CVE-2022-26134_Confluence_OGNL_RCE 漏洞

漏洞描述

最近 Confluence 官方通报了一个严重漏洞 CVE-2022-26134，远程攻击者在未经身份验证的情况下，可构造 OGNL 表达式进行注入，实现在 Confluence Server 或 Data Center 上执行任意代码。

利用范围

Confluence Server and Data Center >= 1.3.0

Confluence Server and Data Center < 7.4.17

Confluence Server and Data Center < 7.13.7

Confluence Server and Data Center < 7.14.3

Confluence Server and Data Center < 7.15.2

Confluence Server and Data Center < 7.16.4

Confluence Server and Data Center < 7.17.4

Confluence Server and Data Center < 7.18.1

环境搭建

使用 docker-compose 文件进行漏洞环境搭建

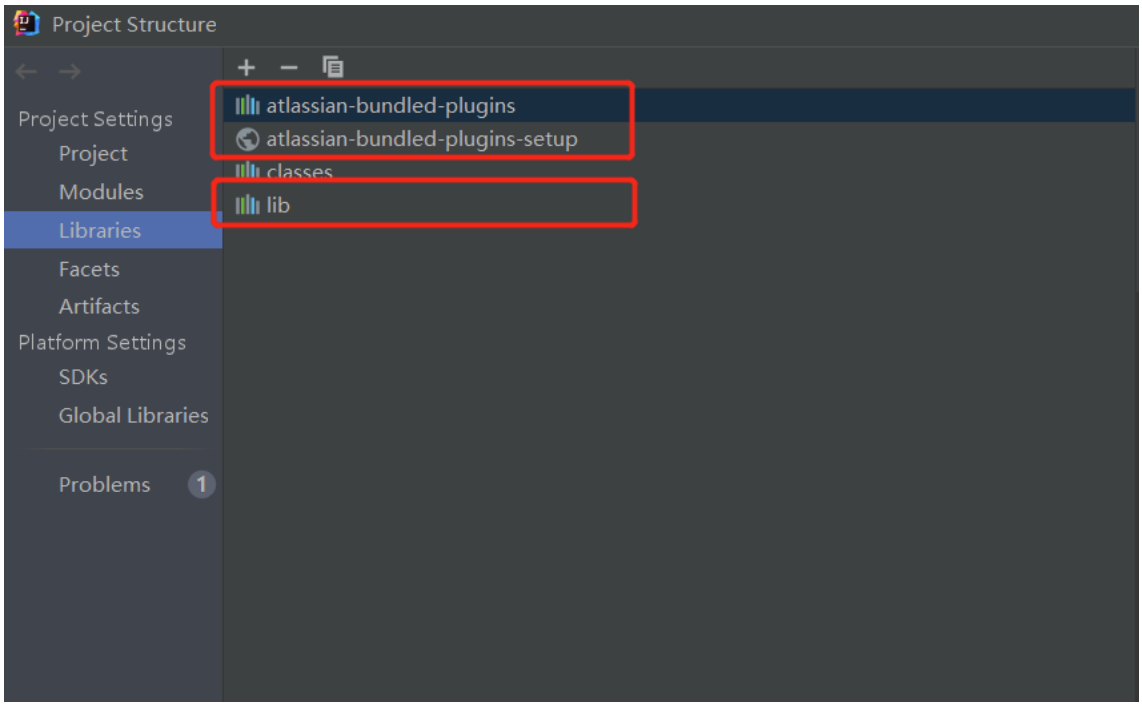
```
Plaintext
version: '2'
services:
  web:
    image: vulhub/confluence:7.13.6
    ports:
      - "8090:8090"
      - "5050:5050"
    depends_on:
      - db
```

```
db:
  image: postgres:12.8-alpine
  environment:
  - POSTGRES_PASSWORD=postgres
  - POSTGRES_DB=confluence
```

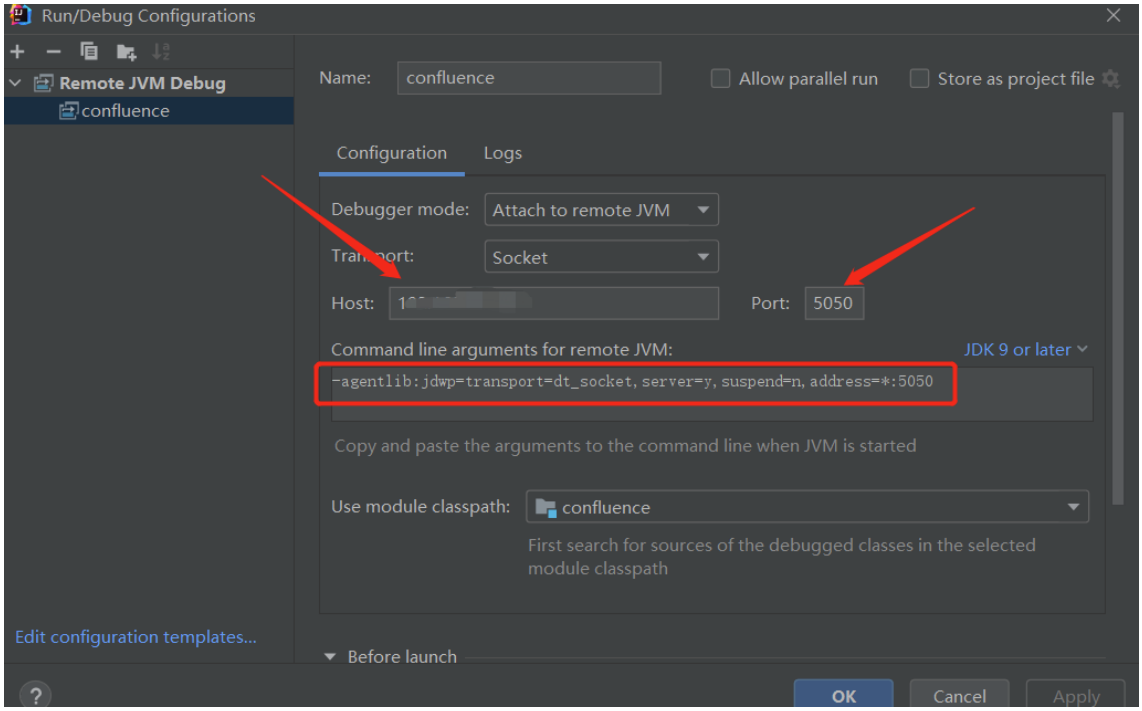
使用命令 `docker-compose up -d` 成功启动环境后，查看容器 id 等相关信息（5050 为远程调试端口）

