CVE-2021-31805_Apache_Struts2_远程代码执行漏洞

漏洞描述

近日，Apache 官方发布了 Apache Struts2 的风险通告，漏洞编号为 CVE-2021-31805。Apache Struts2 是一个用于开发 JavaEE 网络应用程序的开放源代码网页应用程序框架。
此次 CVE-2021-31805（S2-062）是由于 CVE-2020-17530（S2-061）修复不完整造成的。

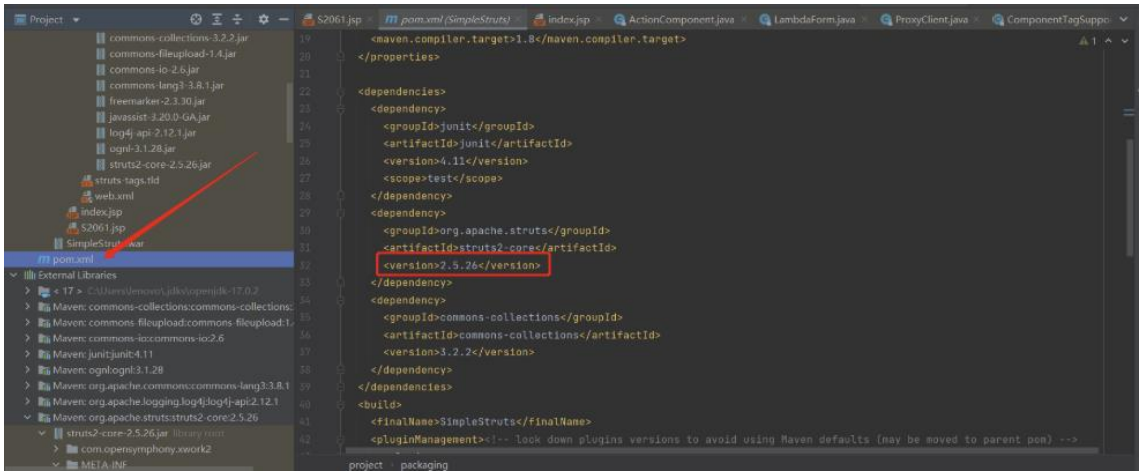
利用范围

2.0.0 <= Apache Struts2 <= 2.5.29

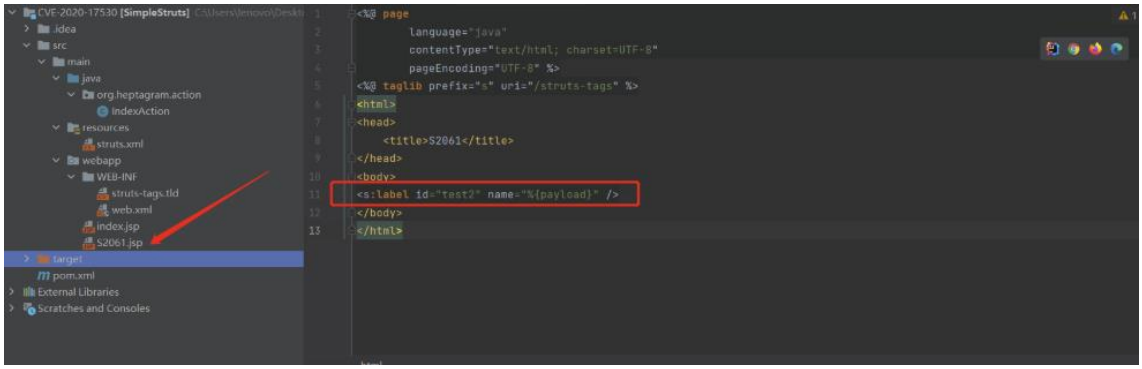
漏洞分析

环境搭建

目前网上已经公开出很多集成环境，如果只需要简单复现漏洞，可以使用 docker 集成环境或者网上公开的在线靶场。
如果想要深入地探究漏洞产生原理，建议使用 idea 手工搭建环境。这里我们使用 s2-061 的漏洞 demo 来进行改造。
在下载好 s2-061 的 demo 源码后，我们需要修改三个地方。
修改一：
maven pom 文件修改 struts2 版本为 2.5.26。

