了它，导致漏洞更加严重而已。

**补充**

本文对struts2的种种安全缺陷，就提到这里。个人觉得这是一个挖漏洞的方向，对framework漏洞的挖掘。  
可能大家都专注于代码安全，没有太多的去看框架本身是否安全，以及使用了框架后，是否真的安全。  
所以很多人忽略这个问题，我相信这不是一个新的开始，也不是一个结束，只是让大家开始更加重视框架安全。  
作者也仅仅在本文提到了struts，webwork这两个框架，事实上框架很多，他们真的安全么？有待考证！  
最后写给那些真正打算把技术应用于实践的同学们，框架漏洞扫描器，是可以做出来的，对于难以解决的猜解问题，可以对网站蜘蛛一下，然后保存那些开发人员喜欢使用的字段名称，关注各个input的名字，action的名字，目录名字等，生成猜解一个列表。而判断是否用了struts更加简单：

特征1：XXX.action 可能是struts或webwork  
特征2：XXX.do 可能是struts或webwork  
特征3：XXX!bbb.action 可能是struts或webwork  
特征4：XXX/struts/..%252f 必然是struts2  
其他细节处，不再一一给出，有待大家发掘。  
相关的漏洞修补方式，作者不在此处献丑，请大家根据漏洞原理自行处理。

**感谢**

—————————————-  
感谢axis （http://hi.baidu.com/aullik5）对本文提出的建议。  
感谢幻影webzine http://webzine.ph4nt0m.org/  
感谢struts所有开发人员种种有趣的代码。  
作者DEMO中有敏感数据，恕不能提供，请大家自行搜索strutsdemo搭建环境测试。  
Struts2：http://struts.apache.org/2.x/  
Struts2 svn：http://svn.apache.org/viewvc/struts/struts2/trunk/

**Tags:** [Struts2](http://huaidan.org/tag/struts2), [WebZine](http://huaidan.org/tag/webzine)

**相关日志**

* [[PSTZine 0x01][0x06][利用httponly提升应用程序安全性]](http://huaidan.org/archives/1816.html)
* [SWAN访谈](http://huaidan.org/archives/2077.html)
* [Ph4nt0m Webzine 0x03](http://huaidan.org/archives/3055.html)
* [[WebZine]Fuzz客户端存储对象，寻找client ddos](http://huaidan.org/archives/3436.html)
* [[WebZine]卡巴虚拟机启发式查毒的绕过方法](http://huaidan.org/archives/3440.html)
* [[PSTZine 0x01][0x01][Introduction]](http://huaidan.org/archives/1811.html)

上一篇:[[WebZine]Flashsky访谈](http://huaidan.org/archives/3416.html)

下一篇:[[WebZine]重谈IP欺骗技术](http://huaidan.org/archives/3435.html)

**楼被抢了 17 层了...**[**抢座**](http://huaidan.org/archives/3433.html#respond)**、**[**Rss 2.0**](http://huaidan.org/archives/3433.html/feed)**或者**[**Trackback**](http://huaidan.org/archives/3433.html/trackback)

* http://0.gravatar.com/avatar/c118e0f1e16b96aada689d8442ee6ea7?s=40&d=mm&r=g**idear** | 2010/01/10 11:39 | [#](http://huaidan.org/archives/3433.html#comment-14503)

struts2确实存在一些问题，本人还发现了struts2整合spring的Action的ognl来的问题。  
Action的service属性应该被严禁访问，但由于注入的service的事务代理类，不能用spring的类型方式注入，只能按属性名匹配注入，所以还是得提供setter方法，无奈啊！

[回复该留言](http://huaidan.org/archives/3433.html?replytocom=14503#respond)

* http://2.gravatar.com/avatar/bb151f6019d37271279669c555981e1f?s=40&d=mm&r=g**jdkleo** | 2010/01/26 13:58 | [#](http://huaidan.org/archives/3433.html#comment-14586)

是否已登录？  
答：逻辑程序员不应关心用户是否已登录  
应该由登录程序员设置拦截器或公共父类处理  
————————  
把人边hibernate.xml都下下来了.  
答：还是一样拦截器，把一些不允许直接访问的路径通通打回  
————————  
你不看这文章 你知道哪个允许 哪个不允许？  
这哥们太牛了  
答：这个是你业务决定的,STRUTS2没必要为你考虑安全性，你自己设置  
——————  
可以查看DWR  
答：DWR可以设置不允许直接访问（即关闭调试模式），纯属程序员自己没有配置好，STRUTS2可能没有给出这样的配置，那么程序员就要自己去做拦截器来实现这样的功能，要DIY DIY,记住我们不是.NET程序员

[回复该留言](http://huaidan.org/archives/3433.html?replytocom=14586#respond)

* http://0.gravatar.com/avatar/9c460e0f1f6728036cf15ace2bee2040?s=40&d=mm&r=g**gg** | 2010/03/12 11:39 | [#](http://huaidan.org/archives/3433.html#comment-14921)

完全是自己的程序问题，怎么怪到struts2或ww上了，奇怪的逻辑！

[回复该留言](http://huaidan.org/archives/3433.html?replytocom=14921#respond)

* + http://0.gravatar.com/avatar/6a7fb06ff9b47454a5f76f5d54e52465?s=40&d=mm&r=g[**hiswing**](http://www.mxjava.com/) | 2010/04/26 17:20 | [#](http://huaidan.org/archives/3433.html#comment-15244)

这篇文章很是莫名其妙。

[回复该留言](http://huaidan.org/archives/3433.html?replytocom=15244#respond)