进入容器后将/opt/atlassian 目录下的 confluence 源码使用 `docker cp` 命令复制到本地

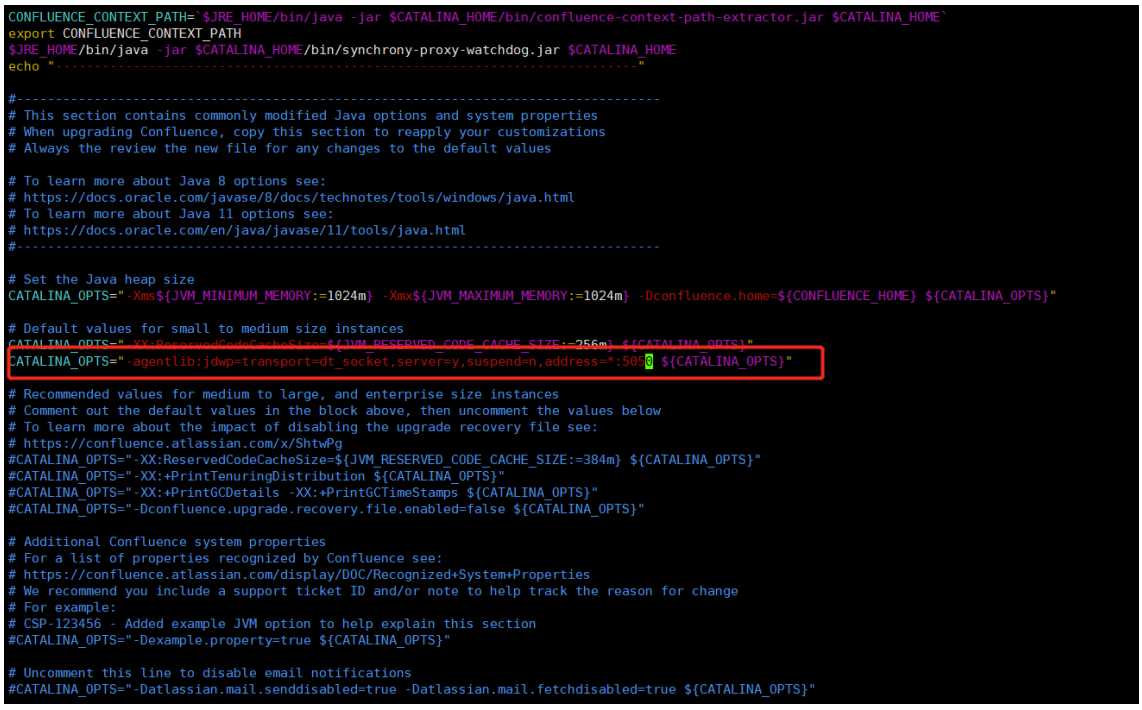
使用 IDEA 将/confluence/WEB-INF 下的 `atlassian-bundled-plugins`、`atlassian-bundled-plugins-setup`、`lib` 文件拉取为依赖文件



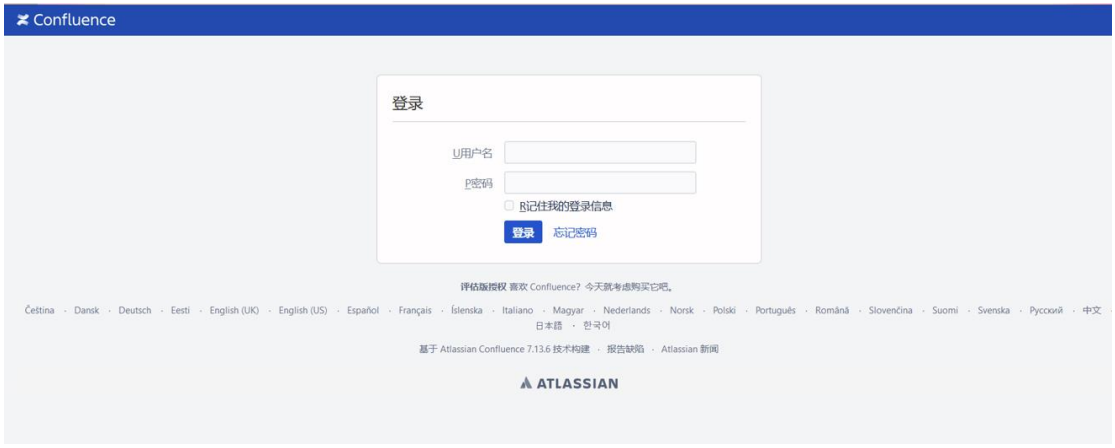
IDEA 中配置远程调试



在容器中/opt/atlassian/confluence/bin 目录下修改 setenv.sh 文件，加入远程调试配置



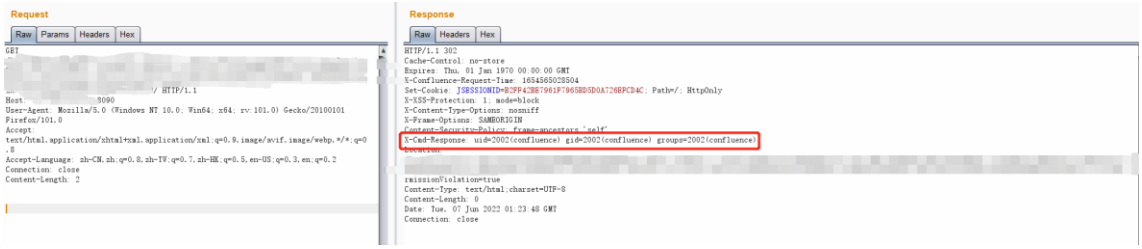
访问 http://IP:8090，进行相关配置（需要登录 confluence 官网进行证书申请）