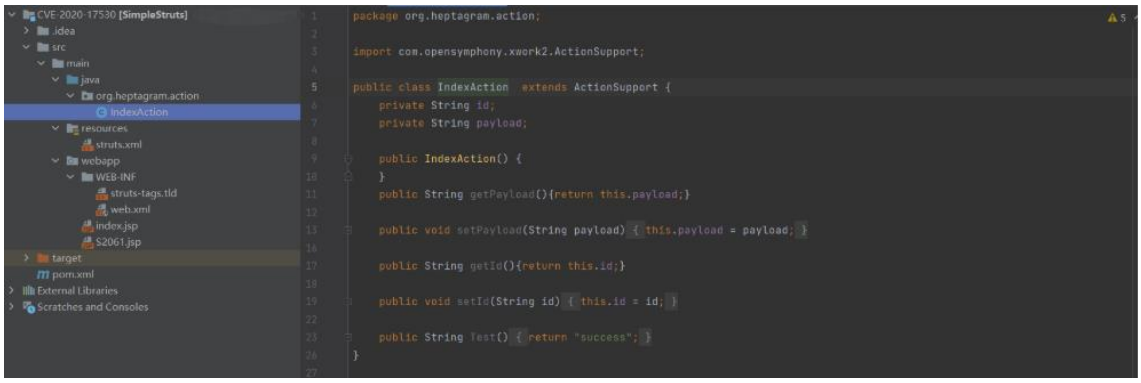


修改二：
前端 jsp 文件，修改 name 属性标签，通过%{payload}进行赋值。



修改三：

修改 action 类

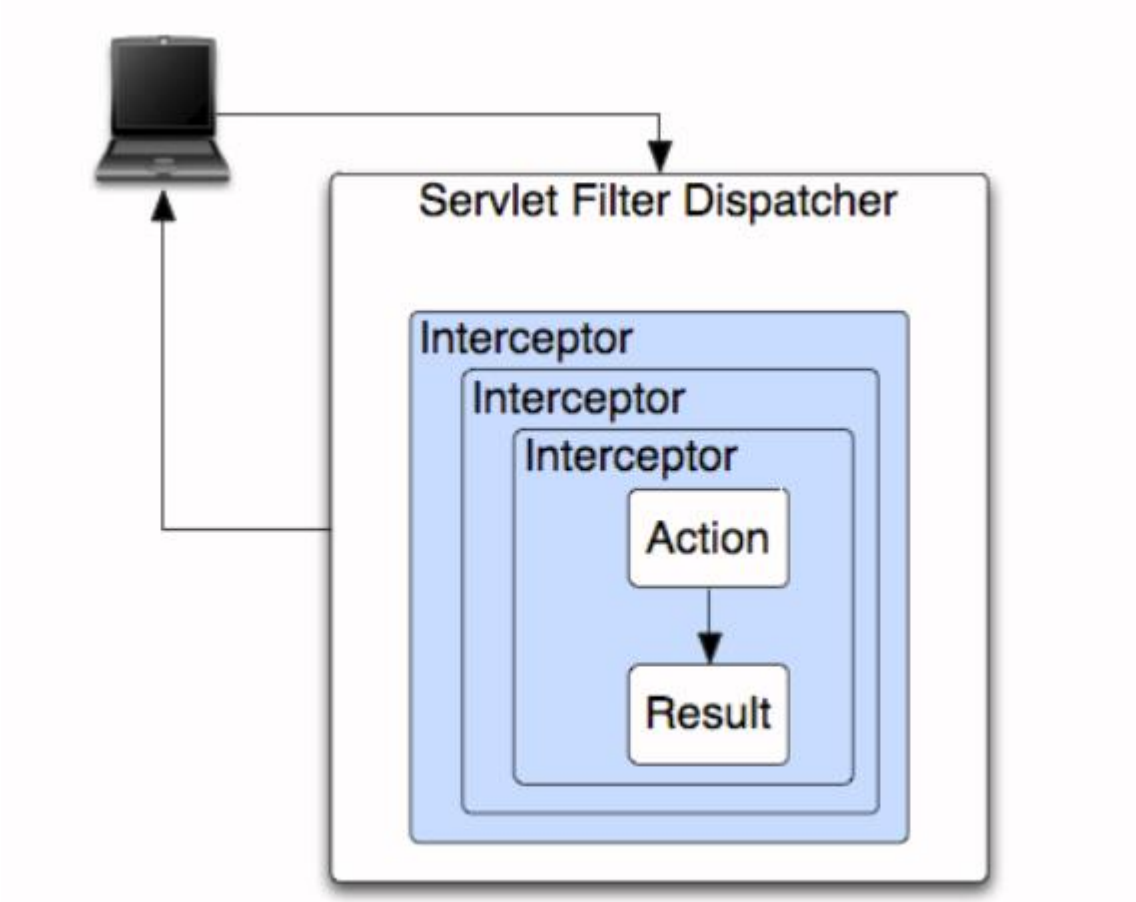


在完成改造之后，只需要将项目编译后运行在 tomcat 容器中即可。

前置知识

在进行漏洞分析之前，我们需要了解一些有关 struts2 框架的基础知识，这样才能更好地去理解漏洞产生的原理。

Apache Struts 2 架构概述



1. 客户端提交 web 请求，指向一个 Servlet 容器，Servlet 容器初步解析该请求
2. 核心处理器 Filter Dispatcher 协调其他控制器处理请求并确定合适的 Action
3. 拦截器自动将通用功能应用于请求，例如工作流、验证和文件上传处理
4. Action 方法执行，通常存储或从数据库中检索信息
5. 结果将输出呈现给客户端，可以是 HTML、图像、PDF 或其他内容

OGNL 语法介绍

OGNL 的全称是对象图导航语言（ Object-Graph Navigation Language），它是一种用于获取和设置 Java 对象属性的开源表达式语言，以及其他附加功能，是 Struts2 的默认表达式语言。

使用这种表达式语言，可以利用表达式语法树规则，存储 Java 对象的任意属性，调用 Java 对象的方法，同时能够自动实现期望的类型转换。

如果将表达式看作是一种带有语义的字符串，那么 OGNL 就是这个语义字符串与 Java 对象之间的沟通桥梁，其功能就是双向转换语义字符串与 Java 对象数据即转换 String 和 Object。

OGNL 执行操作三要素：

表达式（Expression）、根对象（Root Object）、上下文环境（Context）

OGNL 三个重要操作符号

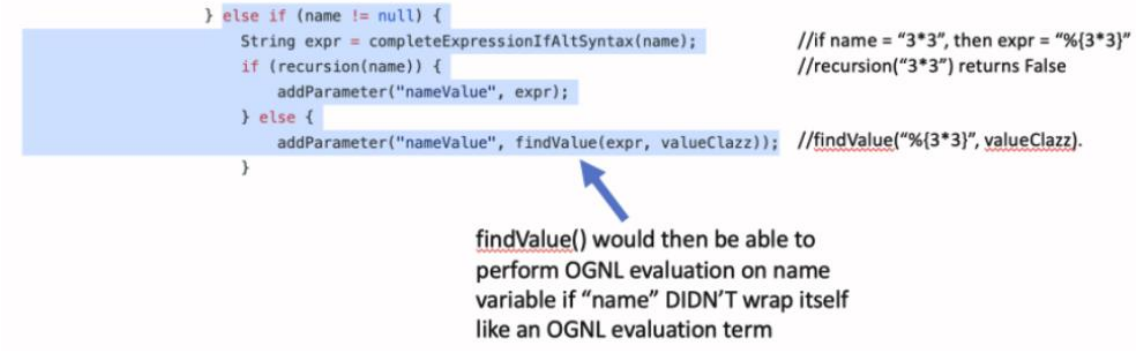
OGNL 中的三个重要符号：#、%、$，这里重点介绍%，其用途是在标志属性为字符串类型时，计算 OGNL 表达式的值，类似 JS 中的函数 eval()。例如：<s:url value ="%{items.{title}[0]}"/>。

漏洞原理

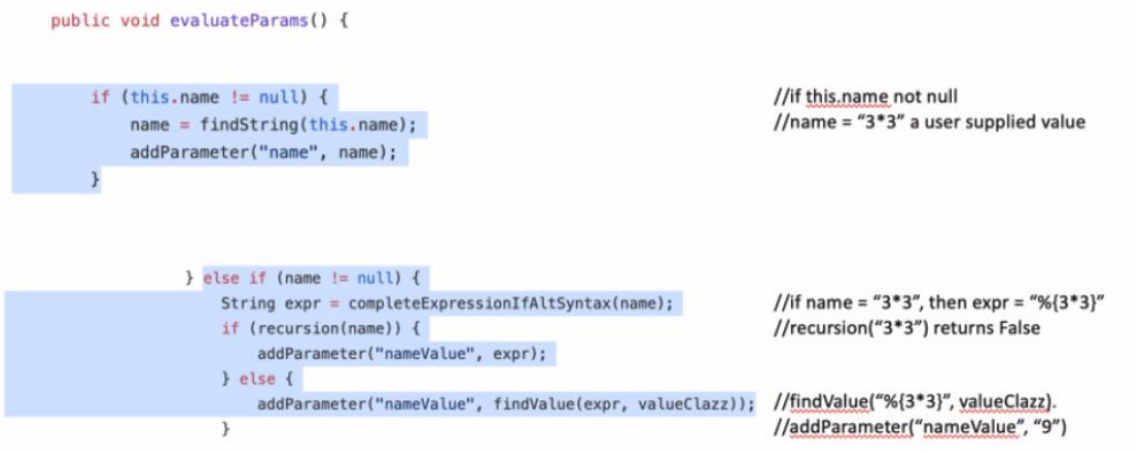
参考国外大神的理解，在 s2-061 问题中，使用在 jsp 中定义的类，类似如下 idVal=%{3*3}输入将执行双重 OGNL 评估，从而导致 id="9"

```
Plain Text
//example
<s:a id="%{idVal}"/>
//result
<s:a id="9"/>
```