* http://1.gravatar.com/avatar/18d364764417e172f206e70ca91c96ac?s=40&d=mm&r=g**不填说不了话** | 2010/05/11 13:18 | [#](http://huaidan.org/archives/3433.html#comment-15263)

精湛

[回复该留言](http://huaidan.org/archives/3433.html?replytocom=15263#respond)

* http://1.gravatar.com/avatar/dd79a40ff91ee43cbb8ec172f7cee82e?s=40&d=mm&r=g**路过** | 2010/05/24 22:55 | [#](http://huaidan.org/archives/3433.html#comment-15389)

完全赞同jdkleo 回答！  
哥么你太扯了  
struts2你学过吗？  
没有的话 不要乱扯哈！

[回复该留言](http://huaidan.org/archives/3433.html?replytocom=15389#respond)

* http://2.gravatar.com/avatar/ec059f5661d1332db6e112c14851f64b?s=40&d=mm&r=g**WEB** | 2011/06/18 14:01 | [#](http://huaidan.org/archives/3433.html#comment-18707)

完全 是瞎扯~

[回复该留言](http://huaidan.org/archives/3433.html?replytocom=18707#respond)

* http://1.gravatar.com/avatar/a13e32579b109789df92f31ac06b5add?s=40&d=mm&r=g**@web** | 2012/03/15 13:41 | [#](http://huaidan.org/archives/3433.html#comment-20032)

呵呵,我觉得这文章写的太过于自我陶醉了,你觉得这些与用户打交道的数据不做任何处理觉得可能吗?如果真是这样只能说这个程序员不适合在这个行业,还有一个字段权限的问题,这完全靠自己的猜测说的不好听就是瞎扯..虽然我只是第一次看你的blog,但是这样的文章也写而且还是说framework的问题,我真是觉得做安全觉得很可笑.我觉得任何其它评论就没有必要在多写了,自我反省吧,但是这篇文章说是给程序员一个忠告还差不多,一上来就取这么一个名字..

[回复该留言](http://huaidan.org/archives/3433.html?replytocom=20032#respond)

* http://0.gravatar.com/avatar/c7147a79a0b03d84e6701648c4c3493c?s=40&d=mm&r=g**TestOK** | 2013/02/21 14:28 | [#](http://huaidan.org/archives/3433.html#comment-20642)

这篇文章看起来很有道理，就像上面有人说的，完全是自我陶醉，比如那个用户type那个，你怎么知道人家就有一个type？而哪个值又代表是管理员？像数据库一样，我们都知道有一个用户名和密码，那你就去猜呗…  
非struts的问题。  
很多结论有些莫名其妙，完全是作者一厢情愿的臆想，我记得高中的时候一个同学说自己的生日正好和礼炮空间站坠落的日期一样，于是就不得了了，自己是什么什么转世…

[回复该留言](http://huaidan.org/archives/3433.html?replytocom=20642#respond)

* http://1.gravatar.com/avatar/aed782d667357ac45c0f3c1f370c1b6a?s=40&d=mm&r=g**午夜狂刀** | 2013/07/19 16:03 | [#](http://huaidan.org/archives/3433.html#comment-20972)

赞! Struts的问题的确非常多;楼主所提出的场景假设和攻防非常有代表性;框架都在努力提升自己的功能和性能,如果什么都得程序员考虑,那还要框架做什么?

[回复该留言](http://huaidan.org/archives/3433.html?replytocom=20972#respond)

* http://0.gravatar.com/avatar/387ac4dc6d171a2c12672816d43e5ecb?s=40&d=mm&r=g**z** | 2013/07/20 13:46 | [#](http://huaidan.org/archives/3433.html#comment-20974)

天哪，struts2竟然有这么多安全漏洞，现在都修复了吗？

[回复该留言](http://huaidan.org/archives/3433.html?replytocom=20974#respond)

* http://1.gravatar.com/avatar/a99ece4292e96e7ca697383dcf0ce72b?s=40&d=mm&r=g**xpg** | 2013/08/26 14:50 | [#](http://huaidan.org/archives/3433.html#comment-21042)

明白你的意思，你攻击下联通的官网吧。http://www.10010.com 或许您要访问的页面不存在。

[回复该留言](http://huaidan.org/archives/3433.html?replytocom=21042#respond)

* http://2.gravatar.com/avatar/b58efa6c8c88b7878aa28966b604649f?s=40&d=mm&r=g**fu** | 2013/08/28 09:58 | [#](http://huaidan.org/archives/3433.html#comment-21045)

厉害，struts2确实爆出了严重漏洞，不知道新版本是如何修复的

[回复该留言](http://huaidan.org/archives/3433.html?replytocom=21045#respond)

* http://1.gravatar.com/avatar/addad1ce59fa4054527d51f18711ac66?s=40&d=mm&r=g[**CFH**](http://www.hao123.com/) | 2013/09/30 16:28 | [#](http://huaidan.org/archives/3433.html#comment-21116)

诶，其实作者完全是抓住Struts2的一个动态动用，直接设个禁用动态方法调用就没上面的问题了

[回复该留言](http://huaidan.org/archives/3433.html?replytocom=21116#respond)

* http://0.gravatar.com/avatar/0c5a5166a5c571b092950468a8490499?s=40&d=mm&r=g**莽游** | 2015/02/09 11:43 | [#](http://huaidan.org/archives/3433.html#comment-439629)

action 中通用方法改成 private，只暴漏 对外访问action方法。 但是 对于 get 方法也是无可奈何啊。