漏洞分析环境成功搭建。

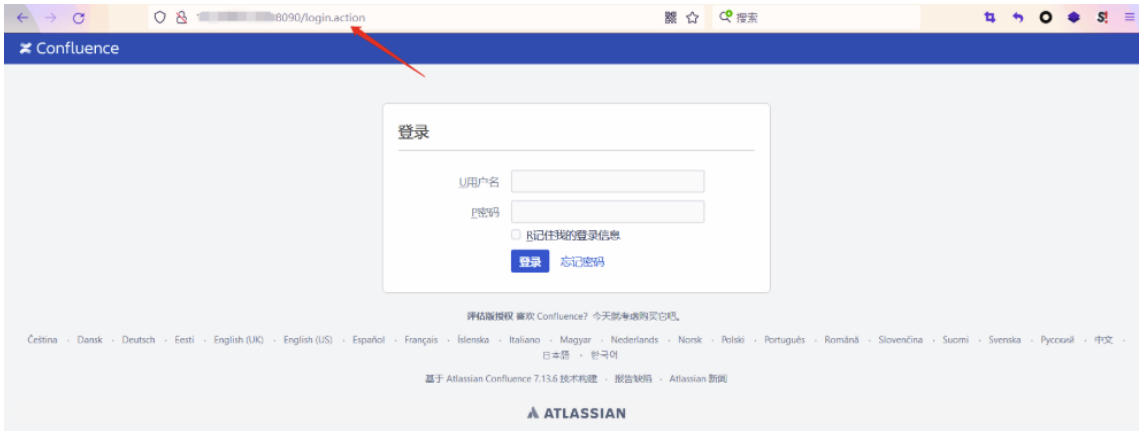
漏洞复现

成功命令执行



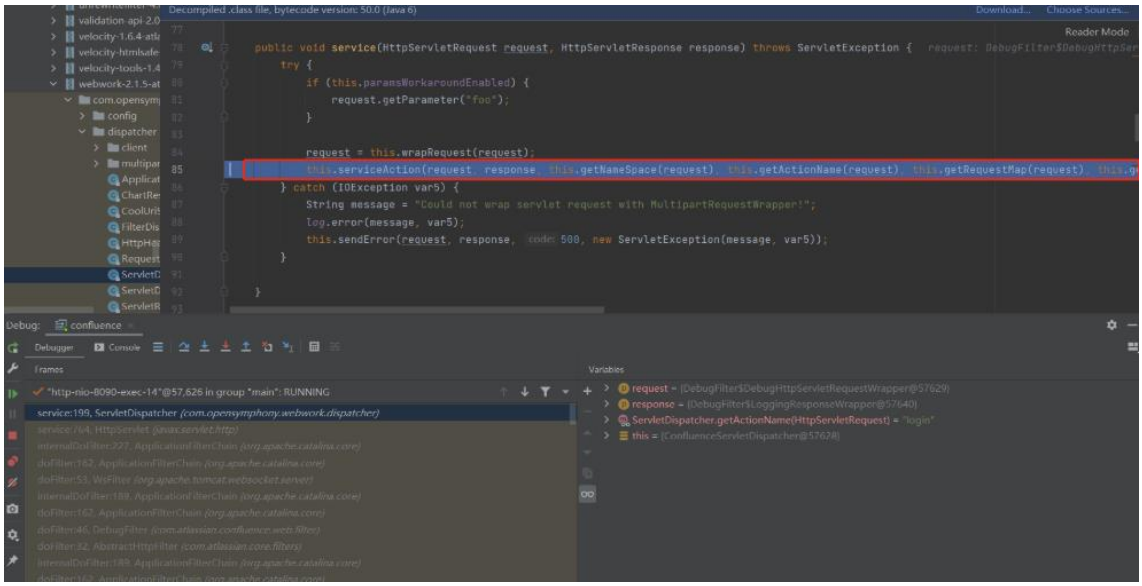
漏洞分析

以登录请求为例，对 confluence 请求处理流程进行调试分析

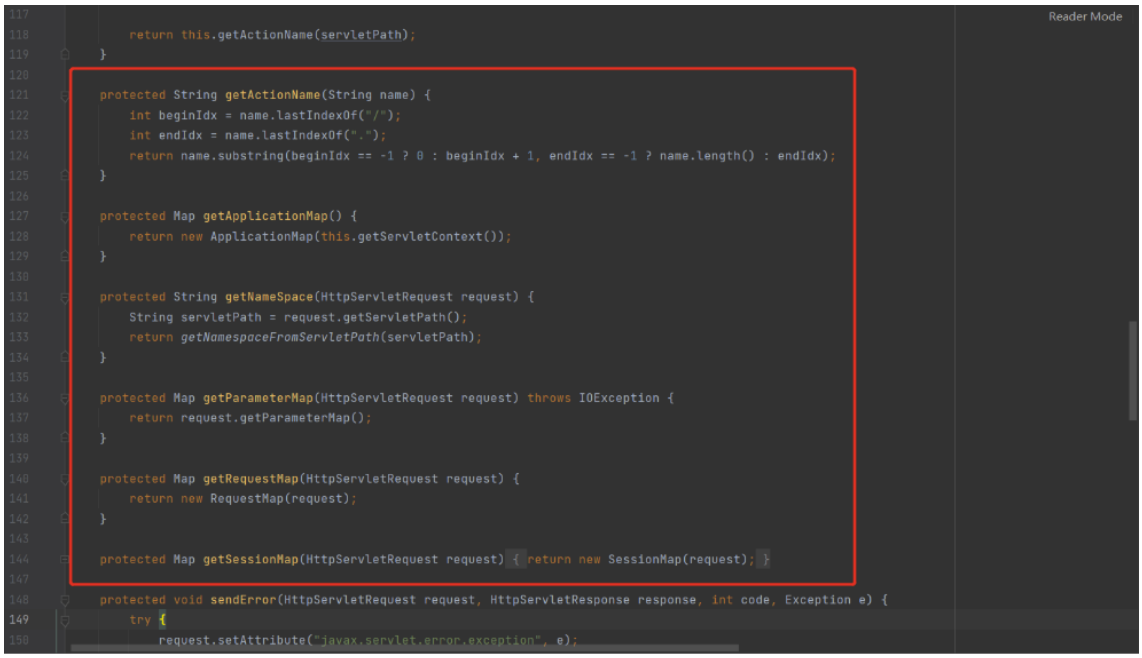


首先访问/login.action，经过一系列 Filter 处理后，将进入 Servlet 的分发器 ServletDispatcher

`com.opensymphony.webwork.dispatcher.ServletDispatcher`

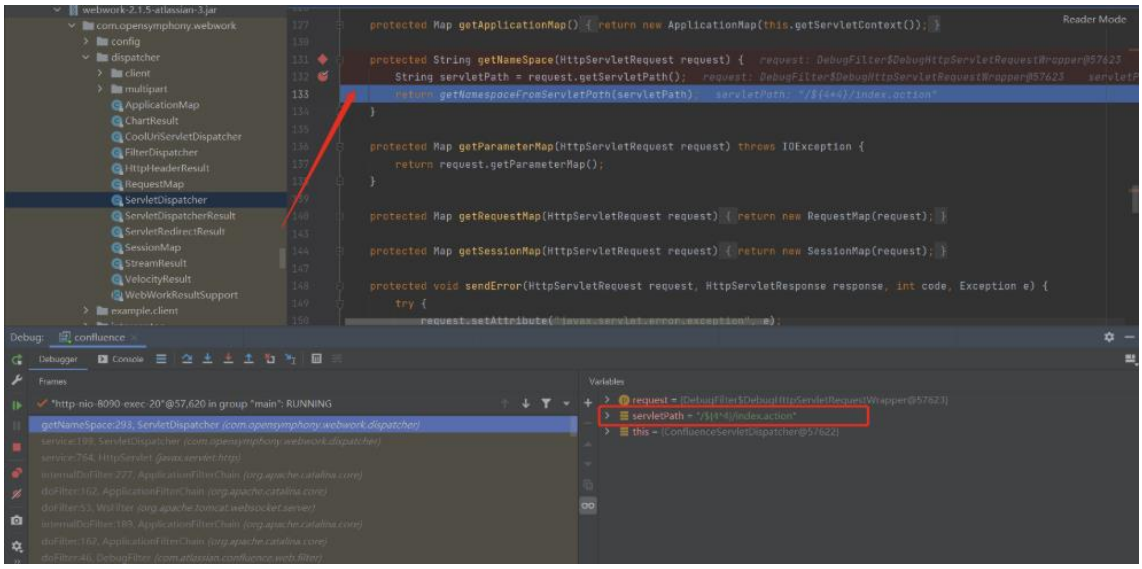


在 `ServletDispatcher` 中，通过 `getNameSpace` 、`getActionName` 、`getRequestMap` 、 `getSessionMap` 、 `getApplicationMap` 函数 ，分别去获取对应的值。（使用 payload=`/${4*4}/`）