从 diff 分析，核心问题的部分在于属性 name 会调用 completeExpressionIfAltSyntax 函数并将其分配给 expr，但在最终 OGNL 解析 expr 之前对 name 进行了递归检查。



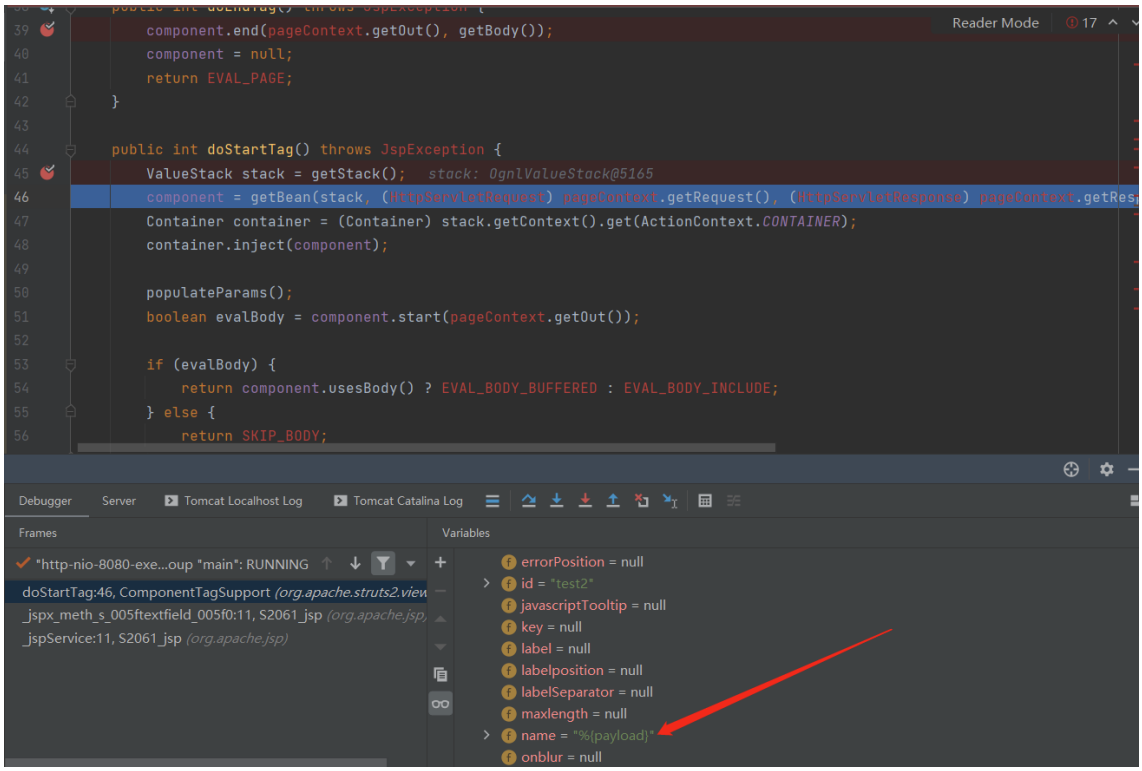
但是如果不对 name 进行第二次 OGNL 解析，name 将不会包含用户提供的来自 URL 参数的数据。但是在前面的 evaluateParams 函数中却执行了另一个 OGNL 解析。



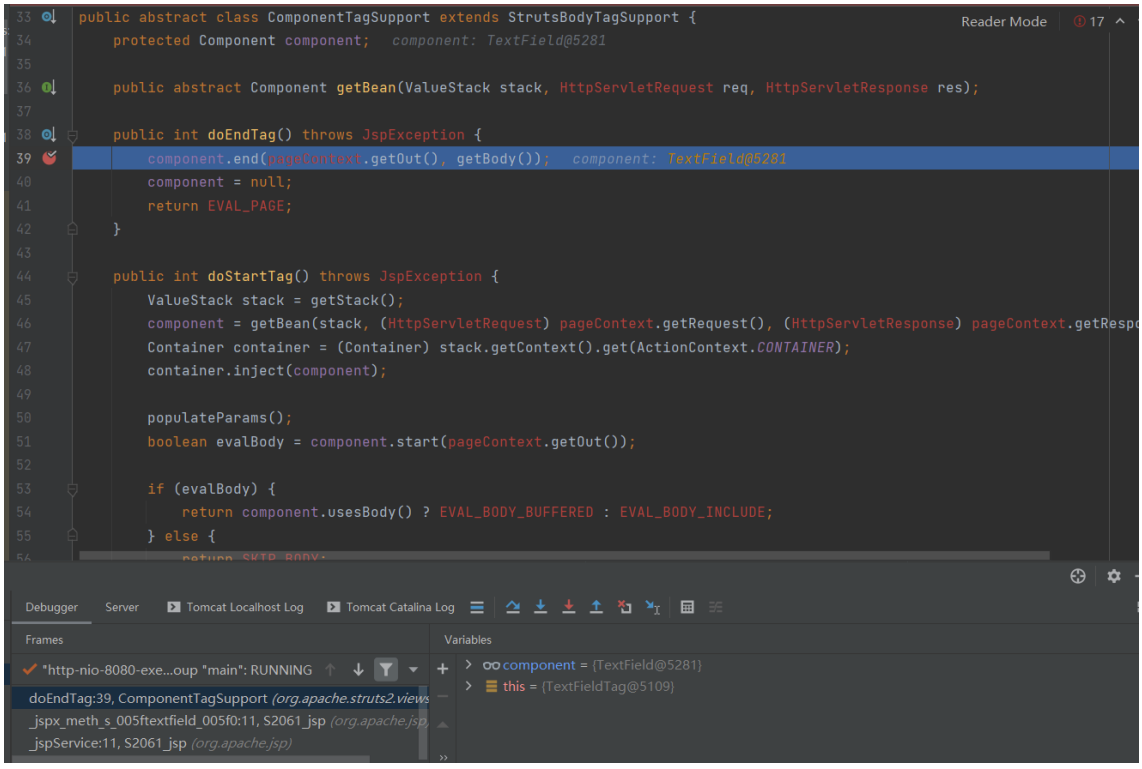
所以对于某些 UIBean 标记的名称属性就很容易受到两次 OGNL 解析，这就导致了远程代码执行。

