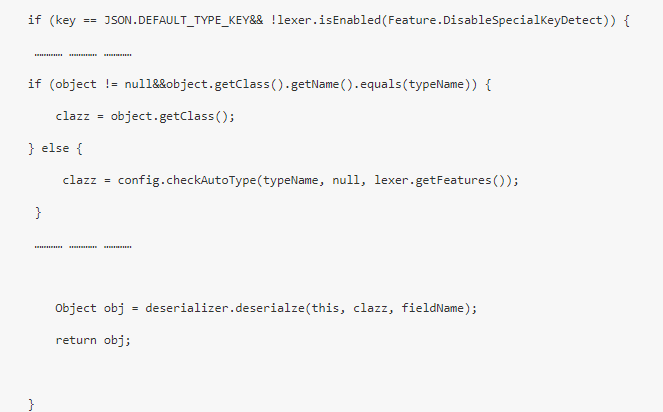
目录

[分析 1](#_Toc35434858)

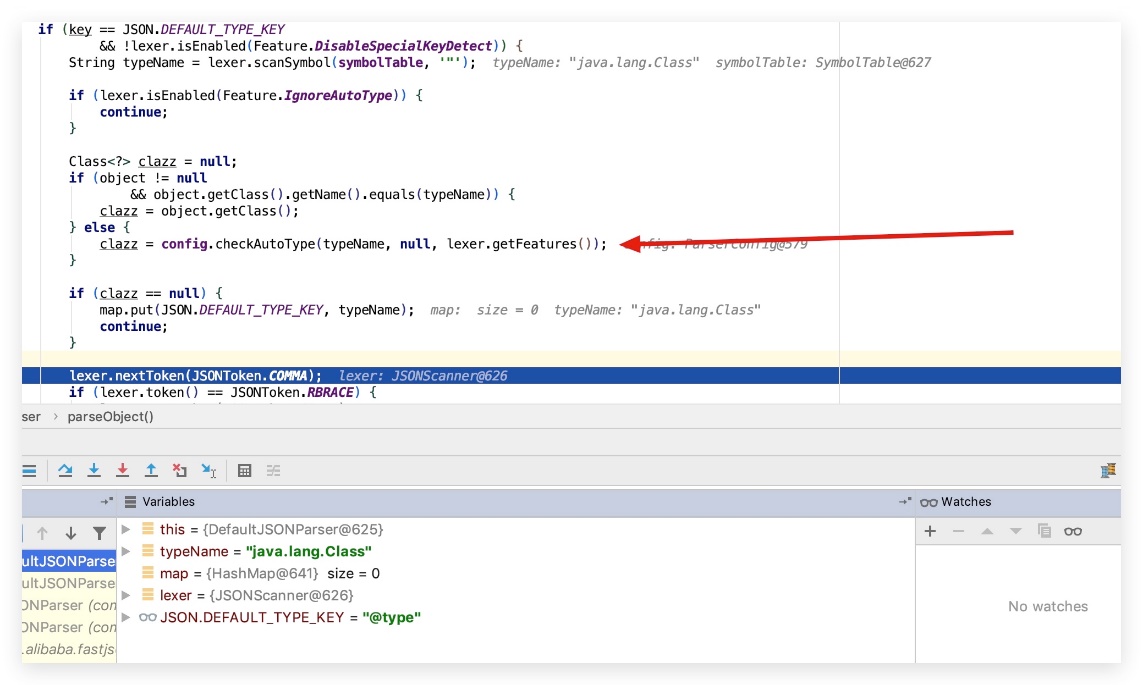
## 分析

1.调用过程和之前的 《fastjson jndi利用方式》 差不多，这边使用了一个特性绕过了黑名单机制，在com.alibaba.fastjson.parser.DefaultJSONParser#parseObject(java.util.Map, java.lang.Object)执行逻辑中:

首先遇到的是第一个key@type，然后进行了以下的判断，如果是@type并且启用了特殊key检查的话，那么就把对应的value作为类来加载。这边摘取片段来进行展示。



fastjson会去检测@type的类是否为黑名单中的类，



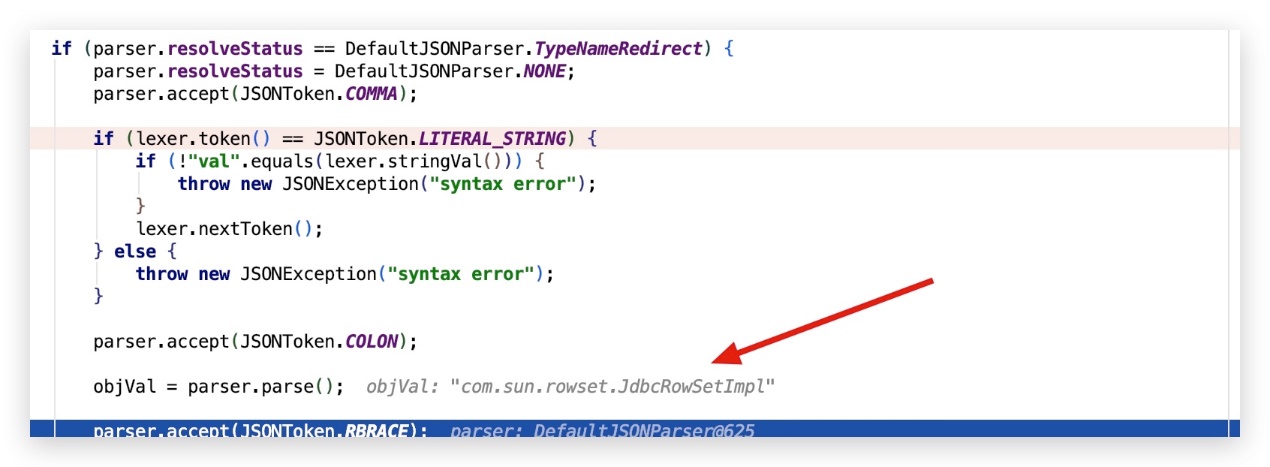
而poc中传入的@type为java.lang.class并非黑名单中的类，所以第一步检测的通过的。

