链滴

Tomcat AJP 文件包含漏洞（CVE-2020-1938） 解决方案 （含 Apache ajp 方式）

作者：TravelEngineers

原文链接：https://ld246.com/article/1582646011284

来源网站：链滴

漏洞原理：

Tomcat 配置了两个 Connecto，它们分别是 HTTP 和 AJP ： HTTP 默认端口为 8080，处理 http 请，而 AJP 默认端口 8009，用于处理 AJP 协议的请求，而 AJP 比 http 更加优化，多用于反向、集群，漏洞由于 Tomcat AJP 协议存在缺陷而导致，攻击者利用该漏洞可通过构造特定参数，读取服务器 ebapp 下的任意文件以及可以包含任意文件，如果有某上传点，上传图片马等等，即可以获取 shell。

漏洞版本：

Apache Tomcat 6

Apache Tomcat 7 < 7.0.100

Apache Tomcat 8 < 8.5.51

Apache Tomcat 9 < 9.0.31

漏洞利用：

https://github.com/0nise/CVE-2020-1938
https://github.com/nibiwodong/CNVD-2020-10487-Tomcat-ajp-POC
https://github.com/Kit4y/CNVD-2020-10487-Tomcat-Ajp-lfi-Scanner
https://github.com/YDHCUI/CNVD-2020-10487-Tomcat-Ajp-lfi/

漏洞修复（官方方案）：

1. 如未使用 Tomcat AJP 协议：

如未使用 Tomcat AJP 协议，可以直接将 Tomcat 升级到 9.0.31、8.5.51 或 7.0.100 版本进行漏洞复。

如无法立即进行版本更新、或者是更老版本的用户，建议直接关闭 AJPConnector，或将其监听地址为仅监听本机 localhost。

具体操作：

（1）编辑 <CATALINA_BASE>/conf/server.xml，找到如下行（<CATALINA_BASE> 为 Tomcat 工作目录）：

```
<Connector port="8009"protocol="AJP/1.3" redirectPort="8443" />
```

（2）将此行注释掉（也可删掉该行）：

```
<!--<Connectorport="8009" protocol="AJP/1.3"redirectPort="8443" />-->
```

（3）保存后需重新启动，规则方可生效。

2. 如果使用了 Tomcat AJP 协议：

建议将 Tomcat 立即升级到 9.0.31、8.5.51 或 7.0.100 版本进行修复，同时为 AJP Connector 配置 scret 来设置 AJP 协议的认证凭证。例如（注意必须将 YOUR_TOMCAT_AJP_SECRET 更改为一个安性高、无法被轻易猜解的值）：

```
<Connector port="8009"protocol="AJP/1.3" redirectPort="8443"address="YOUR_TOMCAT_I
_ADDRESS" secret="YOUR_TOMCAT_AJP_SECRET"/>
```

如无法立即进行版本更新、或者是更老版本的用户，建议为 AJPConnector 配置 requiredSecret 来置 AJP 协议认证凭证。例如（注意必须将 YOUR_TOMCAT_AJP_SECRET 更改为一个安全性高、无被轻易猜解的值）：

```
<Connector port="8009"protocol="AJP/1.3" redirectPort="8443"address="YOUR_TOMCAT_I
_ADDRESS"requiredSecret="YOUR_TOMCAT_AJP_SECRET" />
```

漏洞修复（Apache ajp版本）：

想什么呢，站长已经放弃了，apache2源码没找到怎么配置。

还在使用ajp配置站长此处只有2个建议，

1.**换成Nginx吧**，有多快就多块。

2.**变更Ajp方式为Http,并实现类似ip_hash方式session保持。**此处主要讲第2种方式怎么配置，相信定能解救你于水火。

Apache2使用Http实现类似ip_hash机制session软负载：

1.启用模块

打开httpd.conf启动如下模块：

```
LoadModule headers_module modules/mod_headers.so
```

2.在conf目录下新增配置文件balance.conf，内容如下：

```
#提供基础的代理功能
LoadModule proxy_module modules/mod_proxy.so
#提供负载均衡的功能
LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
#代理http协议
LoadModule proxy_http_module modules/mod_proxy_http.so

#负载均衡的算法模块
LoadModule lbmethod_byrequests_module modules/mod_lbmethod_byrequests.so
LoadModule slotmem_shm_module modules/mod_slotmem_shm.so
#兼容低版本访问
LoadModule access_compat_module modules/mod_access_compat.so

ProxyRequests Off

#启用类似ip_hash机制配置
ProxyPreserveHost on
Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/" env=BALANC
R_ROUTE_CHANGED

#代理关联配置loadfactor可以分发请求权重，loadfactor越大，权重越大
<Proxy balancer://mycluster>
```

```
  BalancerMember http://localhost:8001 loadfactor=1 route=tomcat7_1
  BalancerMember http://localhost:8002 loadfactor=1 route=tomcat7_2

  #热部署，当着备份服务，当tomcat7_1和tomcat7_2死掉的时候，就自动访问tomcat7_3
  #BalancerMember http://localhost:9080 loadfactor=1 route=tomcat7_3  status=+H
</Proxy>
#启用类似ip_hash机制配置
ProxyPass / balancer://mycluster/  stickysession=ROUTEID   nofailover=On
#负载均衡控制台，通过http://localhost/balancer-manager 访问
<Location /balancer-manager>
    SetHandler balancer-manager
    Order Deny,Allow
    Allow from all
    #Allow from localhost
</Location>
```

3.httpd.conf引入步骤2中的配置文件

```
include conf/balance.conf
```

4.重启Apache

5.附赠测试session保持的jsp页面一枚

```
<%@ page contentType="text/html; charset=GBK" %>
<%@ page import="java.util.*" %>
<html><head><title>Cluster Test8002</title></head>
<body>

<h3>this is tomcat3!!</h3>
<%
  out.println("<p> SESSION ID : " + session.getId()+"</p>");
  String name = request.getParameter("name");
  if (name != null && name.length() > 0) {
     String value = request.getParameter("value");
     session.setAttribute(name, value);
  }
    out.print("<table border = '1'>");
    out.print("<tr><th>session key </th><th>session value </th></tr>");
    Enumeration<String> names = session.getAttributeNames();
    while (names.hasMoreElements()) {
        String key = names.nextElement();
        String value = session.getAttribute(key).toString();
        System.out.print(key + " --- " + value);
        out.print("<tr><td>"+key+"</td><td>"+value+"</td></tr>");
   }
   out.print("</table>");
%>
<br />
<form action="testCluster.jsp" method="post">
session key :<input type=text name="name">
session value:<input type=text name="value">
<input type=submit value="添加">
</form>
</body>
```

</html>

有时候，换一种思路，也许你会有新的发现，晚安。