动态调式

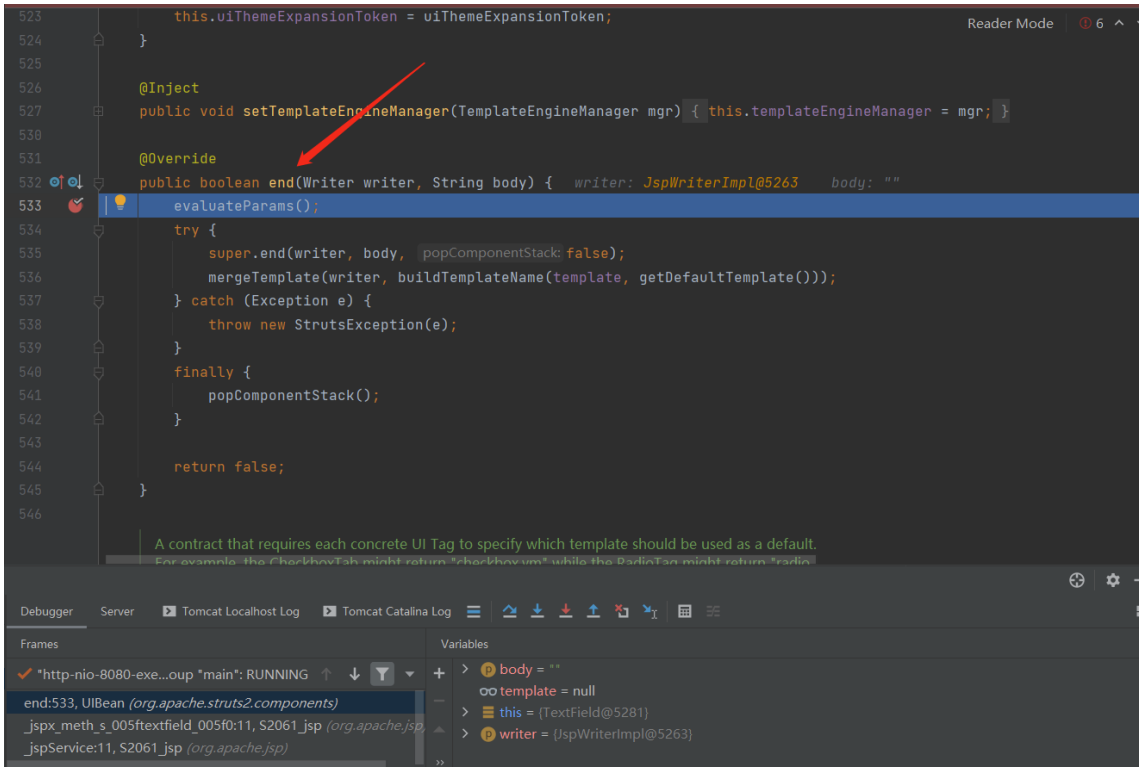
首先在 org.apache.struts2.views.jsp.ComponentTagSupport#doStartTag 处打下断点，这里对标签开始解析