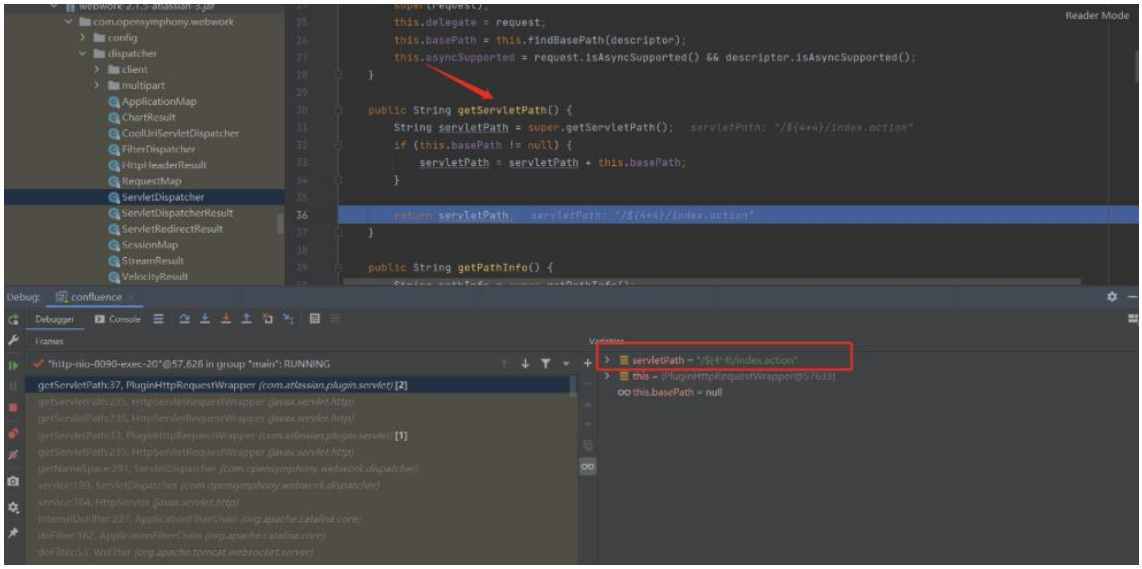


重点分析 `getNamespace` 函数，其对应的值为 `namespace`

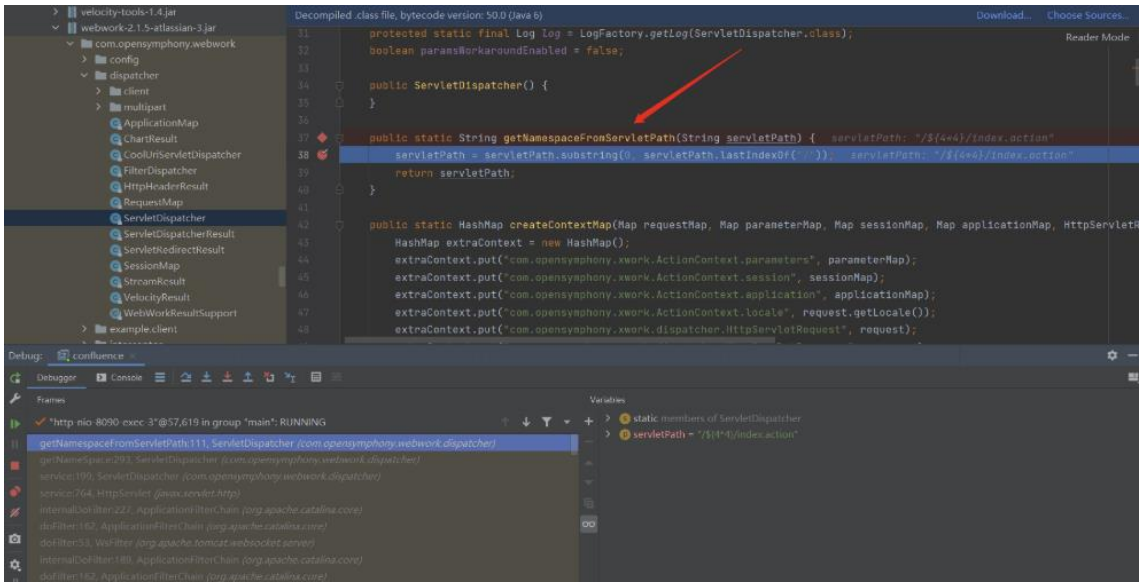
在 `com.opensymphony.webwork.dispatcher.ServletDispatcher#getNamespace` 中



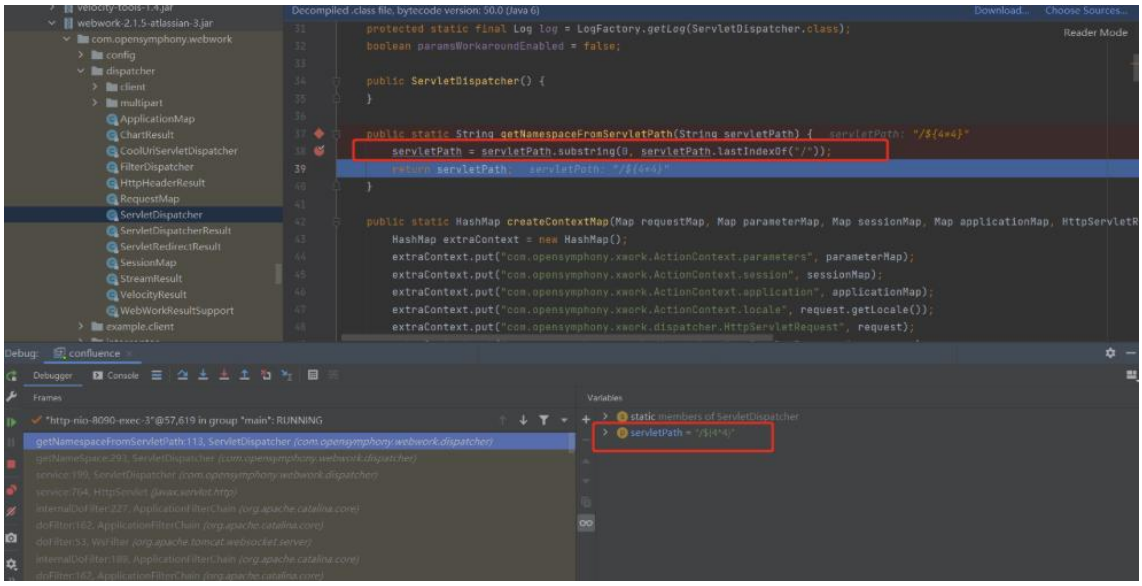
通过调用 `com.atlassian.plugin.servlet.PluginHttpRequertWrapper#getServletPath` 函数，获取了请求 `servletPath` 的值



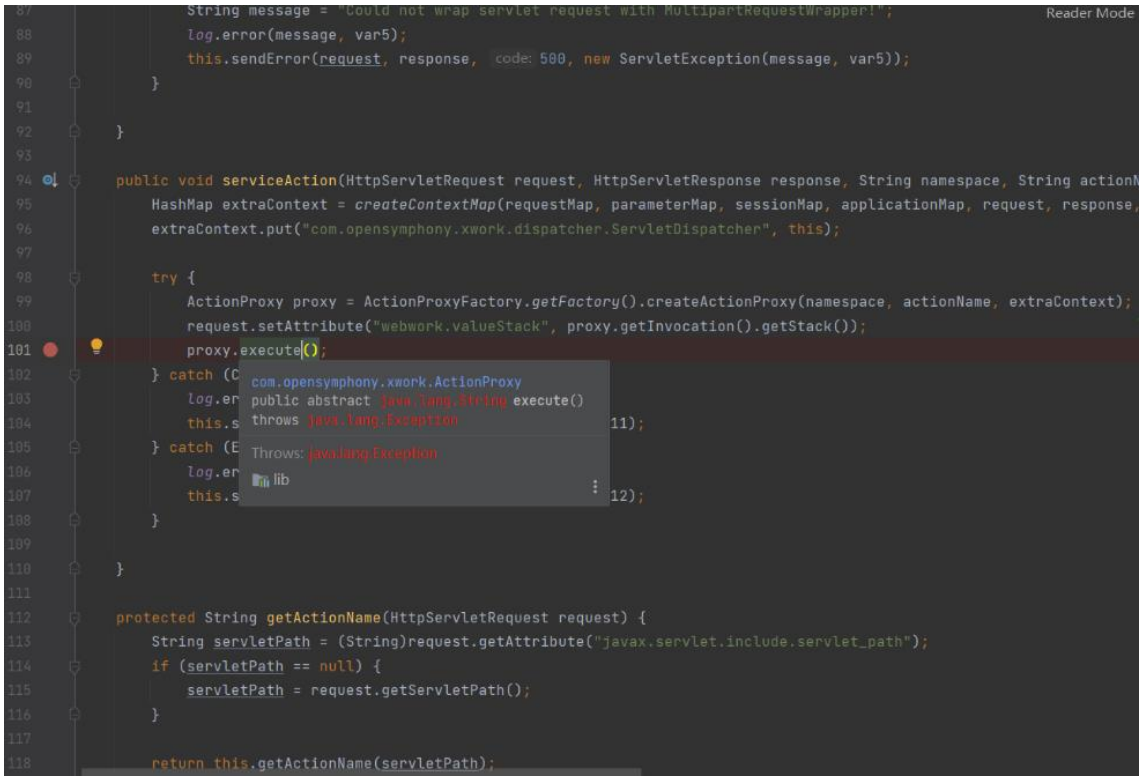
随后进入回到 `com.opensymphony.webwork.dispatcher.ServletDispatcher` 中进入 `getNamespaceFromServletPath` 函数



