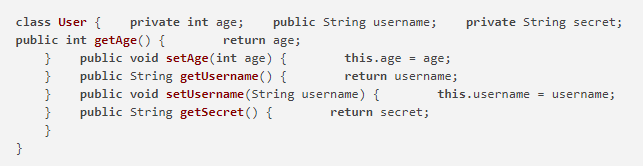
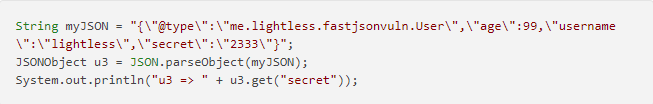
## 分析

在分析漏洞之前，我们先看下这个库都有什么样的功能，我们先创建一个User对象：



我们主要关注一下从JSON还原回Object的方法，主要的API有两个，分别是JSON.parseObject和JSON.parse，最主要的区别就是前者返回的是JSONObject而后者返回的是实际类型的对象，当在没有对应类的定义的情况下，通常情况下都会使用JSON.parseObject来获取数据。

fastjson接受的JSON可以通过@type字段来指定该JSON应当还原成何种类型的对象，在反序列化的时候方便操作。

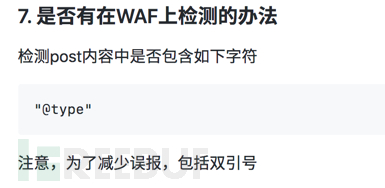


如果需要还原出private成员的话，还需要加上Feature.SupportNonPublicField：



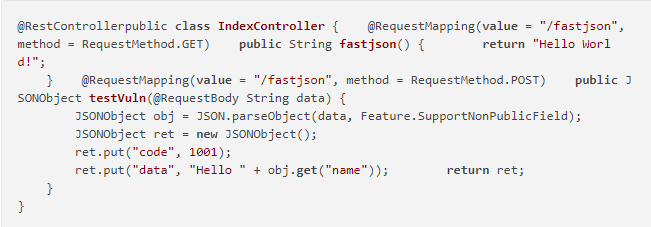
## 跟踪分析

1.根据官方的公告中的WAF检测方法来看，问题很有可能是因为反序列化了任意类型的class从而导致的RCE。

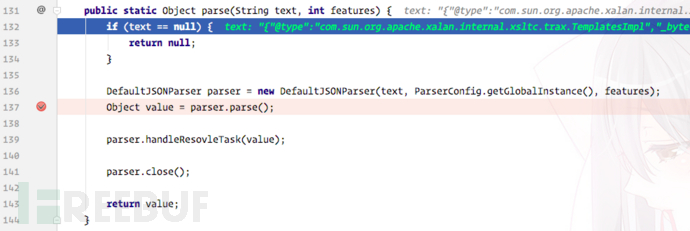


从网上找到的payload中也能看出利用的是TemplatesImpl来执行的命令，这个在之前的JDK7u21中已经分析过了，可能还存在其他的执行命令方法，这些我们暂且不谈，主要来看fastjson的部分。

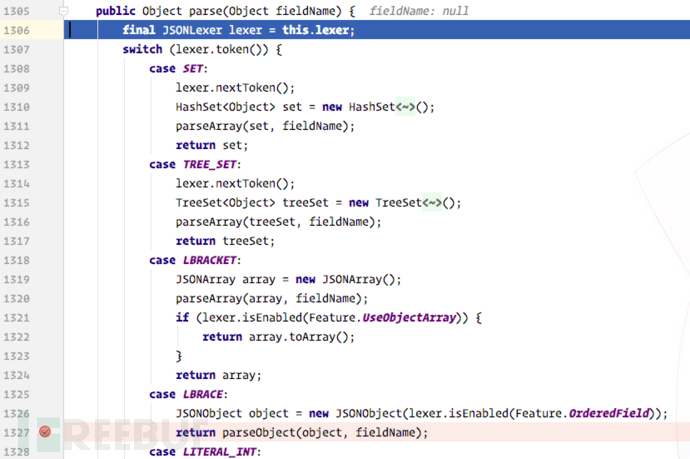
2．我们搭建一个简单的Web应用来接受用户POST过来的JSON并且进行反序列化：



