# JAVA入门到放弃系列之Fastjson反序列化(一)

源自面试的Java拷打，各位可以早做准备：

“有没调过Java的漏洞"

"Jdk不同版本如何绕过"

“说说Java在版本迭代过程中漏洞攻击面的变化"

"当你拿到一份java代码，你会如何审计"

“说说各个版本的fastjson漏洞"

……

## 一、概述

fastjson通过parse、parseObject处理以json结构传入的类的字符串形时，会默认调用该类的共有setter与构造函数，并在合适的触发条件下调用该类的getter方法。当传入的类中setter、getter方法中存在利用点时，攻击者就可以通过传入可控的类的成员变量进行攻击利用。

## 二、fastjson一些要点

- `JSON.parse(jsonString)` 和 `JSON.parseObject(jsonString, Target.class)`两者调用链一致，前者通过解析 jsonString 字符串获取 `@type` 指定的类，后者则会直接使用参数中的Target.class。
- 而`JSON.parseObject(jsonString)` 将会返回 JSONObject 对象，且类中的所有 getter 与setter 都被调用。区别在于其在最后执行了一次JSON.toJSON。
- fastjson 在创建一个类实例时会通过反射调用类中符合条件的 getter/setter 方法，其中 getter 方法需满足条件：方法名长于 4、不是静态方法、以 `get` 开头且第4位是大写字母、方法不能有参数传入、继承自`Collection|Map|AtomicBoolean|AtomicInteger|AtomicLong`、此属性没有 setter 方法；setter 方法需满足条件：方法名长于 4，以 `set` 开头且第4位是大写字母、非静态方法、返回类型为 void 或当前类、参数个数为 1 个。
- 如果目标类中私有变量没有 setter 方法，但是在反序列化时仍想给这个变量赋值，则需要使用`Feature.SupportNonPublicField` 参数。(TemplatesImpl利用链中无setter方法的私有变量`_tfactory`以及`_name`赋值及用到这个知识点)

## 三、fastjson版本更迭的漏洞利用分析

1.2.24(首个漏洞)-->1.2.41(黑名单绕过)-->1.2.42(黑名单绕过)-->1.2.45(新利用链)-->1.2.47(AutoType绕过)-->1.2.68(AutoType绕过)

**1.2.24(首个漏洞)**

Fastjson通过parseObject/parse将传入的字符串反序列化为Java对象时没有进行合理检查。

**1.2.41(黑名单绕过)**

引入了checkAutotype安全机制

- 是否是白名单中的类

- 是否在反序列化cache中(在mappings列表)
- 类有JSONType注解(如:fastjson.annotation.JSONType)

如果在恶意类前后加上"L"与";"，例如"LAutoTypeTest.ForTest;"，这样不仅可以轻易的躲避黑名单，随后在程序执行到这里时，还会将首尾附加的"L"与";"剥去。剥皮处理之后字符串变成"AutoTypeTest.ForTest"，接着"AutoTypeTest.ForTest"被loadClass加载，恶意类被成功反序列化利用。

**1.2.42(黑名单绕过)**

程序并不是直接通过明文的方式来匹配黑白名单，而是采用了一定的加密混淆。通过分析其代码发现第一个if分支用来限制传入的类名长度的，只要传入的poc中类名长度在3与128之间即可绕过。而第二个分支，会去掉首尾"L"与";"再放入原来的黑名单逻辑进行判断，我们进行双写即可绕过。

总结一下：指定长度，双写绕过。

**1.2.45(新利用链)**

@type指定了JndiDataSourceFactory类，而在properties属性中的data_source变量中指定恶意数据源。由于`JndiDataSourceFactory`并不在黑名单上，因此可以顺利通过黑名单校验，在接下来的反序列化过程中，在为Properties变量赋值时调用其setter方法。

**1.2.47(AutoType绕过)**

可以在不开启 AutoTypeSupport 的情况下进行反序列化的利用。

存在一个逻辑问题：`autoTypeSupport`为`true`时，fastjson 也会禁止一些黑名单的类反序列化，但是有一个判断条件：当反序列化的类在黑名单中，且 `TypeUtils.mappings` 中没有该类的缓存时，才会抛出异常。此时我们可以通过将恶意的 val 加载到 mappings 中，再次以恶意类进行 `@type` 请求时即可绕过黑名单进行的阻拦。

**1.2.68(AutoType绕过)**

版本 1.2.68 本身更新了一个新的安全控制点 safeMode，如果应用程序开启了 safeMode，将在 `checkAutoType()` 中直接抛出异常，也就是完全禁止 autoType。不过在 `checkAutoType()` 函数中有这样的逻辑：如果函数有 `expectClass` 入参，且我们传入的类名是 `expectClass` 的子类或实现，并且不在黑名单中，就可以通过 `checkAutoType()` 的安全检测。

本章简单地归纳对各个版本的fastjson漏洞原理，应付一下面试，之后会进一步对其复现和代码调试分析。

**参考资料：**

[1]https://www.javasec.org/java-vuls/FastJson.html

[2]https://paper.seebug.org/1274/

[3]https://paper.seebug.org/1319/

[4]https://paper.seebug.org/1343/