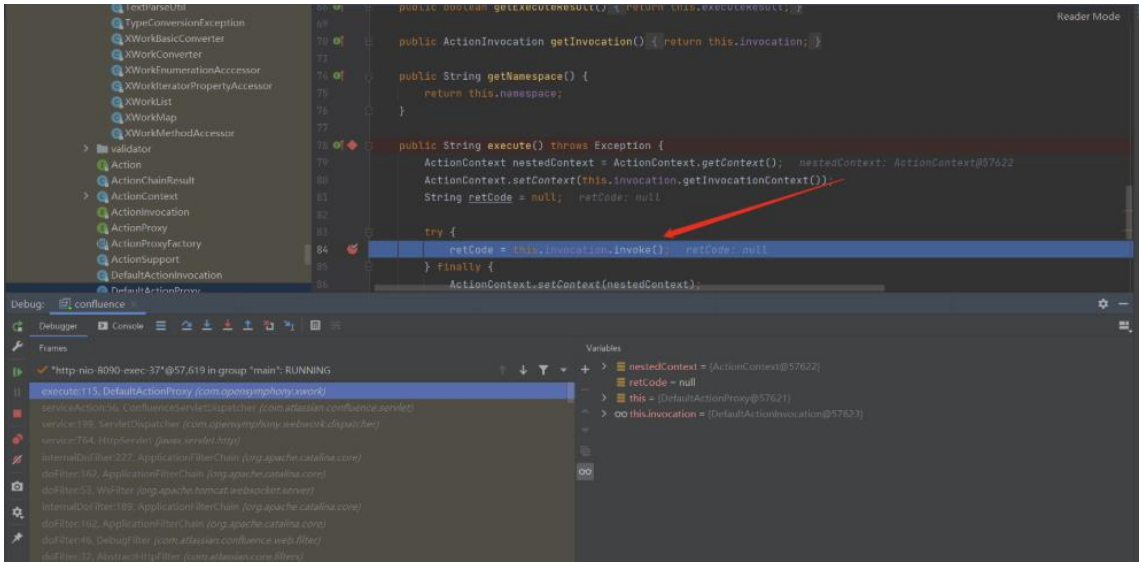
通过对字符串的截取 ， 使得 `namespace` 的取值为请求 `servletPath` 最后一个 `/` 之前的部分。



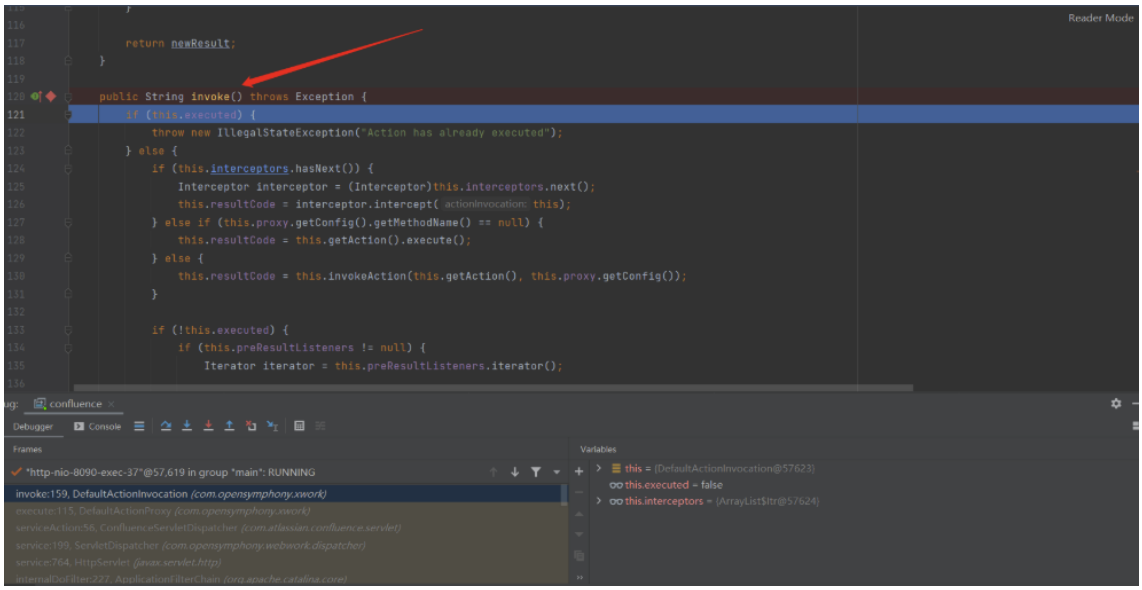
根据 `com.opensymphony.webwork.dispatcher.ServletDispatcher#serviceAction` 函数



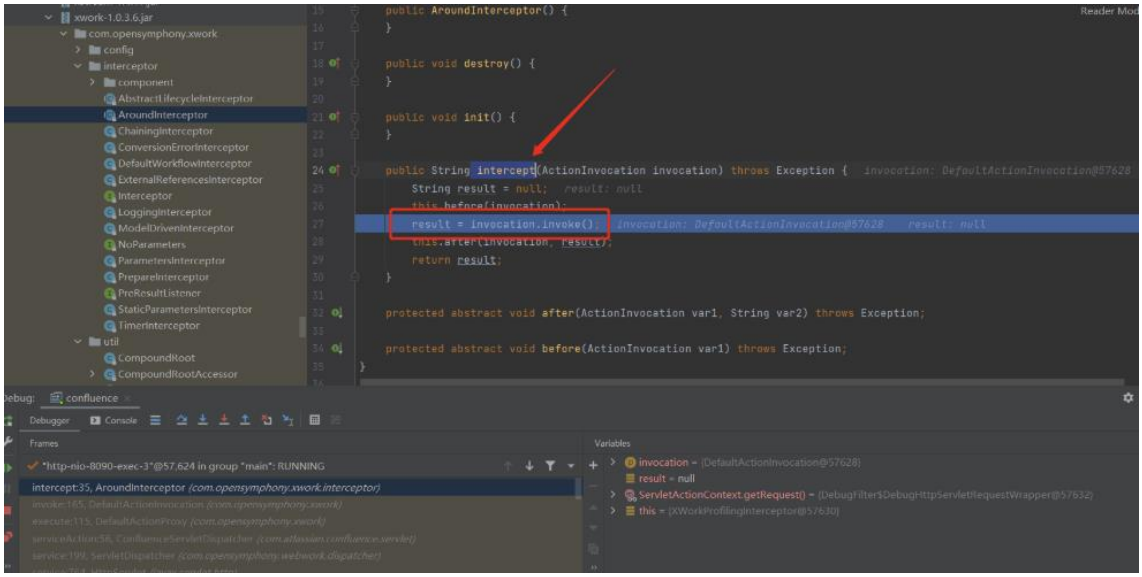
到 `com.opensymphony.xwork.DefaultActionProxy#execute`，实例化 `DefaultActionProxy` 对象，调用其 `execute` 函数