#### 2. 接下来会把对应的value进行加载，也就是加载java.lang.class



跟进deserialze方法（com.alibaba.fastjson.serializer.MiscCodec#deserialze）





可以看到lexer中的stringVal为poc中的val,而val的值为com.sun.rowset.JdbcRowSetImpl.

3. 接下来将objVal赋值给strVal



然后执行下面一大串if判断，其中有个if为：

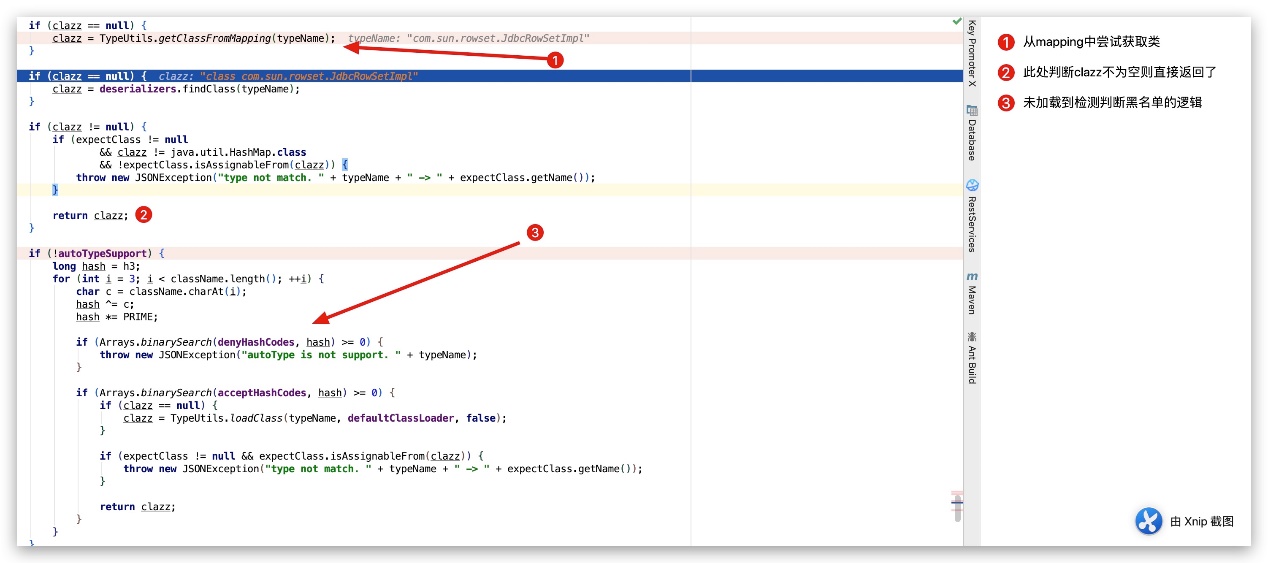
如果传入的clazz为java.lang.class,则会调用TypeUtils.loadClass加载com.sun.rowset.JdbcRowSetImpl类，



跟进loadClass方法



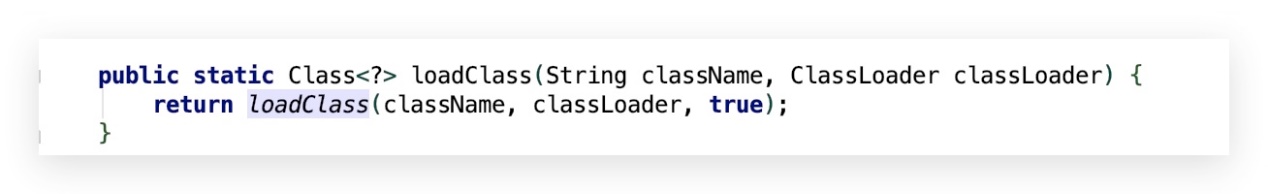
从而导致checkAutoType在检测是否为黑名单的时候绕了过去，因为上一步将com.sun.rowset.JdbcRowSetImpl放入了mapping中，checkAutoType中使用TypeUtils.getClassFromMapping(typeName)去获取class不为空，从而绕过了黑名单检测。



导致将com.sun.rowset.JdbcRowSetImpl放入mapping中的问题点是在loadClass中的第三个参数，该参数是指是否对class放入缓存mapping中。



1.2.47版本中的代码



1.2.48版本中的代码