3.传入我们带有@type类型的JSON字符串并且开始调试。在JSON.parseObject处下断并开始向下跟。



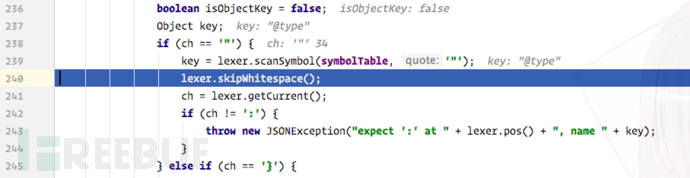
4.一开始会跟进到JSON.parse方法，并且调用了parse()方法继续进行JSON格式的匹配。继续跟进parser.parse()方法。



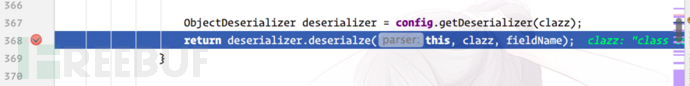
5.到了这里之后，开始依次进行JSON的解析，我们传入的第一个字符是{，所以进入LBRACE这个分支中，并继续进入parseObject(object, fieldName)方法来解析对象。

https://image.3001.net/images/20180719/15319726055314.png!small

6.这个时候lexer所在的字符为”，会进入下面这个分支继续解析JSON字符串，通过scanSymbol方法获取到双引号之间的字符串也就是@type



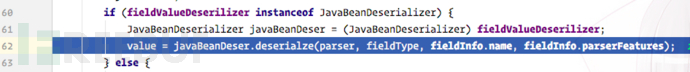
7.之后会获取@type字段的值，并且尝试获取这个类的Class，经过一系列的判断后，调用了deserializer.deserialize(this, clazz, fieldName)方法进行反序列化。



（🡨!一开始没找到sortedFieldDeserializers是在什么地方生成的，想仔细跟一下代码，于是就从头到尾的看了下getDeserializer(clazz)的部分，发现是在这里生成的。这个函数的目的是获取一个可以反序列化我们通过@type指定的类的deserializer，由于预定于的列表中没有，于是会继续调用createJavaBeanDeserializer()来生成一个，实际上是调用了ParseConfig.build()方法，其中会通过反射机制获取我们指定类的一些信息，通过对method进行一些过滤，猜测出getter和setter并推出一些可能存在的field!🡪）

8.紧接着就依次处理JSON字符串中的各个字段，当匹配到payload中的\_tfactory字段的时候，由于我们传入的JSON字符串中是一个空的对象，进入parseField方法后，继续向下跟就会调用到JavaBeanDeserializer.deserialize()方法，在这里会为\_factory创建一个TransformerFactoryImpl对象并赋值。

9.同样的，\_outputProperties字段我们也是传入了一个空对象，会进入和上面相同的流程，仔细分析一下这段代码，这里是触发命令执行的关键部分。



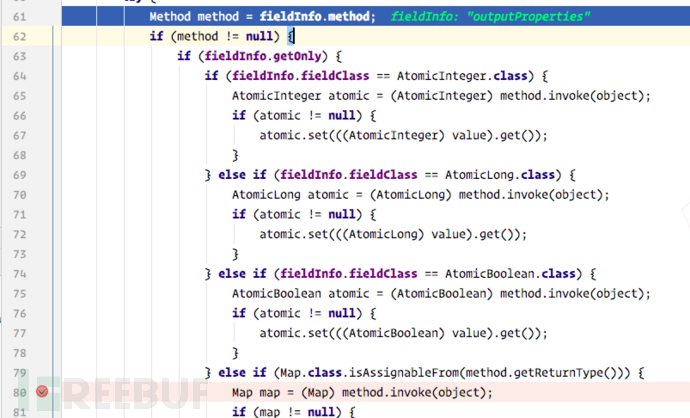
首先依然会进入parseField方法



10.进入之后会调用smartMatch(key)方法，这个方法的主要作用是进行一些『智能匹配』，方便后续获取对应变量的getter和setter。调用后这个方法会去掉字符串中的-、删除开头的下划线等，所以当我们传入了\_outputProperties的时候，实际上就给处理成了outputProperties，并返回对应的FieldDeserializer对象，之后就会调用该对象的parseField方法。进入该方法后，就会调用setValue(object, value)方法，继续跟进。



跟进之后显而易见，getOutputProperties被调用了



11. 然后就会执行我们在\_bytecodes构造的恶意字节码，造成命令执行。