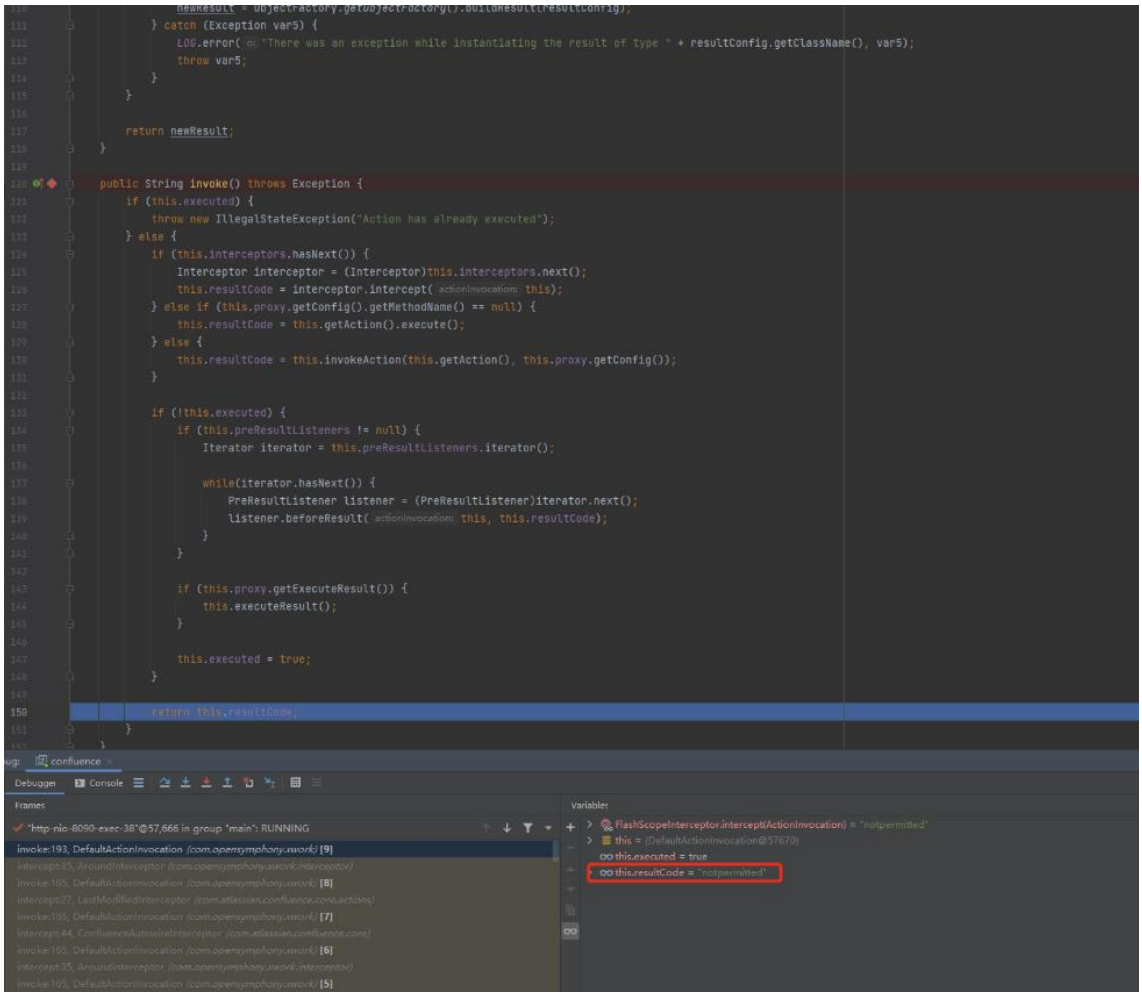
随后进入 `com.opensymphony.xwork.DefaultActionInvocation#invoke` 函数，通过 `Next` 获取拦截器对象



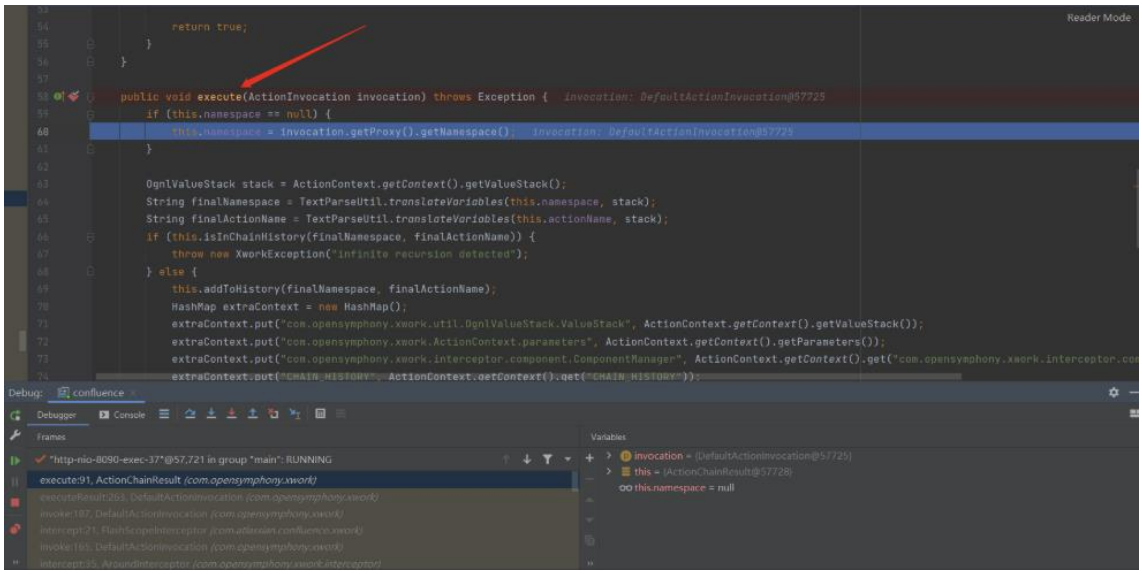
继续跟进来到 `com.opensymphony.xwork.interceptor.AroundInterceptor#intercept`，发现 `invoke` 函数通过调用 `intercept` 方法，形成迭代循环