到 org.apache.struts2.views.jsp.ComponentTagSupport#doEndTag 结束对标签解析

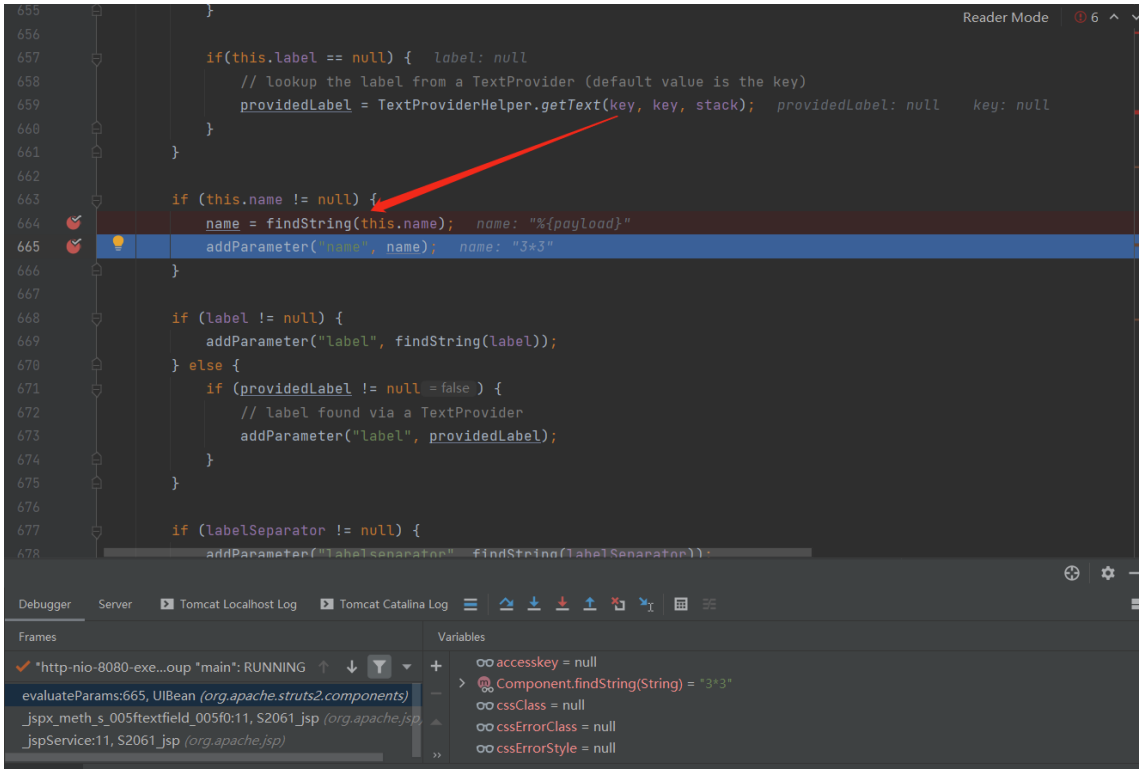


随后进入 org.apache.struts2.components.UIBean#end

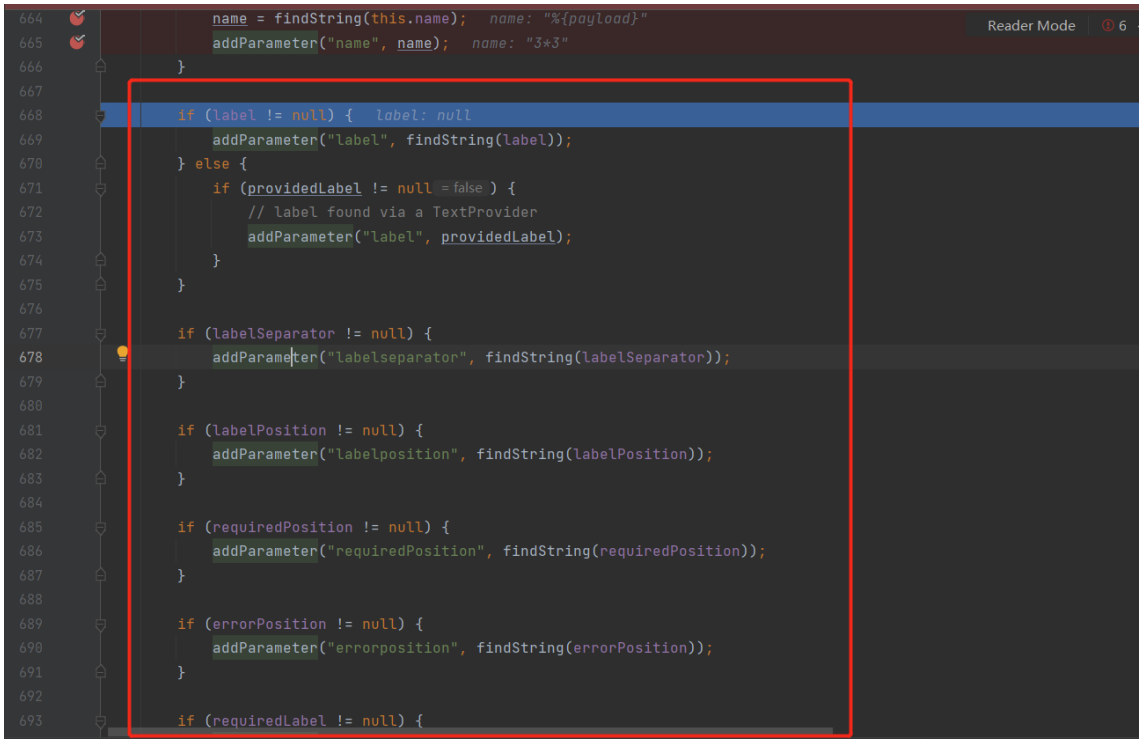


进入 evaluateParams 函数

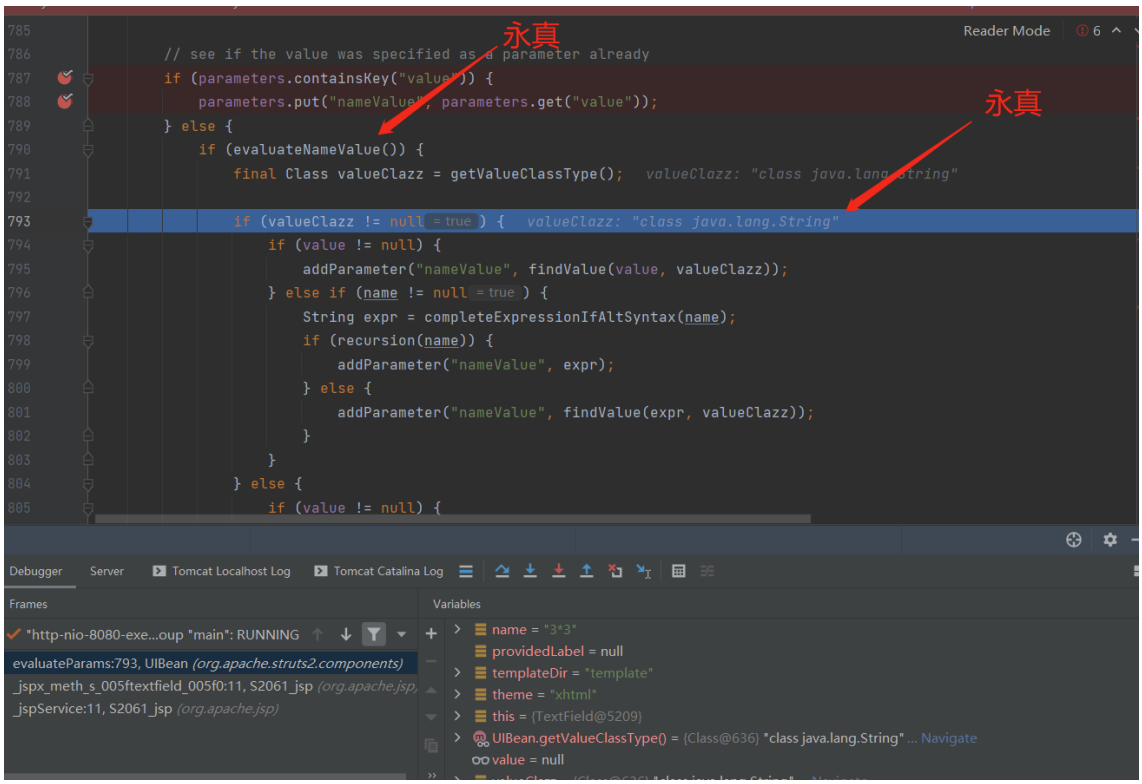
在进入 evaluateParams 函数往后，会调用 findString 对属性 name 进行一次 OGNL 表达式赋值（这里我们标记为第一次 OGNL 表达式赋值），此时 name 已经被赋值为我们所提交的 payload



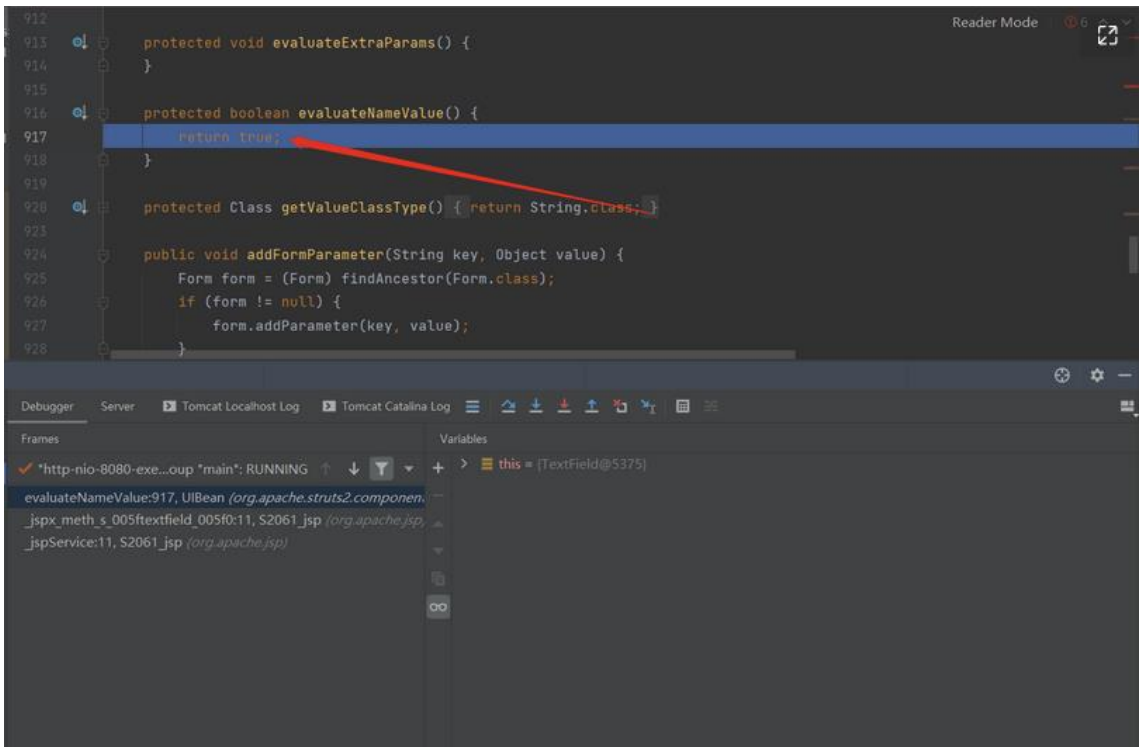
继续往下，会对一系列的属性进行判断，目的是看这些属性是否能被利用

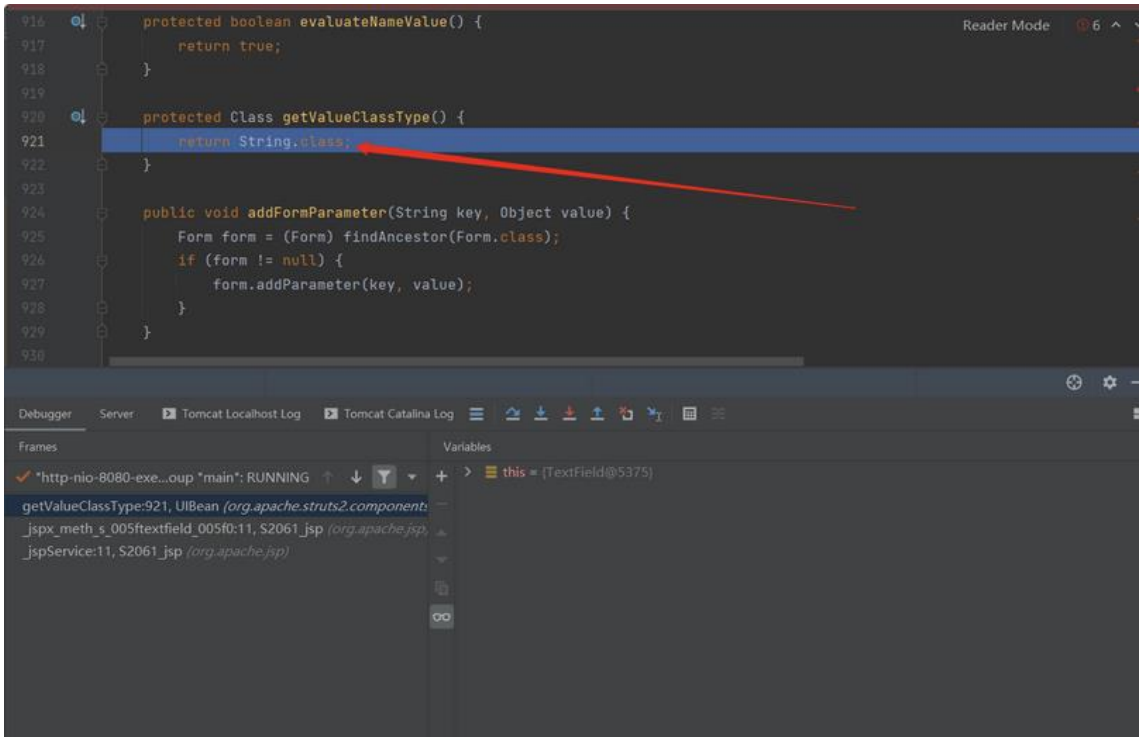


在判断完毕之后，会对属性 name 进行判断

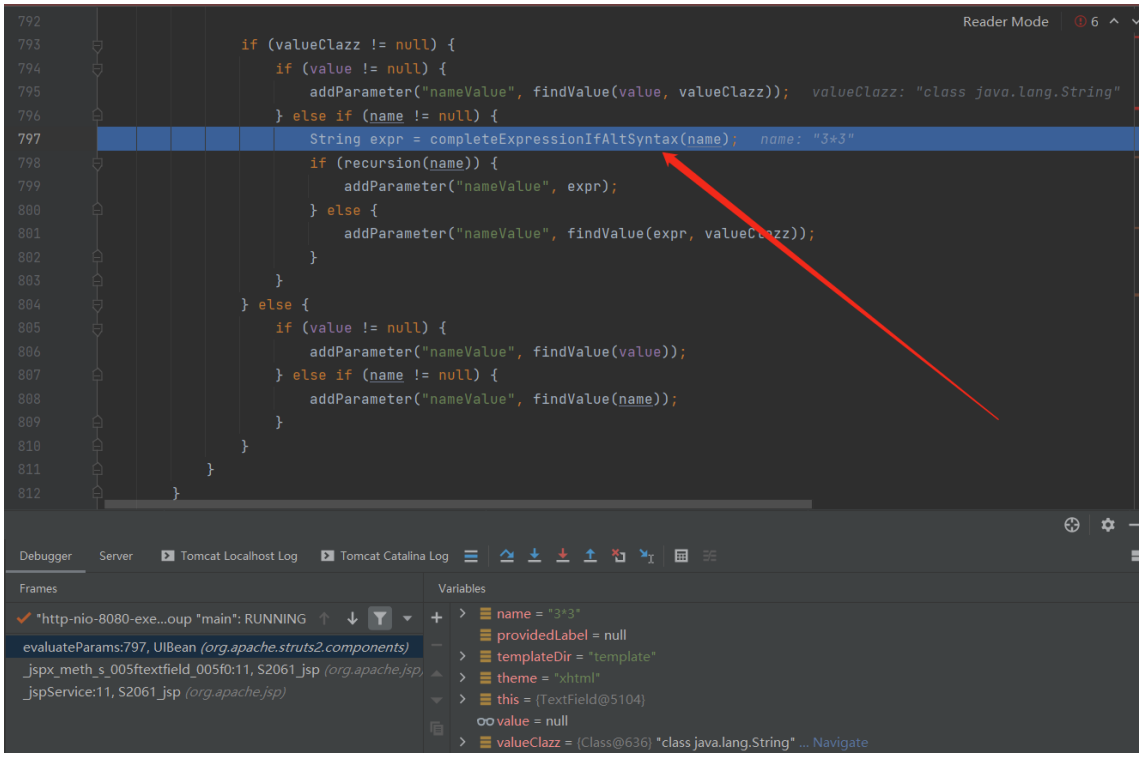


判断的结果如下，此时的 name 属性不存在 value 且非空，所以之前两个 if 的判断都为永真