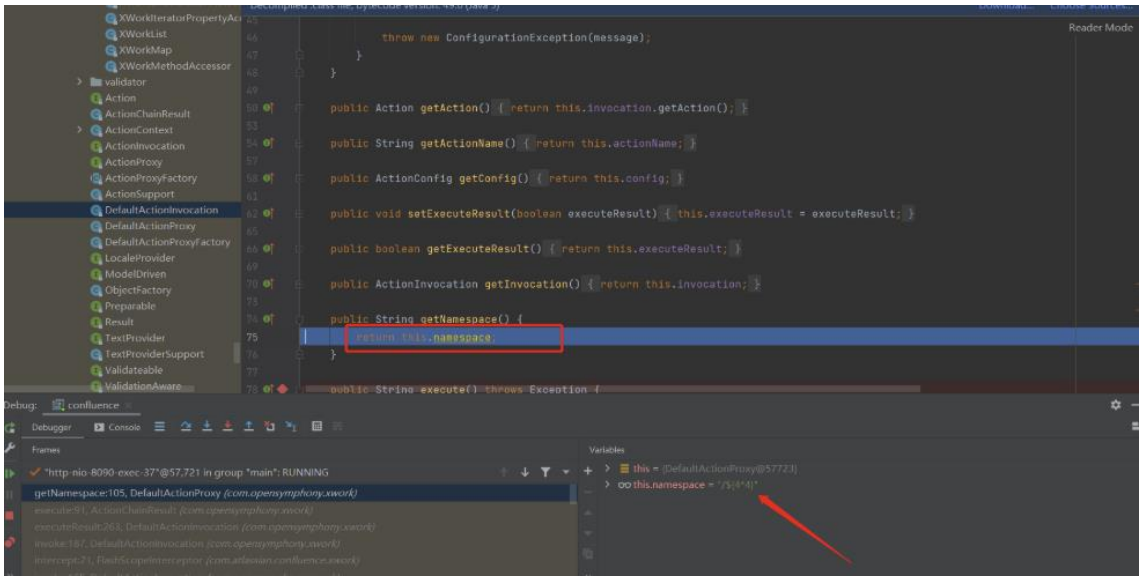
通过不断调试，发现在满足一定条件之后，将不会继续调用 `invoke` 函数，而是将返回 `notpermitted` 并赋值给 `resultCode`，跳出循环。



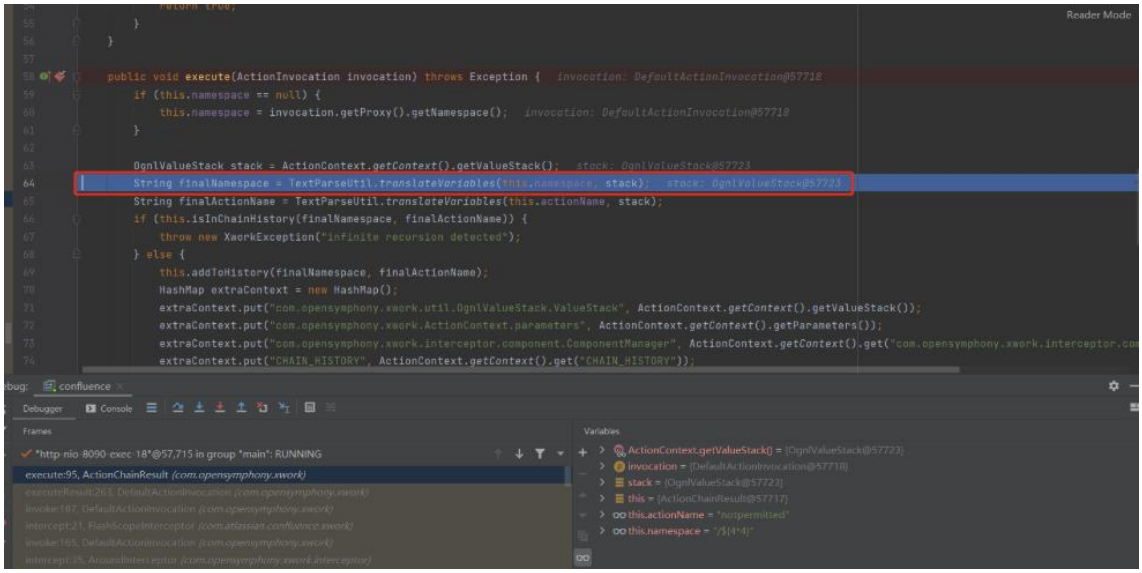
随后进入 `com.opensymphony.xwork.ActionChainResult#execute` 函数



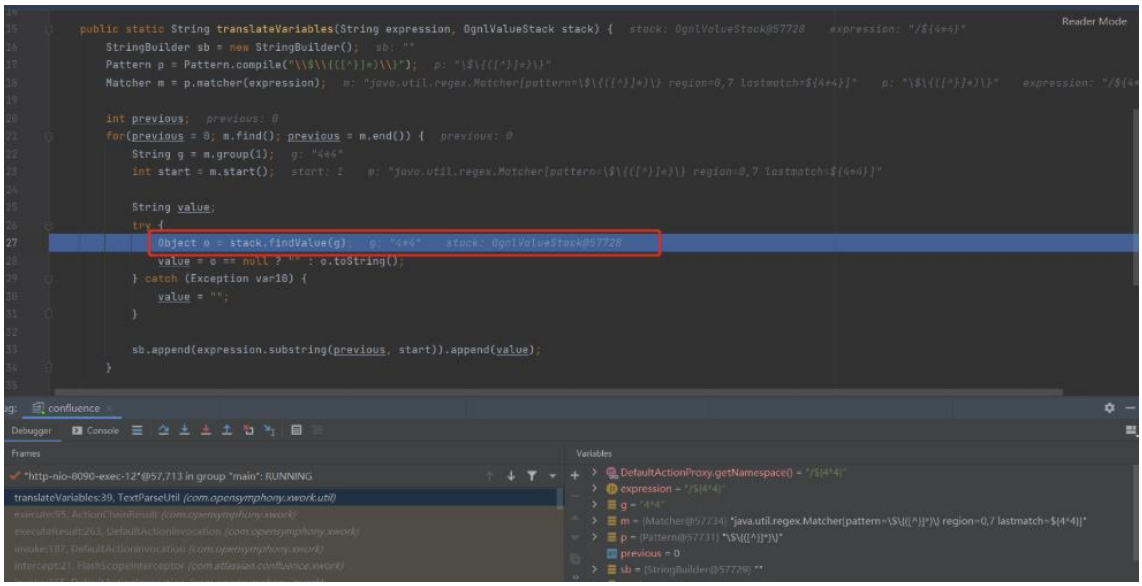
这里通过 `getNamespace` 获取 `namespace` 的值，前面我们已经分析出来：`namespace` 的取值为请求 `servletPath` 最后一个`/`之前的部分



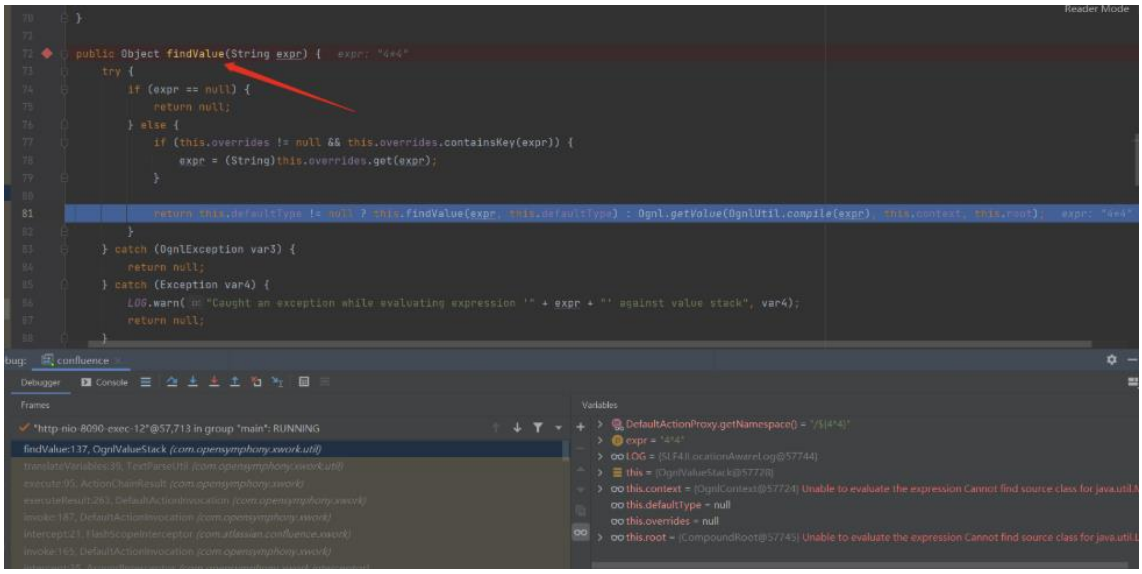
继续跟进，在 `com.opensymphony.xwork.ActionChainResult` 中，会调用 `translateVariables` 函数对 `OGNL` 表达式进行解析，表达式的值为 `namespace` 的取值



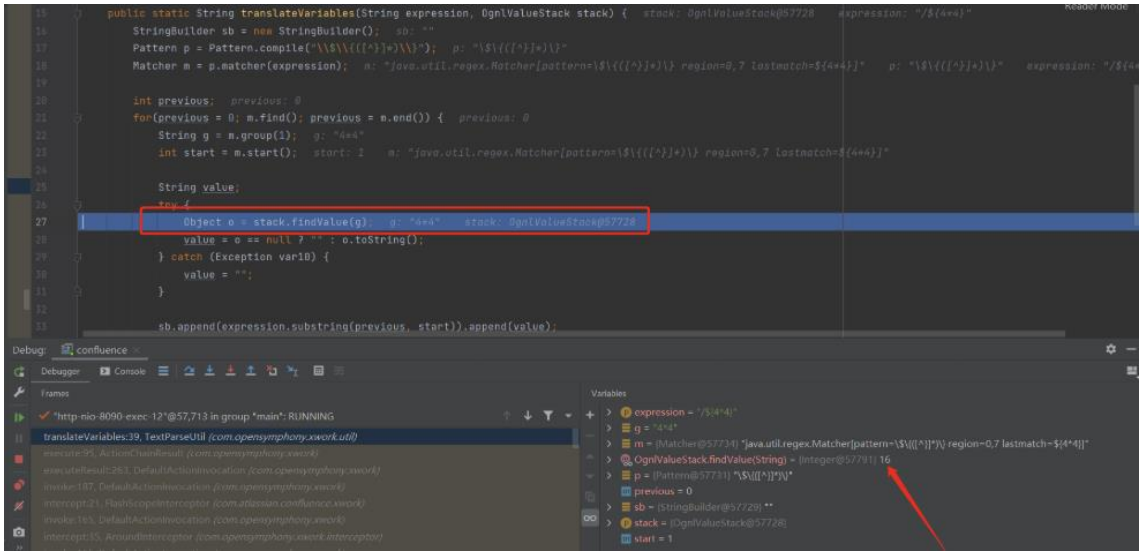
在 `com.opensymphony.xwork.util.TextParseUtil#translateVariables` 中



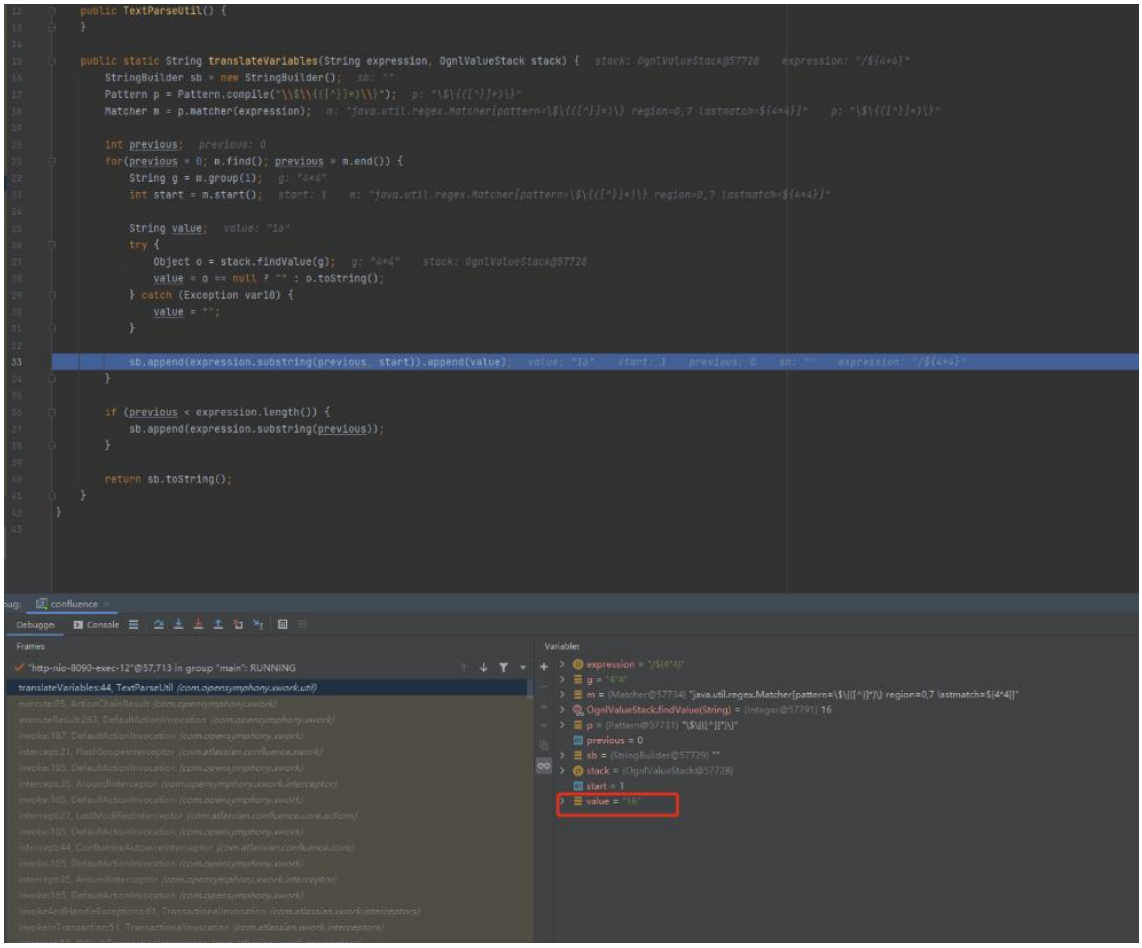
调用了 `com.opensymphony.xwork.util.OgnlValueStack#findValue` 函数对表达式进行解析



一系列操作之后，`findValue` 返回的字符串为 `16`



成功触发 `OGNL` 表达式注入



修复建议

建议升级到 Atlassian Confluence Server and Data Center 至安全版本。

下载链接:

https://www.atlassian.com/software/confluence/download-archives

沙箱绕过

从 v7.15 系列开始，Confluence 在 OGNL 表达式解析时加入了沙箱限制，采取了黑名单、白名单等方式。

绕过的技巧：白名单（限制为静态方法调用）中，仍然有一些静态方法可绕过沙箱限制或者沙箱本身逻辑上也存在相关缺陷可实现方法调用。