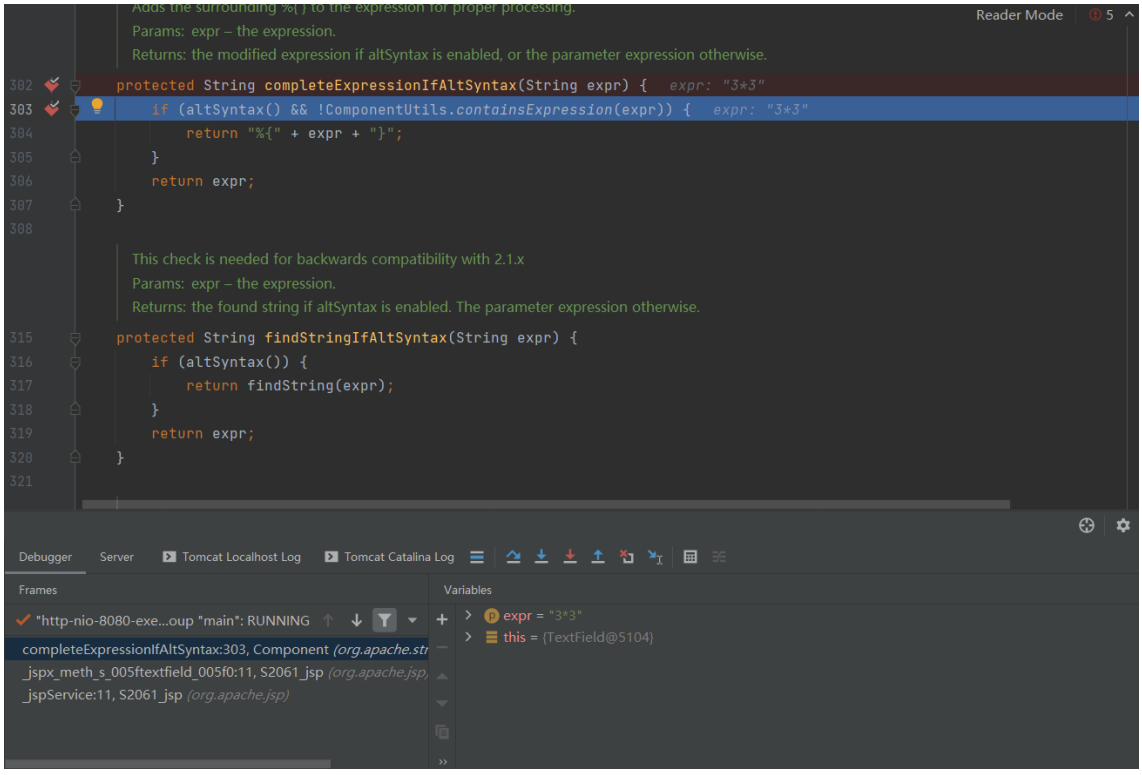




顺利进入 completeExpressionIfAltSyntax 函数

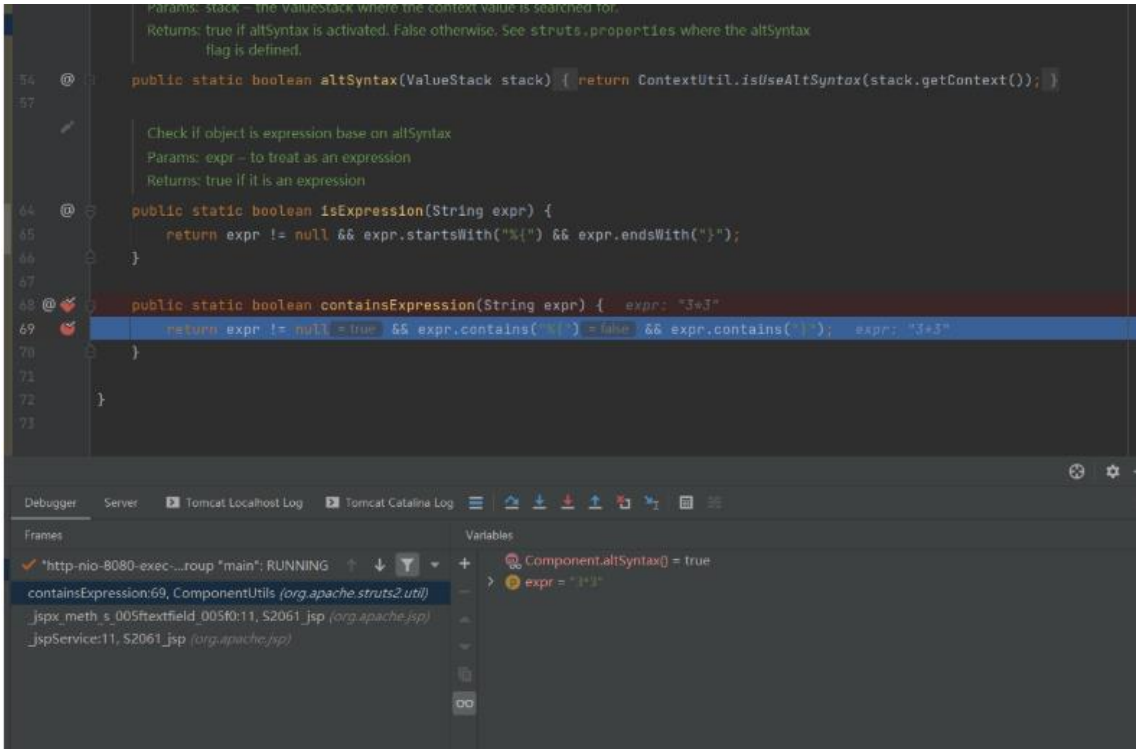


随后进入 org.apache.struts2.components.Component#completeExpressionIfAltSyntax，判断 altSyntax，在 s2-001 修复之后，altSyntax 功能默认是关闭的。

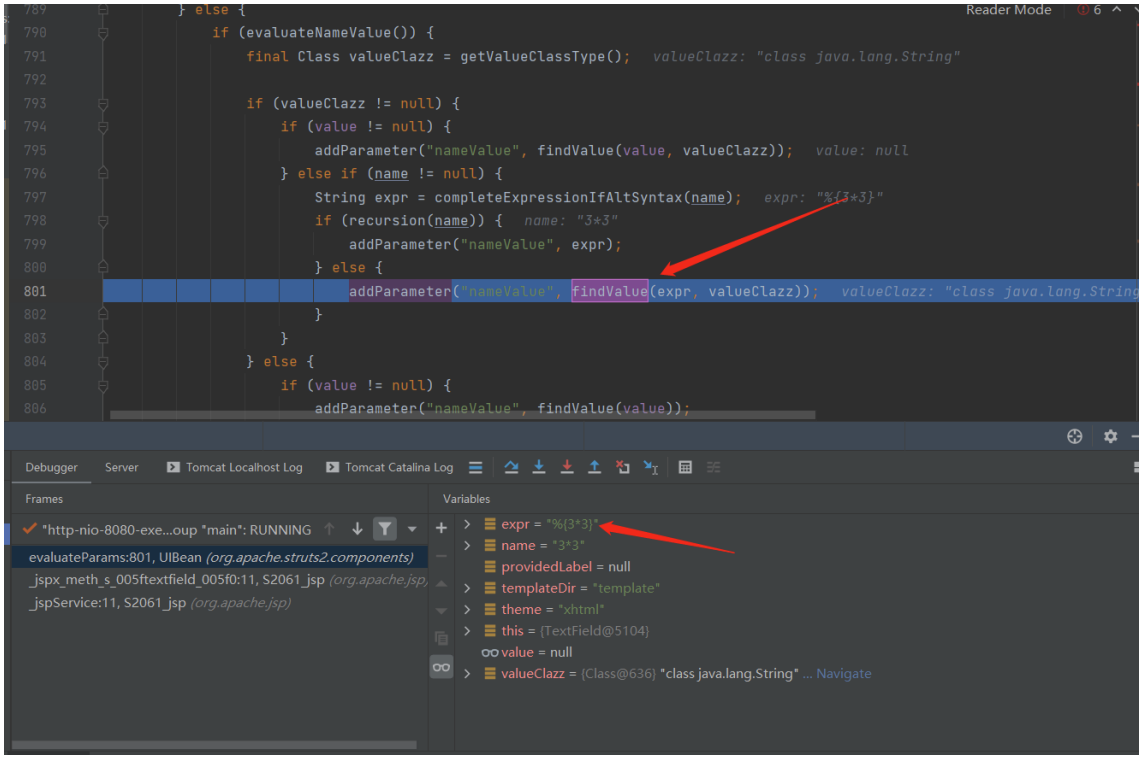


同时往下会对表达式进行检查，看其中是否包含%{}，如果没有就会自动加上%{}

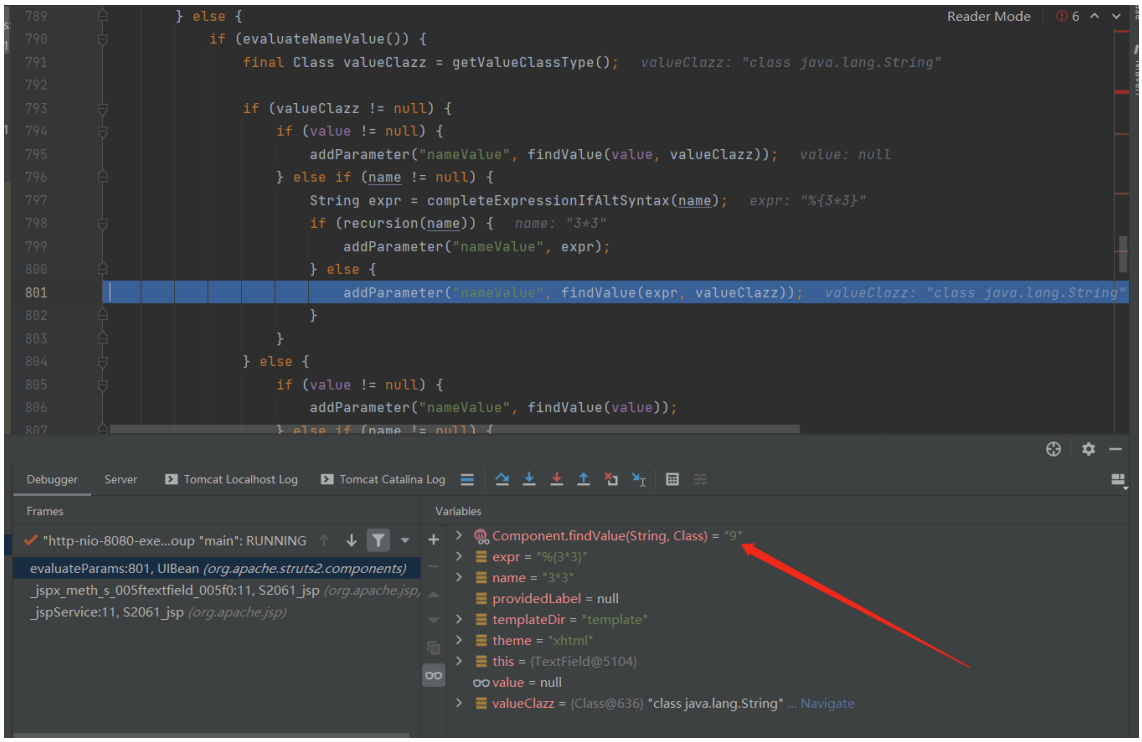
在 org.apache.struts2.util.ComponentUtils#containsExpression 中，具体检查表达式是否含有%{}如下：如果包含%{}就会返回 ture，就不会进入后续的 findValue，反之亦然。



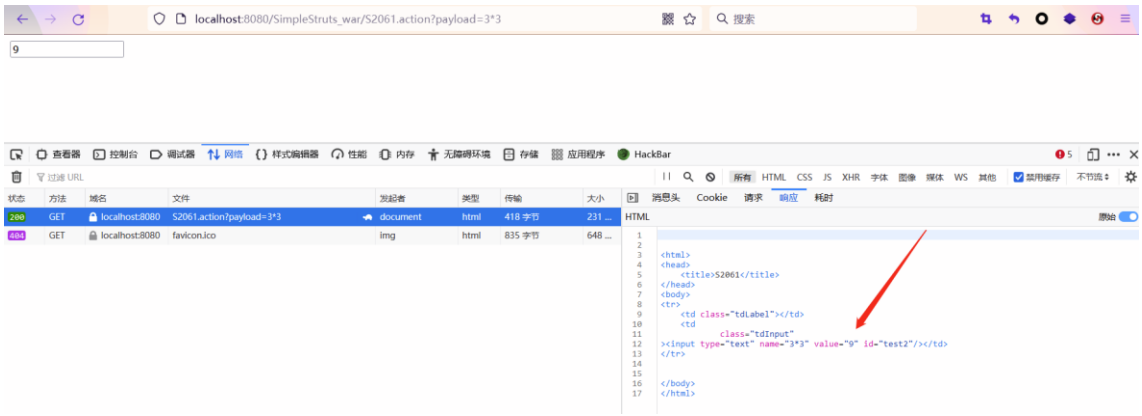
此时，在 org.apache.struts2.util.ComponentUtils#containsExpression 判断的结果为 false，我们的表达式也自动被加上了%{}。接下来就进入到 findValue，在这里，表达式会进行二次 OGNL 表达式赋值。



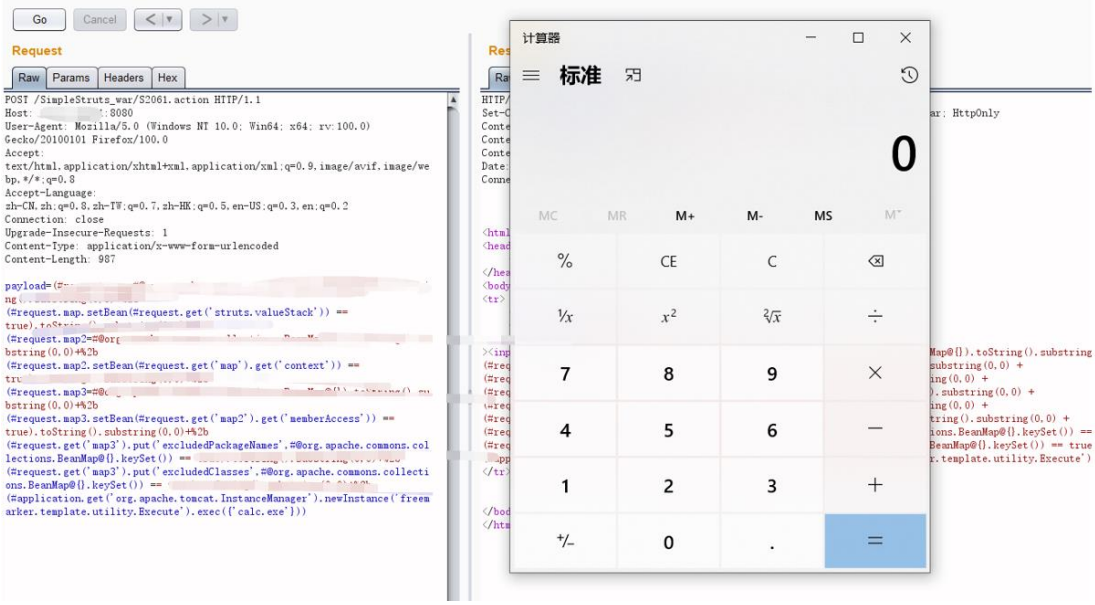
继续跟进，可以看到 nameValue 成功被解析，返回结果为“9”



此时浏览器中也成功解析



漏洞利用



修复建议

目前官方已发布修复版本修复了该漏洞，请受影响的用户升级到安全版本：
https://cwiki.apache.org/confluence/display/WW/Version+Notes+2.5.30